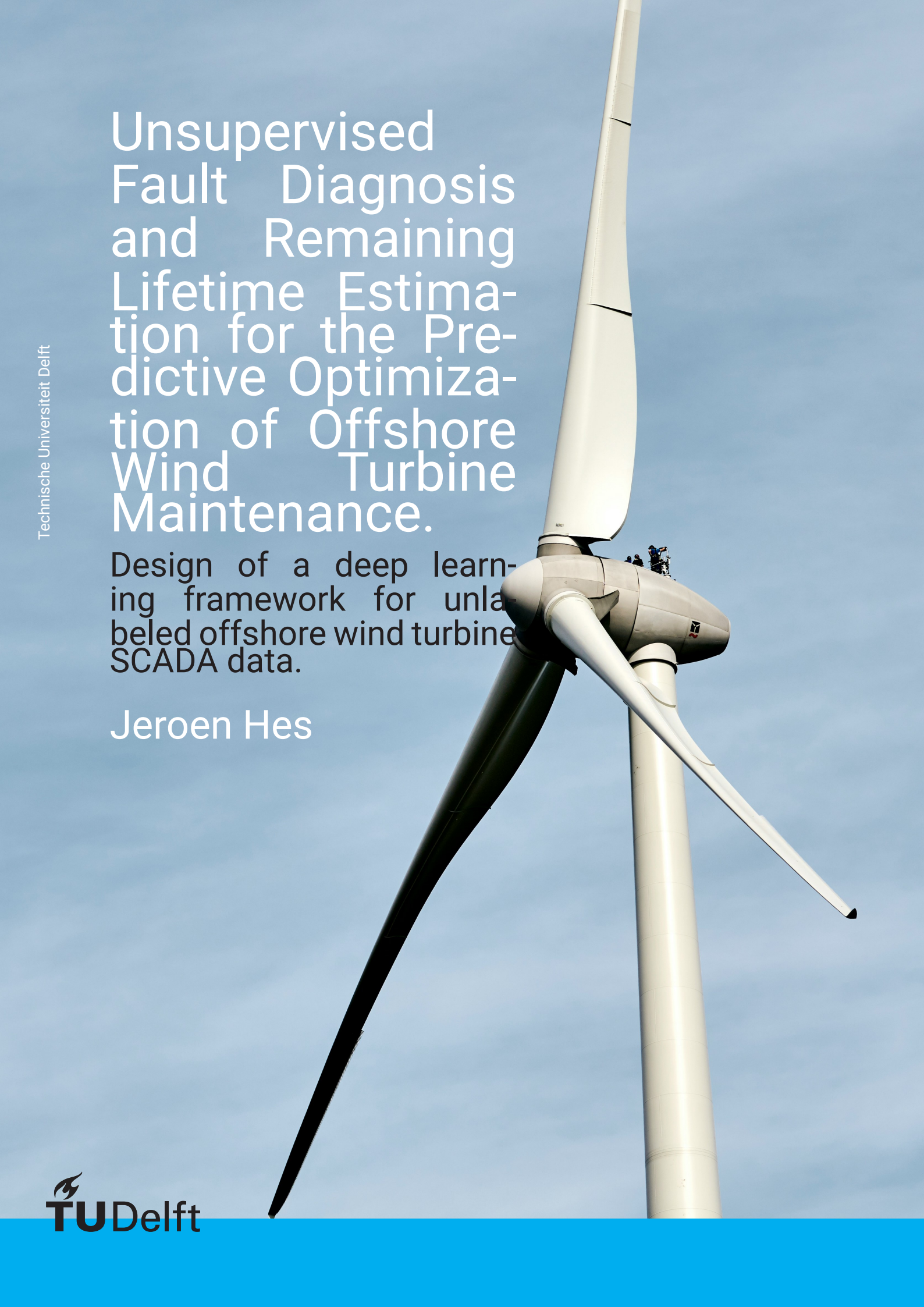


Unsupervised Fault Diagnosis and Remaining Lifetime Estimation for the Predictive Optimization of Offshore Wind Turbine Maintenance.

Design of a deep learning framework for unlabeled offshore wind turbine SCADA data.

Jeroen Hes

Technische Universiteit Delft

TU Delft

Unsupervised Fault Diagnosis and Remaining Lifetime Estimation for the Predictive Optimization of Offshore Wind Turbine Maintenance.

Design of a deep learning framework for unlabeled offshore wind turbine SCADA data.

by

Jeroen Hes

In partial fulfillment of the requirements for the degree of

Master of Science
in Mechanical Engineering

at the Department Maritime and Transport Technology,
Faculty Mechanical, Maritime and Materials Engineering,
Delft University of Technology.

To be defended publicly on Wednesday, 10-9-2025, at 14:00.

Student number:	4698819	
MSc Track:	Multi-Machine Engineering	
Report number:	2025.MME.9077	
Project duration:	June 10th, 2024 – September 10th, 2025	
Thesis Committee:		
- Chair:	Prof. dr. X. Jiang	TU Delft
- Daily Supervisor:	M. Borsotti	TU Delft
- Committee Member:	B. Font	TU Delft

TU Delft

Abstract

For offshore wind turbines (OWTs), effective maintenance decision-making depends on the timely and intelligent anticipation of developing faults. Without requiring the installation of additional sensors, failure-related information can be extracted from the widely available Supervisory Control and Data Acquisition (SCADA) system. This thesis presents an integrated deep learning framework designed to interpret high-dimensional, unlabeled, and often low-quality SCADA data for fault diagnosis and Remaining Useful Life (RUL) estimation, as illustrated in Figure 1.

The framework identifies historical failure events through reconstruction-based anomaly detection and the construction of a health indicator. By clustering detected anomalies, associated failure modes are inferred, allowing classification of future fault types. Using the estimated moments of failure as guidance, it then learns degradation trends in the reconstruction feature space and performs RUL prediction.

Given the complexity of offshore environments and the unpredictable nature of wind turbine faults, the framework is first validated in a controlled setting using NASA's C-MAPSS simulated aircraft engine dataset. The results are competitive and align well with those reported in related studies. Subsequent application to real-world OWT SCADA data demonstrates the practical feasibility of the approach. However, challenges such as data imbalance, obscured features due to SCADA data quality issues, and propagation of errors between model components complicate implementation and reduce prediction reliability.

Despite these challenges, the proposed framework successfully extracts health-relevant insights, enabling predictive maintenance optimization and contributing to more informed data-driven decision-making in offshore wind operations.

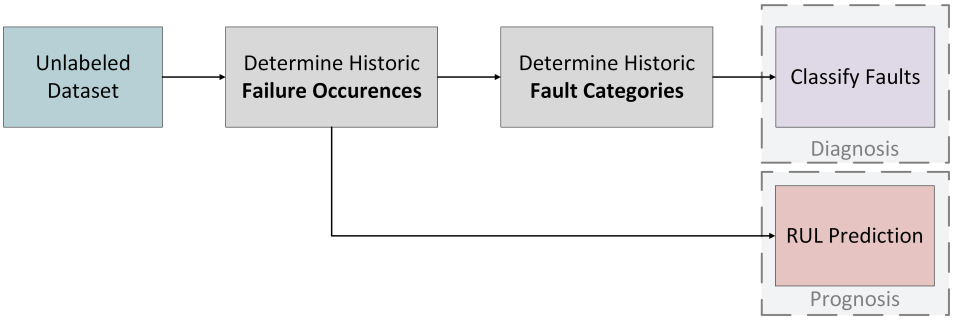


Figure 1: Key steps to achieving diagnosis and prognosis of unlabeled data.

Contents

1

Wind Turbine Predictive Maintenance

Introduction

As global demand for renewable electricity accelerates, offshore wind energy plays an increasingly critical role in the transition towards sustainable energy [1]. To increase the effectiveness of energy generation, progressively larger turbines are installed in deeper waters and farther out to sea, where higher loads, increased failure rates, and challenging, expensive repairs contribute to the increased cost of Operations and Maintenance (O&M) [2, 3].

Due to decreased reliability and increased expenses due to unexpected failures, O&M comprises 25%-50% of the total energy generation cost of Offshore Wind Turbines (OWTs) [4]. In turn, this accounts for a significant 20-35% of the total investment, making maintenance optimization crucial for cost-effective operation [5–7].

This is achieved through the application of intelligent maintenance strategies that aim to minimize the frequency of costly on-site visits (Figure 1.1), and increase the reliability of critical components such as the electrical and control systems, gearbox, generator, and blades, collectively responsible for 90% of repair and replacement costs over a turbine's lifetime [8, 9]. When detected early, flaws can be repaired without complex maintenance operations, before they cause significant losses to the yielded power [10]. Undetected fault development, on the other hand, can cause irreversible structural damage, potentially propagating to other systems. In such cases, repair requires large lifting vessels and the manufacturing of new components, further driving up maintenance expenses [11, 12].

Figure 1.1: Wind turbine blade inspection, from [13].

1.1. Wind Turbine Maintenance Optimization

The optimization of OWT maintenance involves the intelligent management of resources, risks, and costs [14–17]. If done right, this can lead to substantial reductions in O&M costs [18–21]. Scheduling optimization in offshore maintenance must account for constraints such as weather, equipment availability, and time. Key strategies in this field include: (1) Minimizing production losses by scheduling maintenance during periods of low energy yield, as indicated by weather forecasts or market prices [22]; (2) Reducing excessive spare stock [23]; (3) Optimizing routing for service vessels and technicians [24, 25]; (4) Consolidating maintenance and repair tasks to reduce downtime and costs [26]; and (5) Enhancing fleet management [27].

O&M decisions typically rely on traditional rule-based scheduling approaches, often based on fixed age, usage, or power generation thresholds. Lacking the flexibility required to model real-time operational variability, these indicators fail to accurately reflect the actual system condition, resulting in poorly timed maintenance tasks and complicating effective scheduling decision-making [28, 29].

1.2. Wind Turbine Condition Monitoring

Instead, maintenance optimization is facilitated by the timely and intelligent anticipation of developing faults [30]. Through the application of online sensing measurements, generalized by a Health Indicator (HI), Condition Monitoring (CM) provides a remote assessment of the current system health [31].

Abnormal changes in the collected data indicate developing faults, further analyzed to provide additional fault information related to failure mode and location [32, 33]. Advanced interpretation of data collected by CM methods can identify early indicators of faults and predict future degradation trends to estimate Remaining Useful Life (RUL) [34, 35]. As illustrated in Figure 1.2, the RUL is the difference between the current time and the point of failure, which occurs when the predicted health trajectory intersects a predefined failure threshold.

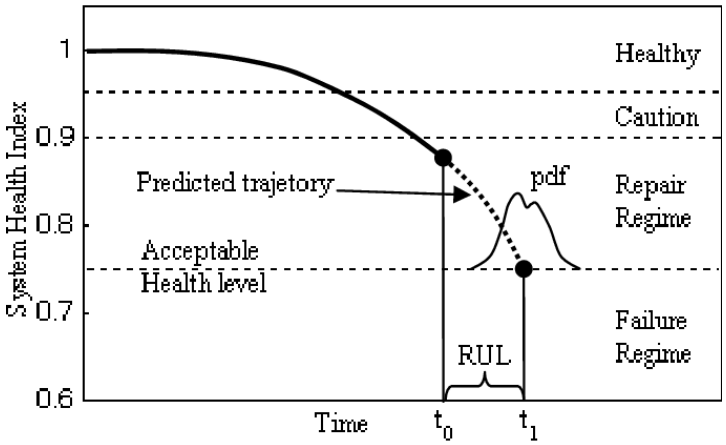


Figure 1.2: RUL defined as current time t_0 to its point of failure t_1, from [36].

The complex operational conditions of wind turbines complicate the use of traditional approaches and necessitate specifically designed health monitoring methods [37]. These either consider the installation of additional sensors for obtaining high-quality information, or analyze operational data provided by Supervisory Control And Data Acquisition (SCADA) systems of the turbine [38–40].

1.2.1. Sensor-based CM

Initial limitations of the SCADA system have encouraged the use of dedicated sensors explicitly designed for health monitoring. These standalone, condition-specific devices measure key variables such as vibration, oil quality, temperature, strain, and acoustic emissions [41]. Unlike SCADA systems, they can provide high-frequency measurements with greater precision, offering a more detailed and responsive view of component health [39, 42–46].

Such systems are particularly valuable in detecting rapid or subtle changes that precede faults. However, complex loads, varying rotational speeds, and harsh environmental conditions make accurate sensing a challenging and expensive task [21, 47, 48]. For OWTs, whose energy yield is relatively low compared to fossil or nuclear plants, the long payback period further limits the economic appeal of large-scale sensor deployment.

1.2.2. SCADA-based CM

Mainly for monitoring the overall performance of wind turbines and their major components, most wind turbines use SCADA systems [49]. Due to its widespread availability, avoiding the financial and logistical burdens associated with dedicated sensor systems, SCADA data poses an attractive alternative for CM.

A standard SCADA system provides critical information at both the wind turbine and wind farm levels. This includes operational data, such as the produced power, wind speeds, electrical parameters, component temperatures, and occasionally vibration or oil debris monitoring data. SCADA systems also typically log availability, status changes, error codes, and component activations, along with extra wind measurements from a meteorological tower.

Newly developed data-driven methods enable the extraction of health-related information through intelligent analysis of SCADA sensor trends. Most component-specific cases demonstrate that indicators for developing faults are visible at least two months before a failure turns critical [50–54]. Some studies highlight the potential of using the alarm system for CM, but data errors and alarm inaccuracies complicate the reliable detection of faults [55, 56].

1.2.3. Challenges of SCADA Data

The success of data-driven condition monitoring depends heavily on data quality. Although SCADA systems offer a cost-effective and capable solution for CM, they present several limitations that hinder reliable fault detection and maintenance optimization [57, 58]. Commonly acknowledged limitations include: (1) Low-frequency 10-minute sampling, which may miss short-duration events crucial for early fault detection; (2) Data acquisition errors occurring as as NaNs, zeros, and outliers; (3) Lack of data structure standardization [59]; and (4) Unreliability of written maintenance records and alarm data [60–63].

Additionally, the definition of a “normal” operating state, to which anomalies can be compared, is highly variable, influenced by changing environmental and operational conditions. This variability complicates the distinction between genuine faults and harmless internal or external influences. As a result, reliable fault detection requires both a sufficient amount and quality of failure samples—yet SCADA datasets are typically scarce in such samples and exhibit a strong class imbalance, with healthy operating data vastly outnumbering abnormal or faulty cases. [63–65].

Given these quality issues, SCADA-based CM approaches benefit from considering unsupervised data interpretation that avoids reliance on frequently absent, imbalanced, and unreliable anomaly labels.

1.3. Related Work & Research Gap

The intelligent anticipation of faults allows for the cost-effective scheduling of maintenance activities, while increasing turbine availability and lifespan, offering a more cost-effective maintenance solution [66–68]. As a result, predictive maintenance (PdM) has become a significant research focus in wind turbine O&M [43, 57].

While significant advances have been made on both CM and O&M optimization aspects, these areas have been primarily studied in isolation, where integration between CM methodologies and the execution of maintenance actions is underdeveloped [69]. Therefore, implementation of an integrated framework, that translates sensor signals into a communicative output through data-driven inferences (Figure 1.3), is an essential step for streamlining O&M.

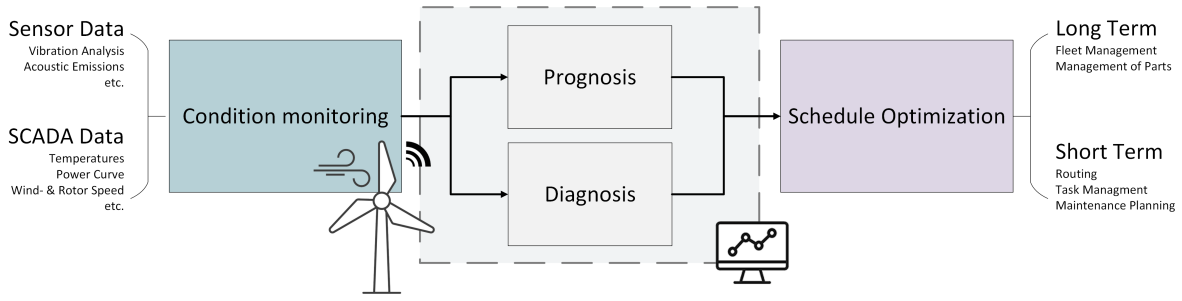


Figure 1.3: Schematic overview of an intelligent predictive maintenance optimization process.

This is also recognized in other fields, such as aircraft maintenance, where Camci et al. formulated an integrated O&M planning approach using current and forecast health information [70]. Similarly, Vianna et al. initialize maintenance optimization based on data trends and future wear values [71]. Nguyen et al. propose a dynamic predictive maintenance framework that provides failure probabilities of different time intervals based on sensor measurements [72]. In addition, Chen et al. and Mitici et al. utilize data-driven RUL prediction for O&M scheduling [73, 74], while Zhuang et al. and Lee & Mitici formulate a Deep Learning problem that considers prognosis as a basis for maintenance optimization decisions [75, 76].

Taken together, these studies highlight the potential of combining data-driven diagnosis, prognosis, and scheduling to improve O&M planning. Building on these advances, early wind turbine-specific works have also emerged. Table 1.1 summarizes their contributions to the diagnosis and prognosis of OWTs, illustrating how these methods can translate into actionable scheduling optimization strategies [77, 78].

Source	Labels	Data Type	Diagnosis	Prognosis
Bogoevska et al. [79]	Labeled	SCADA	Yes	No
Lei et al. [80]	Labeled	SCADA	Yes	No
Santolamazza et al. [81]	Unlabeled	SCADA	No	Yes
Udo & Muhammad [82]	Unlabeled	SCADA	No	Yes
Gomes et al. [83]	Labeled	SCADA	No	Yes
Lutzen et al. [84]	Unlabeled	Sensor	No	Yes
Li et al. [85]	Labeled	Sensor	Yes	No
Nuvvula et al. [86]	Labeled	SCADA	No	Yes
Rajaoarisoa et al. [87]	Unlabeled	SCADA	Manual	Yes
Shah et al. [88]	Labeled	SCADA	No	Yes
He et al. [89]	Partially Labeled	Sensor	Manual	Yes
Qin et al. [90]	Labeled	Sensor	Manual	Yes

Table 1.1: Related Data-driven Studies supporting Maintenance Optimization. Sorted by year.

By reviewing Table 1.1, several shortcomings of the current state of research can be identified: (1) studies are typically component-specific, especially sensor-based studies; (2) most focus only on fault identification rather than full data-driven diagnosis; (3) diagnosis generally relies on labeled failure examples; and (4) prognosis is often limited to early alarms, without a predicted RUL trajectory.

1.4. Scope of this Thesis

To address these gaps, this thesis develops and evaluates an integrated framework for fault detection, diagnosis, and RUL prediction. The framework is deliberately designed to operate on cost-effective and widely available SCADA data, making it more practical for large-scale deployment in offshore wind farms. Unlike most existing approaches, it follows an unsupervised learning strategy, enabling fault identification and degradation tracking without reliance on extensive labeled failure examples. This allows for the identification of failures across multiple turbine components while predicting their future degradation curve.

Given the complexity of offshore environments and the unpredictable nature of wind turbine faults, the proposed framework is first validated using a controlled environment. NASA's C-MAPSS simulated aircraft engine dataset provides a suitable testbed for evaluating fault diagnosis and RUL prediction techniques under known conditions. Additionally, as the dataset is widely used in health monitoring research, it offers a strong baseline for comparison against other studies.

Insights gained from this controlled environment are then applied to a real-world SCADA dataset. This case study demonstrates the identification, diagnosis, and prediction of Gearbox, Generator, Transformer, Bearing, and Hydraulic faults. In doing so, this work contributes to the ongoing transition toward intelligent, cost-effective O&M planning in the renewable energy sector.

1.4.1. Thesis Structure & Research Questions

To achieve the fault diagnosis and prognosis of WT failures based on SCADA data, a main research question is formulated as follows: *How can an Offshore Wind Turbine SCADA dataset be interpreted to detect and diagnose developing failures and evaluate Remaining Useful Life?*.

Several sub-questions are formulated, starting with a literature study in chapter 2. Here, the strengths and weaknesses of available approaches are evaluated before making a selection:

- *What data-driven methods are available for health monitoring and prognostics?*
- *Which method or collection of methods is most suitable for the interpretation of unlabeled OWT SCADA data?*

A direction is chosen from these methods, which is further described by illustrations and mathematical expressions, to reach the final design of the selected model in chapter 3:

- *How can the selected methods be applied to OWT PdM?*
- *What is the architecture of the chosen approach?*

In chapter 4, a benchmark case-study, NASA's widely known C-MAPSS dataset, is implemented to verify the results and compare with related studies:

- *How does the performance of the proposed Predictive Maintenance framework compare to state-of-the-art methods on a benchmark dataset in terms of reliability and accuracy?*

Considering a real-world offshore wind turbine SCADA dataset, model functionality is demonstrated in chapter 5:

- *How does the proposed framework perform when tested on offshore wind turbine SCADA data?*

Finally, this work concludes with chapter 6, covering the main findings and considerations for future work.

A shortened academic paper version of this thesis is included in Appendix A. Then, supporting the contents of this thesis, additional information, explanations, justifications, or figures are given in Appendix B and Appendix C, for the simulated and real-life case-study, respectively. An overview of all relevant model parameters with a short description is given in Table C.4.

2

Data-driven Methods for Health Monitoring and Prognosis

Literature Study

Anticipating developing faults is crucial for initiating effective maintenance decision-making. Based on a detected anomalous state, data-interpretation methods can be applied to determine the most likely fault source and estimate the remaining useful lifetime (RUL) of the machine.

In complex systems such as OWTs, modeling system behavior in physical system descriptions or manually extracted features is challenging [91, 92]. Therefore, data-driven approaches are applied to utilize historical and real-time data for the identification of patterns, trends, and relationships to make health-based assessments [93, 94]. This chapter discusses how such methods have been applied in offshore and other industrial contexts. It poses the research question: "*What data-driven methods are available for health monitoring and prognostics?*". In light of the unbalanced, dynamic nature of offshore wind operations and the complexities of SCADA-based inference, this chapter explores a second research question: "*Which method or collection of methods is most suitable for the interpretation of unlabeled OWT SCADA data?*".

Conventional approaches to data-driven health evaluation often rely on predefined models or statistical assumptions. These include methods that presume linear relationships and known data distributions, limiting their effectiveness in capturing the complex and dynamic behavior of mechanical systems. Therefore, systems like OWTs require techniques capable of extracting high-level features—capturing relationships between sensor signals, failure modes, and varying operational conditions.

Increasingly flexible to operational variations and disturbances prevalent in offshore data, Machine Learning (ML) offers a powerful approach to analyzing signal relationships in large datasets of complex systems [65, 95–100]. Widely discussed in academics, ML methods are typically categorized into supervised approaches (regression and classification) and unsupervised approaches (clustering), as illustrated in Figure 2.1 [101].

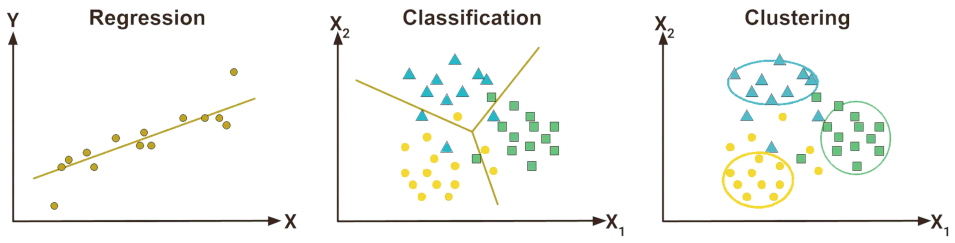


Figure 2.1: Three most common ML techniques [102].

Numerous traditional machine learning methods have been developed for supervised ML tasks, including decision trees, k-nearest neighbors, random forest, linear regression, and Support Vector Machines (SVMs) [34, 103–105]. However, their effectiveness depends heavily on the quality of the manually engineered input features.

To overcome this limitation, more advanced neural network (NN)-based deep learning models have been developed. These models typically outperform traditional machine learning, statistical, and physics-based approaches by handling high-dimensional, nonlinear problems without relying heavily on expert knowledge or manual feature extraction [106–108]. As a result, they excel at learning complex patterns from sensor data, making them particularly effective for the fault diagnosis [109–111] and prognosis [112–114] of mechanical systems.

Method selection largely depends on the application. For PdM and time series analysis, the availability of labels primarily influences this process, as they can serve as failure examples. Based on failure examples, the model learns to recognize data based on learned historical patterns, which is a supervised, classification operation [115]. In the absence of labels, other approaches are applied to solve the task, including unsupervised clustering or reconstruction tasks, learning an intended purpose through comparative methods [116]. An overview of various NN categories and their typical supervised or unsupervised applications is given in Figure 2.2.

This chapter reviews the functionality of these methods and their implementation in offshore wind and other industrial fields to highlight their respective opportunities and challenges. The state-of-the-art reveals the differences in the suitable purpose of studied methods, revealing a trend towards a collective implementation of methods, where each selected approach can be optimized explicitly for an intended purpose, setting the foundation for the proposed framework for anomaly detection, diagnosis, and prognosis.

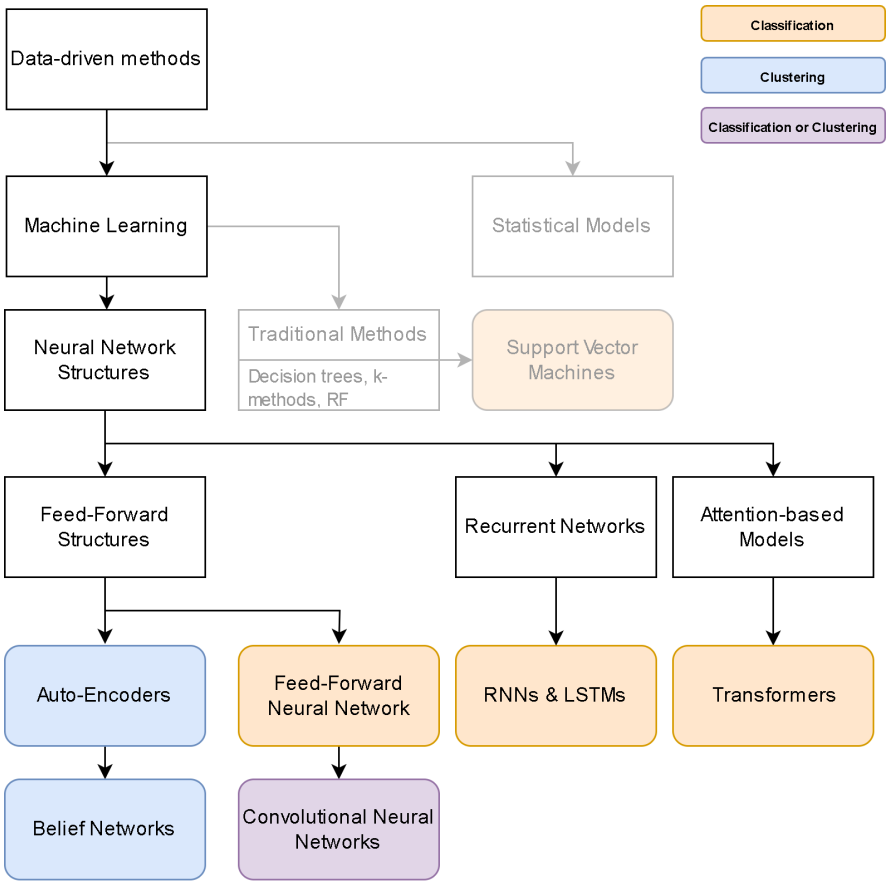


Figure 2.2: Flowchart of ML Types and their typically associated task.

2.1. Neural Networks & Deep Learning

The flexibility and adaptability of Neural Network (NN) architectures have made them increasingly popular for data interpretation tasks in predictive maintenance and health monitoring. An NN comprises interconnected layers of computational units (neurons) that transform input signals through learnable parameters to extract patterns or generate predictions [117].

The representational capacity of an NN can be enhanced by increasing the depth of the network, resulting in deep neural networks (DNNs). Such architectures learn hierarchical representations from raw data, where earlier layers typically extract low-level features and deeper layers capture progressively more abstract concepts [118]. This hierarchical feature learning reduces reliance on domain-specific, manually engineered features, thereby streamlining the development of robust Remaining Useful Life (RUL) prediction models [34].

As a result, there has been a clear shift towards deep learning (DL) methods, which have demonstrated strong performance in fault detection and prognostics tasks across many engineering domains [97, 119].

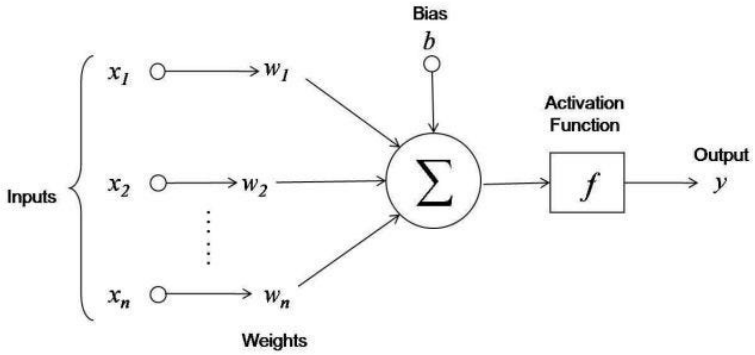


Figure 2.3: Overview of a single neuron, from [120].

2.1.1. Feed-Forward Neural Networks

The feed-forward neural network (FFNN) is the most fundamental NN architecture, in which information flows in a single direction from input to output. As illustrated in Figure 2.3, each neuron applies a weighted linear transformation to its inputs, adds a bias term, and passes the result through a nonlinear activation function. These parameters are iteratively updated during training to minimize a predefined loss function, thereby allowing the model to capture patterns present in the data.

FFNNs can effectively model simple relationships without requiring domain-specific feature knowledge, allowing for detecting and predicting WT component failures based on SCADA data [65, 121, 122]. However, they are limited in handling temporal dependencies, complex data sources, or dynamic systems involving multiple operating conditions.

Therefore, more advanced NN structures are employed for improved analytical power or time series tasks, such as in condition monitoring and RUL prediction. These include recurrent structures, convolutional layers, and transformer-based models. The following sections elaborate on the fundamental architectures most relevant for modeling SCADA data in the context of offshore wind turbine health monitoring. A visual summary of these methods is adapted from [123] and given in Figure 2.4.

2.1.2. Auto-Encoders

Autoencoders (AEs) are unsupervised neural architectures designed to learn compact, informative representations of input data during reconstruction. The encoder maps the input into a lower-dimensional latent space, while the decoder reconstructs the original input from this representation. This compression forces the network to retain only the most salient features, making AEs valuable for dimensionality reduction, anomaly detection, and feature extraction from complex sensor datasets [34, 124].

By comparing the reconstructed output with the original input, reconstruction errors can be used as a Health Indicator (HI), demonstrated by Wang et al. and Zhao et al. for blade, bearing, and gearbox failure [125, 126]. Renström demonstrates this AE-based HI allows for fault detection in many different components of a WT [127].

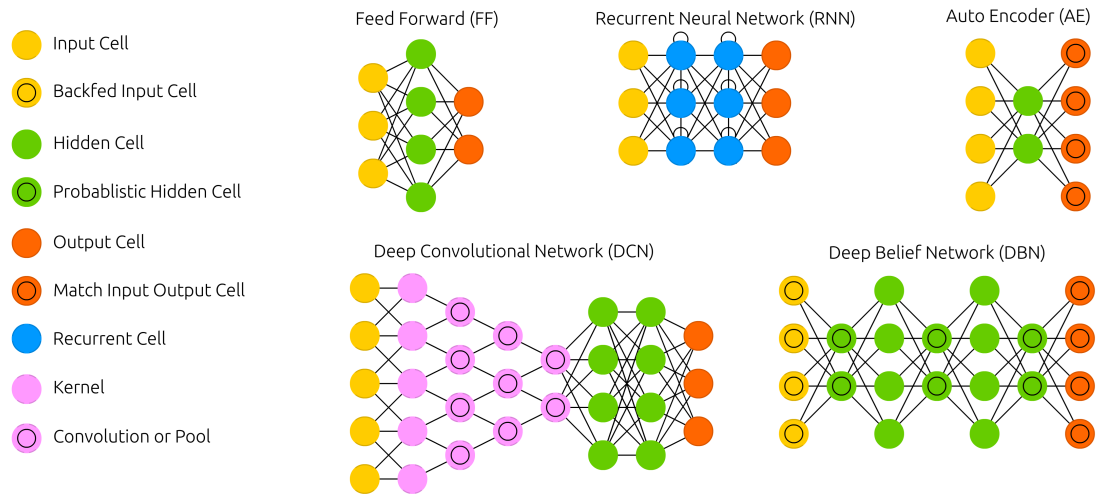


Figure 2.4: Illustrative interpretations of the neural network structures mentioned in this chapter. Adapted from F. van Veen of the Azimov institute [123]. Backfed input cell indicates a bidirectional connection to the input nodes. Recurrent Cells can be both memory and gated memory cells. The width and depth of these structures can be varied based on the application.

AE variants incorporate more elaborate layers, such as Long Short-Term Memory (LSTM) networks. These recurrent layers allow for the modeling of sequential data, capturing temporal dependencies critical for detecting anomalous trends [128–130], and RUL prediction [119, 131].

At the same time, the rich health-related signals in the latent space or reconstruction signal can serve as an intelligent feature extraction step [132, 133]. For instance, Wu et al. predict WT blade damage based on the reconstruction error of the original unlabeled SCADA signal [134]. Similarly, Chen et al. propose applying an AE for interpretation of unlabeled data, which is used to train a BiLSTM to predict RUL [135].

2.1.3. Belief Networks

Restricted Boltzmann Machines (RBMs) are probabilistic, generative models that learn a joint distribution over input features. RBMs have been employed for feature extraction and dimensionality reduction, particularly in settings with noisy sensor data, due to their capacity to model uncertainty [124, 136]. Applied to bearings, Deutsch et al. demonstrate regression analysis of these features allows for RUL predictions [137].

Stacking multiple RBMs yields a Deep Belief Network (DBN), which captures increasingly abstract feature representations across layers. This makes DBNs well suited for modeling or learning normal behavior of WTs from SCADA data, enabling the fault detection of, for example, bearing, gearbox, and generator, demonstrated by [138–140].

For explicitly temporal modeling, Hidden Markov Models (HMMs) remain widely used. As a form of dynamic Bayesian network, HMMs are adept at modeling transitions between discrete health states, thereby offering interpretable probabilistic insights into system degradation trajectories [141].

While Belief networks are capable models that can be used for health monitoring tasks, they require elaborate training, struggle with long-term dependencies, and are vulnerable to disturbances and noise [114]. For this reason, RBMs are more commonly applied in generative or transfer learning tasks. Auto-encoders tend to be the more popular approach in time series analysis.

2.1.4. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) extend the feed-forward architecture with convolutional and pooling layers that enable local feature extraction across spatial or temporal dimensions [142–144]. Convolutional layers apply learnable kernels in a sliding-window manner, detecting patterns such as shapes, trends, or anomalies in time series signals. Pooling layers reduce the dimensionality of intermediate representations while retaining salient features, which are subsequently passed to fully connected layers for task-specific processing. Stacking multiple convolutional–pooling blocks enables hierarchical feature extraction.

Indicators that relate to failure mechanisms can occur as short-term spikes or changes in the signal, or as gradual permanent variation from a normal state. By varying the kernel-based operation of the CNN across

layers, we can obtain equal responsiveness to short- and medium-term features, which makes CNNs an effective tool for time series tasks.

While considering smaller samples of the dataset at a time, CNNs struggle to fully capture the long-term temporal relationships that characterize slow fatigue-based failure mechanisms. That is why Xiang et al., Kong et al., as well as Sun et al. propose a combination of machine learning methods that applies the analyzing capability of CNNs to LSTM-based temporal modeling for effectively learning health-based representations in WT SCADA data [51, 145, 146].

2.1.5. Recurrent Neural Networks

To effectively model time-dependent behavior in sequential data, Recurrent Neural Networks (RNNs) deploy connections between nodes to form cycles. This allows the network to maintain an internal state or memory, enabling it to consider not only the current input but also the historical context provided by previous time steps, making LSTMs well-suited for RUL estimation using sensor data [147]. Sequential approaches are capable of extracting long-term data relationships, showing promising results in SCADA data-driven regression [148] and anomaly detection [82].

To mitigate difficulties in learning long-range dependencies, architectures such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) have been developed. These models use gating mechanisms to control the flow of information, selectively retaining or discarding past information during training [149–151]. LSTMs use distinct input, output, and forget gates, whereas GRUs adopt a streamlined gating structure, often achieving comparable performance with reduced computational cost.

Sequential reasoning capacity can be improved by implementing Bidirectional LSTMs to capture context from both past and future observations [152–154], or increasing the number of layers [149, 155, 156].

2.1.6. Attention-based Models

Attention mechanisms were developed to allow neural networks to dynamically focus on the most relevant parts of the input when making predictions. By computing relevance scores, the model can selectively focus on the most informative elements, improving the model's ability to handle missing or noisy data, increasing flexibility, and reducing computational complexity [157–161].

Through increased computational efficiency of attention, Transformers have been designed to leverage an intricate structure of multiple network layers and self-attention mechanisms that enable the identification of relevant interactions between distant elements in a sequence [162]. This capability allows for the model to focus on both short- and long-term features, without relying on recurrence or convolution, making transformers highly adaptable [163–165].

In health monitoring, Transformers are used to analyze long-term dependencies from multiple sensor inputs, often combining them with an AE for denoising and feature extraction in RUL prediction tasks [166–169].

Although transformer networks have shown promise in power forecasting [170–172], the development of transformers for Wind PdM remains behind. As of this writing, only a few studies have applied transformers for wind turbine CM. Zhao et al. utilized transformers for gearbox fault detection through predicted temperature [173], while Zheng et al. implemented a semi-supervised anomaly detection method using only a small amount of labeled data [174]. Results show the transformer-based prediction model can effectively extract the temporal dependence among multivariate time series for an unsupervised failure warning system.

2.2. Method Selection

Given the high-dimensional nature of SCADA data, the absence of anomaly labels, and the unpredictable nature of OWT conditions, fault detection, diagnosis, and prognosis tasks are designed step-by-step to be able to verify and optimize each result specifically. This allows for a deeper understanding of system capabilities, reduced complexity of the task, optimal matching of methods, and increased solution precision [175].

While summarizing the findings of this literature study, suitable data-driven methods are selected. The selection process considers method capability, involving accuracy and resilience to OWT PdM challenges, such as robustness to noise, operational conditions, and unknown failure modes, but also the feasibility of implementation and effectiveness of methods across fields.

Combining the identified suitable methods in a framework allows for effective extraction of failure information required for informed maintenance decision-making, providing a promising approach for addressing OWT challenges without relying on domain-specific expertise.

2.2.1. Anomaly detection

Detection of an anomaly is described as recognizing a pattern that deviates from expected behavior. In health monitoring, this can include a broad spectrum of irregularities that indicate defects and require maintenance action.

Separately designing a model that is trained to recognize faulty behavior can serve as an automated feature extraction step, responsive to deviations from a healthy state. To avoid the necessity of labeled anomaly samples, and increase robustness to varying and possibly unprecedented conditions and failure mechanisms, fault detection and health monitoring studies typically consider the application of unsupervised anomaly detection methods [176]. Even in the presence of labeled SCADA data, label reliability is not a guarantee, where unsupervised methods are more robust to possible uncatalogued anomalies or label inaccuracies [177].

Based on both wind and other studies, different variants of the AE structure pose a promising solution for the interpretation of unlabeled data. AEs enable unsupervised extraction of latent features from high-dimensional condition monitoring data, making them highly valuable for degradation tracking and health monitoring of mechanical systems. By integrating LSTM nodes into the encoder-decoder architecture, responsiveness to time-based features is improved, further increasing its capability in PdM applications [131, 134].

2.2.2. Diagnosis

Involving the further localization of a fault, data-driven fault diagnosis consists of the classification of an input dataset. Learning the characteristics of a set of historic failure examples allows for the supervised recognition of future defects if the data shows similar degradation patterns.

However, in offshore wind SCADA data, incomplete or missing labels, dataset imbalances, and sensors with limited responsiveness to early-stage fault developments argue for the implementation of additional unsupervised or semi-supervised clustering techniques into the fault diagnosis problem [178, 179]. These approaches improve the ability to identify faults when faced with limited or no fault examples, reducing the influence of dataset imbalances, enabling improved system reliability.

Fault Clustering

Dataset augmentations can improve the classification performance of unbalanced SCADA data. However, singular transformations risk inducing overfitting or amplifying existing imbalances, ultimately reducing the accuracy of fault diagnosis results [180, 181]. This highlights the need for more advanced strategies to enhance the discriminative power of the diagnosis model.

In computer vision and natural language processing fields, a proven solution is the learning of an input's alternate feature representation. To improve class distinctions, the raw input is mapped into an encoded space where samples of the same class are pulled closer together, while samples of different classes are pushed apart [182]. Early applications of such contrastive learning methods show improvements in the accuracy of fault diagnosis results of unbalanced and unlabeled SCADA data, and are beneficial to the accomplishment of classification and scheduling tasks [122, 183, 184].

For time-series data, a particularly effective extension of contrastive learning is Deep Embedded Clustering (DEC). Unlike traditional clustering methods that rely solely on distance or dissimilarity metrics, DEC leverages deep neural encoders to construct latent feature spaces that are more robust to high-dimensionality, noise, and imbalance [185, 186]. By optimizing a discriminative loss, the encoder learns representations that enhance cluster coherence [187–189].

Convolutional Neural Networks (CNNs) are frequently employed as the encoder in DEC due to their ability to capture both spatial features of different failure mechanisms and temporal patterns within the signal. CNN filters sliding over the input can detect gradual degradation trends as well as sudden changes, while preserving temporal relationships in the latent space. This makes CNNs highly effective for contrastive learning in the fault clustering domain.

When applied to SCADA data, each time step corresponds to a multivariate one-dimensional signal. In this context, 1D Convolutional Neural Networks (1D CNNs) provide a compact and efficient solution. They reduce computational cost while maintaining strong performance in multivariate applications with limited labeled data and high signal variability [190].

Classification
The classification task involves learning the relationship between the features embedded by the DEC model and the labels it has assigned. Then, when new samples are introduced in the future and embedded by the trained encoder, they are assigned the most likely class.

The contrastive encoding may produce cluster soft assignments that correctly capture fault labels on its own. However, when future data is introduced sequentially, the outputs can become unstable in cases such as: (1) limited data availability; (2) testing inputs containing only a single fault category; (3) the introduction of an unseen testing turbine; or (4) uncertain or overlapping failure mechanisms. In such situations, the addition of a final classification layer improves class separation and model robustness, while also enabling clearer reasoning and interpretation of the diagnosis.

Because of the relatively low complexity of this classification task, a standard feed-forward NN is applied. NNs provide a computationally efficient solution able to learn the characteristic differences between different clusters in the embedding, while minimizing the loss of information [105].

2.2.3. Prognosis
The prediction of future values is greatly supported by inherent mechanisms to model temporal dependencies. Commonly, RUL prediction methods follow supervised reasoning. By analyzing the similarity between current and historical run-to-failure data profiles, the degradation trend and corresponding point of failure can be estimated. By interpreting the output of the fault detection model for historic data and future values, degradation features can be linked to the inferred moments of failure. This data can be used to inform a time-sensitive model to obtain RUL prediction.

While Transformers are relatively new and unexplored for time series analysis, comparison of benchmark tests performed by Vollert et al. [97] with Transformer applications on the same dataset [163, 164, 167], show transformer-based methods consistently demonstrate improvements in terms of training speed, predictive performance, and reduced data requirements, enhancing temporal pattern recognition and improving RUL prediction accuracy.

2.3. Discussion: Uncertainty Quantification
While out of scope for this thesis, properly managing the inevitable uncertainty that arises when making predictions is crucial for effective risk management and maintenance decision-making [191–195]. In the face of disturbances and a lack of knowledge originating from the multitude of complex loads and external influences on a turbine, Uncertainty Quantification methods serve as a heuristic to produce a prediction, along with an indication of the reliability of their outcome. This assessment provides upper and lower bounds of a confidence interval that can serve as crucial decision values in the optimal scheduling of maintenance, providing a significant benefit in the scheduling of future maintenance tasks [73, 74].

By far the most popular approach to modeling uncertainty considers the application of Bayesian methods [194, 196, 197]. Bayesian deep learning has become the primary solution for uncertainty estimation in applications, where safety and robustness are crucial, providing a versatile tool for integrating stochastic reasoning into DL structures, increasing accuracy and robustness to over-fitting [198–200]. These advantages effectively overcome drawbacks originating from signal disturbances and data quality issues, increasing application frequency in a variety of WT PdM studies [119, 201–203].

3

Designing A Predictive Health Monitoring Framework

Methodology & Model Design

Informed proactive scheduling of maintenance activities improves resource management and equipment reliability, requiring the extraction of high-level features and relationships between sensor signals, failure modes, and varying operational conditions. In the absence of labeled anomalies and with the presence of noisy or incomplete data, a challenging learning process is created, necessitating specialized data interpretation steps to achieve effective outcomes.

Excelling at extracting the complex patterns hidden in sensor data, indicative of degradation, deep learning methods specifically optimized for their respective tasks are combined to create a unified framework for anomaly detection, fault diagnosis, and Remaining Useful Life (RUL) estimation. Alongside essential preprocessing measures, the four key steps illustrated in Figure 3.1 form the foundation of the proposed methodology, guiding the transition from input to actionable health-related predictions.

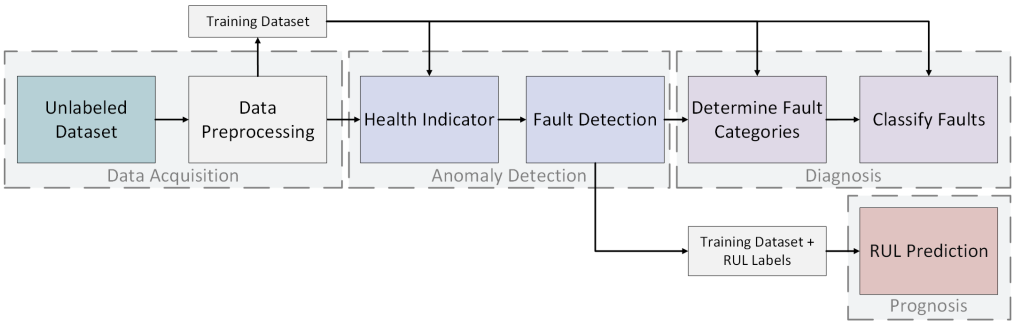


Figure 3.1: Four characteristic steps of obtaining a fault diagnosis and RUL prediction.

This chapter specifies the design and mathematical formulation of the proposed architecture, outlining the tools required for case-study implementation in the following chapters. Specifically, it addresses the research questions: *How can the selected methods be applied to OWT PdM?* and *What is the architecture of the chosen approach?*.

Following the sequence shown in Figure 3.1, preprocessing steps are discussed first, ensuring the input dataset is conditioned correctly for further interpretation. Then, the chapter continues by providing in-depth descriptions of the selected methods. Each section covers the architecture and training process of a method, followed by implementation guidance on generating and evaluating its outputs. After defining the structure and functionality of each step, the chapter concludes with an overview of the complete architecture of the proposed solution.

3.1. Dataset Preprocessing

Model input is considered as a multivariate SCADA dataset of an OWT at timestep $t \leq T$, at number of variables k, $\boldsymbol{x_t} = \{x_t^{(1)}, x_t^{(2)}, ..., x_t^{(k)}\}$, $t = 1, 2, ..., T$.

In data-driven applications, proper conditioning of the input signal is a critical step in ensuring optimal model performance and reliable results, including dataset normalization, removal of outliers, selection of informative sensors, and sequence generation.

3.1.1. Normalization

Neural Network operation is, in essence, generalizable as a series of multiplications. For this reason, they are susceptible to input signals with greatly varying ranges, distributions, and outliers. To ensure stable and efficient training of machine learning models, data inputs should be normalized to reduce these variations and reduce the risk of scaling and gradient-exploding problems. The choice of normalization method depends on the nature of the data and can significantly impact model performance. Three methods are illustrated in Figure 3.2 to demonstrate the effect of different scaling techniques.

1. **Z-score normalization**, also known as standardization, transforms data to have a mean of zero and a standard deviation of one. Beneficial in combining values of multiple ranges, this method preserves data distributions, making it suitable for algorithms assuming normally distributed inputs [135]. Given a data value x, data points can be normalized by subtracting the dataset mean μ and dividing by standard deviation σ, as follows:

$$x' = \frac{x - \mu}{\sigma} \tag{3.1}$$

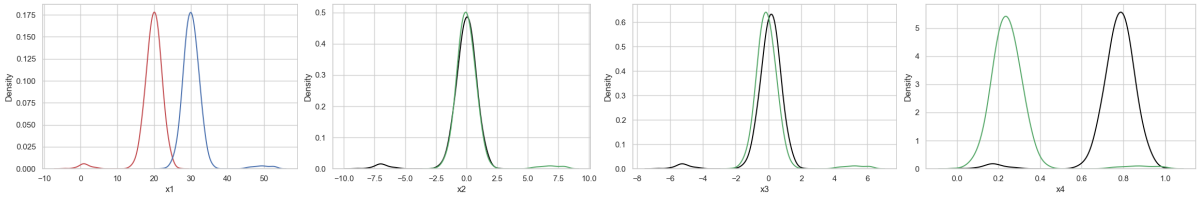
2. Standardization assumes features are normally distributed, and possibly distorts feature relationships if the data distribution is asymmetrical. In this case, **Min-max normalization** offers a more robust normalization technique [204, 205], popular in deep learning models [74, 134, 206, 207]). Datapoints are converted from their natural range into a standard range using minima and maxima as follows:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{3.2}$$

3. Because both scalers maintain the data structure, they are susceptible to outliers, which typically originate from data acquisition errors. To mitigate the impact of outliers, without distorting feature relationships during normalization, Sklearn's "**RobustScaler**" centers on the median and scales by the interquartile range (IQR) [208], as follows:

$$x' = \frac{x - \text{median}(x)}{\text{IQR}(x)} \tag{3.3}$$

To improve the interpretability of expressions, a normalized dataset $\boldsymbol{x} = \boldsymbol{x'}$ is used in the following sections.



(a) Before scaling (b) After Robust Scaling (c) After Standard Scaling (d) After Min-Max Scaling

Figure 3.2: Comparison of Normalization Techniques.

3.1.2. Sequence Generation

Due to the computational complexity of large datasets, such as in time series analysis, DL methods are typically designed to consider a set of fixed-length samples of the input at the same time. A popular approach to generating these sequences is the application of a sliding window technique, which breaks the data into smaller pieces of fixed length [209, 210]. By transforming a continuous time series into sequences, the time complexity is reduced, allowing for more detailed analysis and improved accuracy.

By introducing overlap in the time windows, a diverse range of samples can be generated, mitigating overfitting. As illustrated in Figure 3.3, given a multivariate time series, the sliding window technique generates overlapping sequences of length m using a stride or step size s. These sequences are collected in training, testing, and validation datasets and inserted into the model.

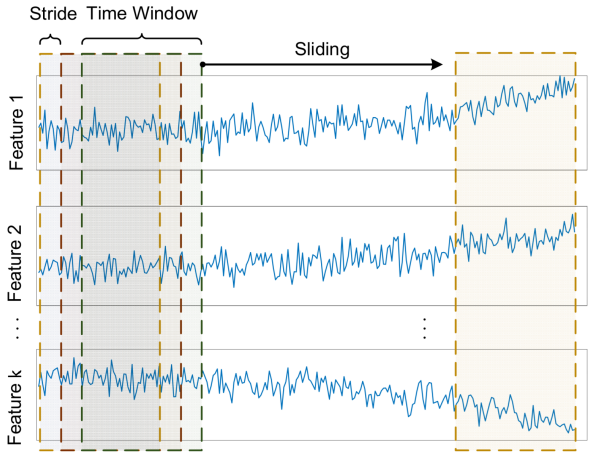


Figure 3.3: Visualization of the sliding window approach [210].

3.2. Anomaly Detection

Anomalies that occur in sensor data can be indicative of a deteriorating health condition of mechanical systems. Given a training input, an unseen test time series $\mathcal{T} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_T}\}$ is fed into the model, tasked to output a series $\boldsymbol{y} = \{y_1, ..., y_T\}$, where $y_t \in \{0, 1\}$ is used to denote healthy or unhealthy data at a timestep $t \in T$.

This section introduces the structure and training of AEs, followed by the construction of a Health Indicator (HI) from the AE Reconstruction Error (RE). After explaining how this signal is used for the detection of faults, HI evaluation metrics are introduced.

3.2.1. Long-Short-Term-Memory Auto-encoder

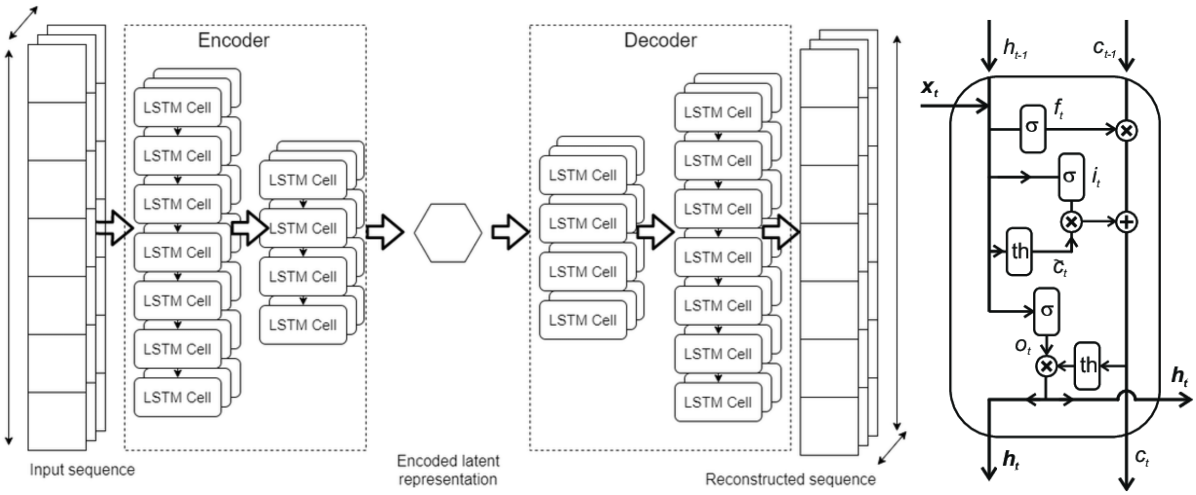
Operation of an AE involves compressing an input signal $\boldsymbol{x}$ by the encoder into a smaller latent representation. Then, this lower-dimensional dataset is converted by the decoder into an attempted reconstruction of the original $\hat{\boldsymbol{x}}$. During training, a training objective can be defined that minimizes the difference between the input time series and the reconstructed output, so that the encoder is forced to extract the most essential patterns in the data.

Both the encoder and decoder are built using other LSTM nodes, influencing the functionality and characteristics of the AE. Figure 3.4a shows how LSTM cells are used to construct AE layers, each with specified height and depth. Multiple sequences, corresponding to the number of selected features, with a length corresponding to our sliding window size, are fed into the model.

The LSTM cells manage the flow of information by gating mechanisms that allow it to retain important information for longer, while discarding irrelevant data [148, 207, 212]. The core of an LSTM cell, or otherwise known as a memory block, involves three key steps that determine the cell state or 'memory', illustrated in Figure 3.4:

1. Starting at the top, the **forget gate** learns to decide what information is filtered out from the cell state of the previous timestep, based on learned weight matrix W_C and bias b_i:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$



(a) Architecture of a LSTM Auto-Encoder, from [211]

(b) LSTM Cell, from [212]

Figure 3.4: Overview of LSTM cells in an AE structure.

2. The **input Gate** controls which new information is added to the cell state. This involves generating a candidate for the new cell state first, represented by $\tilde{C}_t$, containing potential information from the current input x_t and the previous hidden state h_{t-1}, which the cell might want to add to its memory. Then, the input gate vector i_t controls to what extent this candidate will be added:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

where, W_i, W_C, b_i, b_C are learnable weight matrices and bias terms.

3. The **output gate** decides what to output at the current time step, based on the current cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \cdot \tanh(C_t)$$

4. Finally, **updating** of the cell state is achieved by adding the element-wise multiplication of i_t and $\tilde{C}_t$ to the previous cell-state C_{t-1}, scaled by the forget gate f_t:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{3.4}$$

These equations ensure that each LSTM cell can control which information to retain, discard, and pass along, making it possible to model complex temporal dependencies.

3.2.2. Training AEs

An AE takes an observed sequence as an input vector $\boldsymbol{x}$ and then maps it to the hidden or latent representation $\mathbf{y}$, through mapping $\mathbf{y} = f_\theta(\boldsymbol{x}) = s_f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ [213]. Here, $\boldsymbol{W}$ is a weight matrix, $\boldsymbol{b}$ a bias vector, and s_f the encoder activation function that calculates the node output values. This representation is then mapped back into a reconstruction vector $\hat{\boldsymbol{x}}$, by a similar mapping $\hat{\boldsymbol{x}} = g_\theta(\boldsymbol{x}) = s_g(\boldsymbol{W'}\boldsymbol{y} + \boldsymbol{b'})$ [214].

When training the model, the collection of tunable parameters $\theta = \{\boldsymbol{W}, \boldsymbol{b}, \boldsymbol{W'}, \boldsymbol{b'}\}$ is adjusted such that $\boldsymbol{x} \approx \hat{\boldsymbol{x}}$. This can be achieved by minimizing a chosen loss function L. The choice of activation functions and L depends largely on the input domain range and nature and is usually chosen so that L is related to the RE, such as with a Mean Squared Error (MSE):

$$\mathcal{L}_{\mathrm{r}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{n}\sum_i (\boldsymbol{x_i} - \hat{\boldsymbol{x}}_{\boldsymbol{i}})^2 \tag{3.5}$$

To learn the optimal weights and biases of the encoder and decoder in θ, backpropagation is applied. The derivative of the loss function is calculated with respect to the output ($\frac{\delta L}{\delta \hat{x}_i}$), which defines how much the RE changes when the output value changes. This is the gradient, which, in the case of MSE, is characterized by:

$$\mathcal{L}(\theta; \boldsymbol{x}) = \frac{\delta L}{\delta \hat{x}_i} = 2(\hat{\boldsymbol{x}}_{\boldsymbol{i}} - \boldsymbol{x}_{\boldsymbol{i}})$$

Having obtained the gradients of the loss for the decoder's final layer, the error is propagated backwards, layer by layer, using the chain rule to compute how much each neuron's weight w has contributed to the final error ($\frac{\delta L}{\delta w}$) to update the weights with learning rate η:

$$w' = w - \eta * \frac{\delta L}{\delta w}$$

3.2.3. Health Indicator Construction

By training the AE on healthy data, the model weights and biases are adjusted to reproduce any new healthy data input accurately. If the input sequence contains unusual features, the model will not be able to reconstruct the sequence, resulting in a larger RE, serving as an indicator for data anomalies and health deterioration.

The RE can be interpreted as a quantified difference between the input and output of the AE, measured by methods such as Euclidean distance [128, 130, 215, 216]. Another measure for dissimilarity is the Mahalanobis distance (MD), which considers the covariance structure of the data, ensuring that highly correlated signals do not dominate the results [127, 217, 218].

Given a model input $\boldsymbol{x}^{i,j}, i \in [1,k], j \in [1,m]$, where k is the number of features and m is the size of the sliding window, the RE can be calculated per feature to obtain a reconstruction residual vector $\boldsymbol{r}_i = \boldsymbol{x}_i - \hat{\boldsymbol{x}}_{\boldsymbol{i}}$. Now, if C_x is the covariance matrix of the residuals, the MD for each feature i can be calculated as:

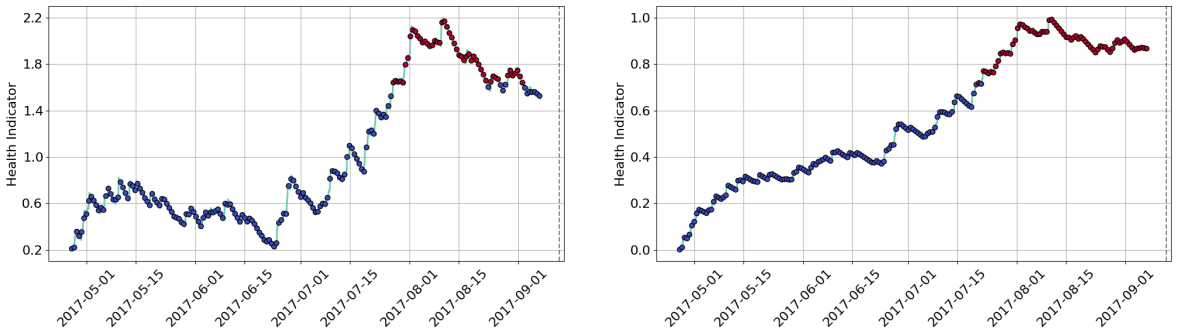
$$MD_i = \sqrt{\boldsymbol{r}_i^T C_x^{-1} \boldsymbol{r}_i} \tag{3.6}$$

Scaling & Normalization

To improve the stability of the HI, scaling and normalization are applied. These measures help ensure that different failure cases exhibit consistent degradation trends, while mitigating the influence of early-life anomalies, and enhancing the reliability of the fault detection threshold.

To reduce the influence of outliers in the LSTM-AE output, the Robust Scaler is used to scale the MSE and MD according to Equation 3. This improves the HI's relationship to a developing defect as illustrated in Figure 3.5.

To further increase the HI consistency, MinMax Normalization is applied as introduced in Equation 2. The validation dataset is passed through the model to identify normal reconstruction values that can be used to inform both scaling and normalization.



(a) No Scaling & Normalization

(b) With Scaling & Normalization

Figure 3.5: Constructed HIs without Robust-Scaler and Min-Max normalization fitted on the validation dataset.

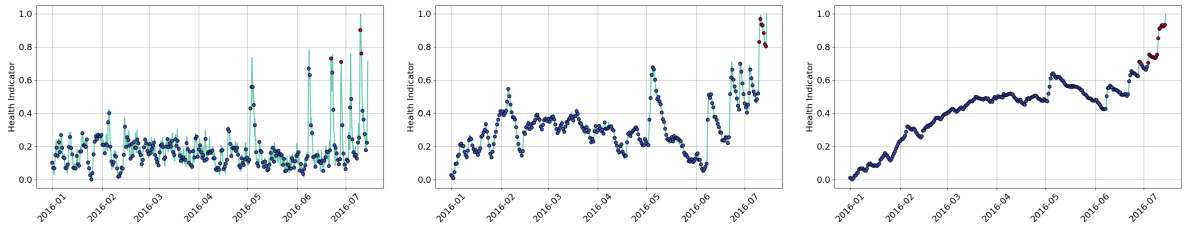
To reduce the impact of rarely occurring spikes, while limiting the amount of lost information of smaller deviations, the signal is smoothed [215]. An exponentially weighted moving average (EWMA) smooths the

input data while still accumulating a history of previous inputs [125, 127]. For a single feature, the EWMA is calculated as follows:

$$z_t = \lambda MD_t + (1-\lambda) z_{t-1} \tag{3.7}$$

Here, z is a smoothed projection of the MD, and λ is a constant, defining the weight of newly introduced values. A higher weight increases the responsiveness to short-term variations, at the cost of reduced smoothing. In the interpretation of SCADA data, changes that indicate system degradation commonly occur in the form of data peaks, instead of a more gradual and permanent change in state, necessitating more aggressive smoothing, as illustrated in Figure 3.6 [219].

Excessive smoothing can be detrimental, however, as an overly smoothed HI may lose its responsiveness to sudden faults and become more dependent on time progression rather than actual system behavior. This results in reduced sensitivity to new anomalies and a slower recovery from deviations unrelated to faults, ultimately undermining the reliability of the HI.



(a) $\lambda = 0.01$ (b) $\lambda = 0.001$ (c) $\lambda = 0.0001$

Figure 3.6: Three HI trends at varying EWMA smoothing factors.

3.2.4. Fault Detection

Finally, to decide between acceptable levels of deviations and failure, a fault threshold can be defined, functioning as an alarm decision criterion. Depending on the similarity in range of different deterioration profiles, a controllable fixed threshold might offer consistent generation of alarms at the right time. When facing fluctuations in failure profile due to factors such as imperfect maintenance, varying failure modes, or a lack of monotonicity, the health indicator might not be sufficiently similar between life-cycles, requiring an intelligent thresholding approach based on local data properties.

Fixed Threshold

A fixed threshold is defined by a constant, which is defined by an acceptable level of deviation, given by the root mean square of the average AE RE during normal operating conditions [126]. Given a scaling factor L that determines the sensitivity of fault detections, the alarm threshold T is defined as:

$$T = L \cdot \text{RMS}(\frac{1}{k}\sum_{i=1}^{k}(MD_i{}^{\text{val}})) \tag{3.8}$$

Adaptive Threshold

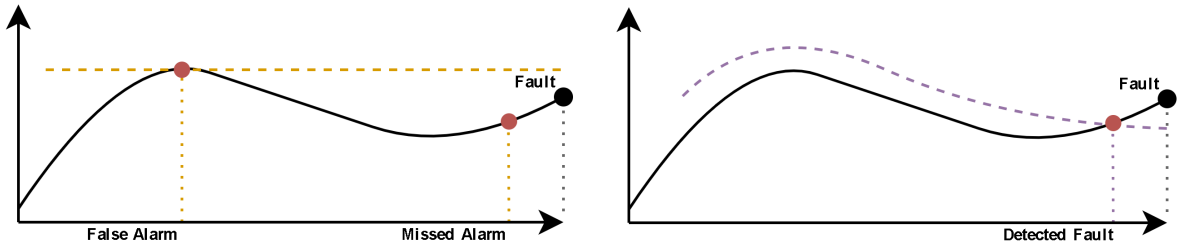
To address the limitations of traditional constant thresholds, an adaptive thresholding approach is proposed. Inspired by Liu et al., this adaptive threshold dynamically adjusts based on recent data trends, captured by the statistical properties of recent observations [220]. The adaptive nature of this threshold makes it particularly suitable for SCADA-based HIs that exhibit complex degradation trajectories due to turbine start-up, newly placed components, maintenance action, and varying fault severity. This ensures that failure detection is sensitive to gradual degradation trends while minimizing the occurrence of false alarms due to transient variations, as made visible in Figure 3.7.

At a certain time t, the threshold accounts for variations by computing a mean of the HI values μ_{t-w} and standard deviation σ_{t-w} over a predefined window of size w. Combined with a sensitivity parameter κ that controls the threshold's responsiveness to variations, the threshold value is determined as:

$$T_i = \mu_{t-w} + \kappa \cdot \sigma_{t-w} \tag{3.9}$$

Alarm Generation
An alarm system can be used to communicate a detected fault. This requires formulating a decision-criterion based on the smoothed HI value z and the determined fault threshold T, where an alarm is raised if:

$$\text{Alert} = \begin{cases} 1 & \text{if } z_t \geq T, \\ 0 & \text{otherwise.} \end{cases} \tag{3.10}$$



(a) Fixed Threshold (b) Adaptive Threshold

Figure 3.7: Comparison of fixed and adaptive threshold for a complex health profile

3.2.5. Evaluation: Health Indicator Performance Metrics

Ideally, a HI used to inform CM and prognostic systems should display three key qualities: monotonicity, prognosability, and trendability, which have been widely adopted in health monitoring studies (e.g. [97, 221, 222]). These metrics, each in the range [0, 1], describe the quality of the model output in prognostic applications and allow for comparable parameters across different studies. Formulations are proposed by Coble et al. and discussed below [223].

- *Monotonicity* represents the condition of being unchanging or unvarying in the increasing or decreasing trend of a feature. An accurate HI for a system that does not undergo maintenance should display a high monotonicity. In contrast, a low monotonicity means the feature is usually a non-desirable predictor for PdM applications. Calculating the level of monotonicity is defined as Equation 3.2.5, where the sign (sgn) function is applied to evaluate the degree of variation, M being the number of HIs corresponding to the amount of tested cycles, N_j the length of the j_{th} HI, $x_j(k+1)$ and $x_j(k)$ the HI values of timesteps $k+1$ and k, respectively.

$$\text{Monotonicity} = \frac{1}{M}\sum_{j=1}^{M}\left|\frac{1}{N_j-1}\sum_{k=1}^{N_j-1}\text{sgn}(x_j(k+1)-x_j(k))\right| \tag{3.11}$$

- *Trendability* measures the similarity of the HI trajectory between multiple run-to-failure data cycles. Trend similarity is typically described by the Pearson correlation, calculated by Equation 3.2.5, where x_j and x_k are the values of compared HIs, indexed by j and k. Studies typically calculate the minimum value of this correlation to determine the worst case. To compare general performance, a mean trendability value is also calculated in this work.

$$\text{Trendability} = \min_{j,k}\left|\text{corr}(x_j, x_k)\right|, \quad j,k = 1,2,\ldots,M \tag{3.12}$$

- *Prognosability* characterizes the dispersion of HIs between faults and normal states, where closeness to 1 means failure measurements at the EoL are similar. The prognosability of the prediction method is calculated by Equation 3.2.5, where the standard deviation of the each cycle $j \in M$ its last value $x_j(N_j)$ is divided by the mean of the absolute differences between initial and final HI values $x_j(1)$ and $x_j(N_j)$, respectively.

$$\text{Prognosability} = \exp\left(-\frac{\text{std}_j(x_j(N_j))}{\text{mean}_j\left|x_j(1)-x_j(N_j)\right|}\right), \quad j = 1,2,\ldots,M \tag{3.13}$$

3.3. Fault Diagnosis

As information on historic failure examples is often unavailable, the identification of a fault type introduces an unsupervised clustering problem. Correlation in failure behavior, reflected by similarity in LSTM-AE output, is key to identifying historically occurred failure modes. Then, diagnosis involves matching historical data characteristics with the inferred fault label for the identification of future values.

Given the the LSTM-AE reconstruction signal as input, the model for fault diagnosis should produce an output $\hat{\boldsymbol{y}} = \{y_1, ..., y_{\hat{T}}\}$, where $y_t \in \{0,1\}^m$ is used to denote the occurrence of a failure mode m at a timestep t. This section describes how fault labels are obtained through Deep Embedded Clustering and, after optional post-processing of this output, how newly input data can be classified.

3.3.1. 1D-CNN for Deep Embedded Clustering

Without labels to inform failure modes, the model should be designed to capture relevant features while simultaneously solving this fault-based clustering objective. Therefore, an encoder-like structure is defined to obtain a latent space of the input data that only contains features relevant to failure characteristics. Defining clusters based on this embedding is referred to as Deep Embedded Clustering (DEC). As concluded in the literature review, a time series-specific 1D Convolutional Neural Network (1D-CNN) is applied for DEC, capable of equally analyzing multiple SCADA variables for short or long-term failure mode-related features.

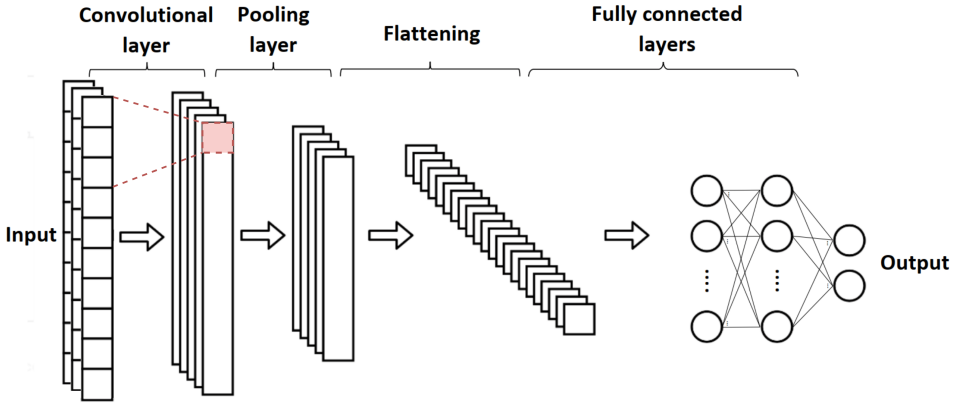


Figure 3.8: Architecture of a 1D-CNN, adapted from [224].

Figure 3.8 illustrates a 1D-CNN consisting of a single layer (whereas multi-layer CNNs have additional repetitions of convolutional and pooling layers), which applies convolutional filters along a single spatial dimension. Its process involves three elemental mathematical operations [225, 226]:

1. Before sliding over other parts of the input, a single operation of a **Convolutional Layer** is highlighted in red in Figure 3.8. This is the core operation of a CNN, where given an input sequence x, bias b, and kernel w of kernel size k, an output z at position j can be computed as:

$$z_j = \sum_{i=0}^{k-1} w_i x_{j+i} + b$$

2. To reduce dimensionality of the output, while retaining important features, a **Pooling Layer** uses a sliding window across the input, transforming the values into representative ones by taking the average or the maximum value from the input values within the window. With a sliding window of stride s, a new array p_j is computed as:

$$p_j = \max_{i \in [j, j+s]} z_i$$

3. After **Flattening**, which transforms the pooling output into a 1D vector f, the final embedding is obtained by a **Fully Connected Layer**. Similar to a standard feedforward NN, using an activation function σ, weight matrix W, and bias vector b, each element of the output vector y can be calculated as:

$$y = \sigma(W p + b)$$

3.3.2. Contrastive Learning

To encourage the formation of distinct and meaningful clusters when mapping input data to a lower-dimensional space, the model is iteratively trained using a self-supervised clustering objective based on two contrastive loss functions [186, 227–229]. This process is illustrated in Figure 3.9, where similar data samples are encouraged to be closer, while separating data samples with significant differences.

(a) Input data before contrastive learning

(b) Output data after contrastive learning

Figure 3.9: Visualization of contrastive learning of arbitrary clusters indicated by color in a 2D t-SNE space, from [230].

A "Non-Parametric loss" ($\mathcal{L}_{seq}$) is calculated between sequences, refining the global structure of the latent space. In contrast, a "Pair loss" (L_{pair}) acts at a finer granularity, ensuring small clusters form around data points that share common characteristics. Considering an optional gain g to balance out the difference in loss range at a certain training epoch e_{entry}, the coefficient is shifted from the global structure to the local optimizations during η epochs at a rate $\alpha(e) = \min(1, \frac{e-e_{\text{entry}}}{\eta})$:

$$\mathcal{L}_{\text{c}}(e) = \begin{cases} \mathcal{L}_{\text{seq}} & \text{if } e < e_{\text{entry}} \\ (1-\alpha)g \cdot \mathcal{L}_{\text{pair}} + \alpha \cdot \mathcal{L}_{\text{seq}} & \text{if } e \geq e_{\text{entry}} \end{cases}$$

Pairwise Contrastive Loss

A traditional similarity metric, introduced in [231], involves comparing pairwise distances in both the latent and input space [186, 232].

Here, the Euclidean distance $D(i,j) = ||i-j||^2$ is applied between pairs i and j as the scoring function in the embedding space. To enforce structure in the local relationships between a number of samples N, the distance between similar pairs masked by a binary value S_{ij} is minimized while the distance between negative pairs is encouraged to be larger than a given margin m:

$$\mathcal{L}_{\text{par}} = \frac{1}{N}\sum_{i,j} S_{ij} D_{ij}^2 + (1-S_{ij})\max(0, m - D_{ij})^2 \tag{3.14}$$

Instead of using a fixed margin m for all dissimilar pairs, an adaptive margin can be calculated, helping the model distinguish similar and dissimilar pairs more effectively based on the characteristics of the data, for instance, based on the softmax-based model confidence [233].

Non-Parametric Classification Loss

In contrast to explicit margin-based constraints, non-parametric approaches compare each latent sequence against all others within a batch using Softmax-normalized similarity scores [186, 234]. This allows the model to flexibly learn meaningful representations without needing categorical labels. Cosine similarity, computed as the dot product of normalized latent representations $S_{i,j} = \frac{z_i \cdot z_j}{||z_i|| ||z_j||}$, is applied between positive (similar pairs) z_i and z_j, scaled by a temperature parameter T, encouraging each sequence to be closer to its most similar counterpart:

$$\mathcal{L}_{\text{seq}} = -\frac{1}{N}\sum_{j=1}^{N} \log \frac{\exp(S_{j,i^*}/T)}{\sum_{i=1}^{N} \exp(S_{ji}/T)} \tag{3.15}$$

In the unsupervised implementation of this approach, more commonly known in image processing as Normalized Temperature-scaled Cross Entropy (NT-Xent) loss, the model defines a positive pair heuristically,

by matching a sample to its closest nearest neighbor. In the case of limited label availability, positive pairs can be defined as samples sharing the same label, allowing for increased understanding of sequence relations, guiding the formation of meaningful clusters.

3.3.3. Embedded Clustering

By calculating the backward gradients of the 1D-CNN, each epoch should reduce the contrastive losses and therefore improve fault class distinctions. Clusters are initialized with k-means clustering on the encoded features, ensuring dense, spherical clusters, suitable for refinement. Based on the updated parameters of the 1D-CNN, these cluster soft-assignments are iteratively updated at each epoch, ensuring smooth clustering behavior and robust adaptation of cluster centers to the data distribution [185, 188, 235].

For a latent representation z_i and cluster centroid μ_k, cluster updating is guided by its soft assignment q_{ik}, which can be interpreted as the probability of assigning a sample i to cluster k. Soft assignments are calculated by a Student's t-distribution, and used to update centroids as follows:

$$q_{ik} = \frac{(1 + ||z_i - \mu_k||^2)^{-1}}{\sum_j (1 + ||z_i - \mu_k||^2)^{-1}}, \qquad \mu_k = \frac{\sum_i q_{ik} z_i}{\sum_i q_{ik}}$$

Once centroids are updated after the final training epoch, each sample is assigned to the cluster with the highest probability $c_i = \underset{k}{\text{argmax}}\, q_{ik}$, where c_i is the assigned cluster label for sample z_i.

Serving as validation loss, ensuring the learned clustering assignments become more confident and better separated, a Kullback-Leibler (KL) divergence is calculated between the predicted soft assignments and their sharpened high-confidence distribution p_{ik} [185]. Sharpening refines cluster boundaries by pulling samples closer to the most likely cluster center. When the model output distribution is becoming very similar to the target distribution, and the KL-loss approaches zero, this indicates more confident and distinct cluster centers. Continued training after this point increases the risk of overfitting and cluster collapse. KL-divergence is computed as follows:

$$\mathcal{L}_{\text{KL}} = \sum_i \sum_k p_{ik} \log \frac{p_{ik}}{q_{ik}}$$

3.3.4. Cluster Post-processing

After DEC, the inferred clusters in the encoded space are derived from their initial k-means estimate. For simple data structures that show clear distinctions between classes of interest, this distance-based clustering algorithm is sufficient. However, initial SCADA-based results in this thesis have shown that this operation is prone to result in overlapping, distributed, or collapsed clusters.

The problem originates from limitations imposed by the rigid structure of k-means, which requires a given number of clusters, is limited to spherical clusters, and relies on a statistical operation, possibly leading to unstable clustering results [236]. Faced with irregular clusters, imbalanced data, or an unknown number of true clusters, clustering results benefit from re-evaluation or post-processing of the DEC outcome.

Density-based clustering does not require a number of clusters and can locate clusters with arbitrary shapes. For this reason, a frequently considered alternative to distance-based clustering is Density-Based Spatial Clustering of Applications with Noise (DBSCAN). Typically applied to increase image and data-mining performance [236, 237], sparse applications of WT SCADA clustering for power forecasting [238] and used for anomaly detection [218, 239].

DBSCAN collects a minimum number of similar points d_{scan} in a radius of size ϵ_{scan} for each sample, creating clusters around high-density regions. Low-density regions are marked as noise and can be excluded as they are considered uncorrelated to the problem. For complex structures or varying densities, a hierarchical version of DBSCAN (HDBSCAN) is applied to evaluate clusters at varying levels of ϵ_{scan}.

3.3.5. Fault classification

At this point, critical fault-related features have been extracted by data compression of either the LSTM-AE or 1D-CNN. Learning the relationship between the extracted features of historic data and their failure modes to identify future inputs necessitates a final classification step. By learning the relationship between encoded clusters and their inferred fault categories with a Fully Connected Neural Network (FCNN), unseen testing data can be encoded and classified.

The FCNN is trained to describe the relationship between fault category and input data by minimizing a Cross-Entropy loss function [174, 240]. This loss encourages confidence in correct answers, while implicitly penalizing mistakes. This loss expresses the error between the predicted labels $\hat{c}$ and learned clusters c as follows:

$$\mathcal{L}_{cl} = -\sum_{i=1}^{C} c_i \log \hat{c}_i \tag{3.16}$$

Having obtained trained clustering and classification models, a test sample x_{test} is encoded by a 1D-CNN f with learned parameters θ to obtain an encoded representation $z_{test} = f_\theta(x_{test})$. When passing this embedding through the FCNN with learned weight matrix W and bias vector b, we obtain $h_{test} = W z_{test} + b$. To introduce limited flexibility to uncertainty, a Softmax activation layer can be applied to obtain class probabilities p [241]. Taking an argmax of p delivers the class of highest probability as the final predicted class at that timestep $\hat{y}_t$.

3.3.6. Evaluation: Diagnostic accuracy & Clustering Visualization

Calculated as the percentage of correctly assigned fault labels, diagnostic accuracy is straightforward. However, evaluating the quality of contrastive learning and embedded clustering can be more complex and is best understood by visualization of data structures.

Visualization of a high-dimensional dataset requires reducing dimensionality to two or three dimensions. Unlike typical machine learning methods for regression, classification, or clustering, dimensionality reduction techniques can be used for this task, while preserving key structures.

While some traditional ML methods, such as SVM and k-means, or encoder-like DL methods mentioned in the literature study can be applied for dimensionality reduction, various specialized linear and non-linear methods have been developed to fit this purpose [242, 243]. A frequently used technique for dimensionality reduction is Principal Component Analysis (PCA). Because of its linear transformation, PCA is computationally efficient, easy to implement, and provides an intuitive and reproducible interpretation of data variance and relationships, making it preferred for global comparisons across datasets.

When facing non-linear data, PCA requires transformation, potentially altering data interpretation [244]. Therefore, t-Distributed Stochastic Neighbor Embedding (t-SNE) is a capable solution for preserving both the local and global data structure during visualization of complex clusters and patterns prevalent in the obtained embedded clusters [235].

3.4. Fault Prognosis

Efficient scheduling of maintenance action requires an estimate of remaining lifetime. By applying the anomaly detection model to the historical measurements, detected failures provide certain end-of-life labels $\hat{T} < T$. Based on these training labels, a supervised classification problem is obtained, requiring identifying the relationship between an input sample and corresponding RUL values, as visualized in Figure 3.10.

Specifically, given an input sequence $\mathbf{x}_k = \{\mathbf{x}_{k,1}, \ldots, \mathbf{x}_{k,m}\}$ of m time steps, a transformer aims to predict a corresponding sequence of RUL values $\mathbf{y}_k = \{y_{k,1}, \ldots, y_{k,m}\}$. This section outlines Transformer operation, training, and performance evaluation.

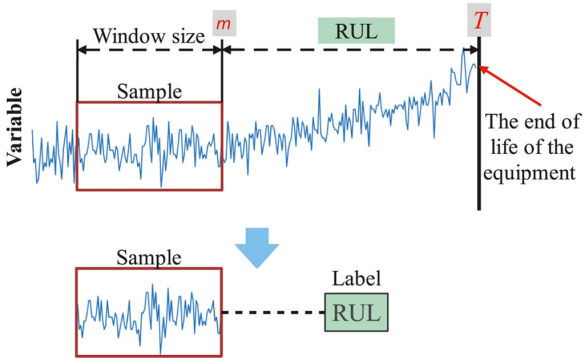


Figure 3.10: Sequence-based RUL prediction, from [245].

3.4.1. Transformer-based RUL prediction

Key long-term dependencies and inter-variable correlations are captured by the attention mechanisms (AMs) of the Transformer. Similar to AEs and some other DL applications, the Transformer follows an encoder-decoder structure, as illustrated in the left and right halves of Figure 3.11. In case of multiple layers, a Transformer consists of a stack of N identical encoder and decoder units that do not share weights.

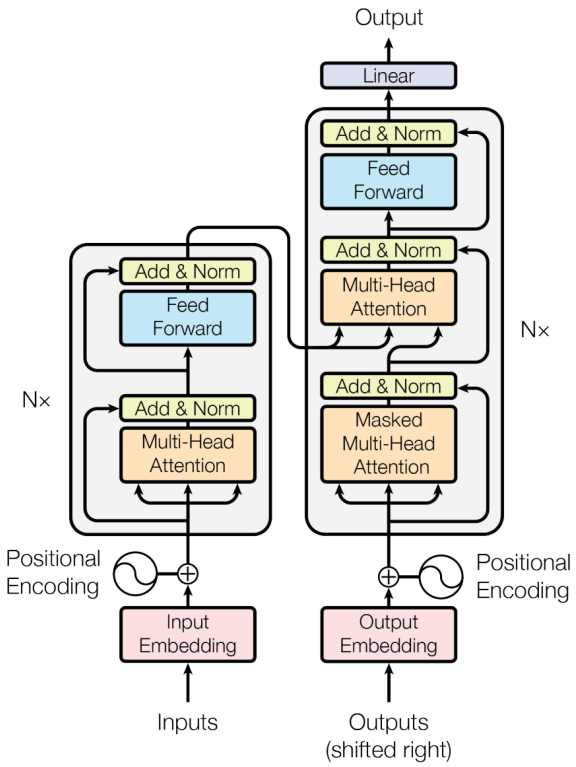


Figure 3.11: Transformer Architecture for sequence-to-sequence operation, adapted from [162].

First, a linear embedding layer maps the input data to the model dimension. Then, after positional encoding, the embedded input data passes through a self-attention layer of multiple heads, where all the other input data points are taken into context [162, 163]. The principles of positional encoding and multi-head attention are further discussed in the sections below.

Each operational sub-layer is wrapped in a residual connection. This provides an additional path for data to reach deeper layers of the model, improving the stability of backpropagation and preserving important information. These blocks then apply layer normalization to reduce statistical differences between features and stabilize learning.

Each sequence passes through a fully connected feedforward network to obtain the encoded input. This network consists of two linear transformations with a non-linear activation, projecting information into a higher-dimensional space, enabling the handling of more complex representations.

Before decoding and generating the output, an encoded version of the previously generated output is included as an additional input. This enables auto-regression, where predicted timesteps consider the previously made predictions as well. The decoder then follows the same operations as the encoder. To project the model calculation to the desired output, the output from the final feedforward layer passes through a single linear transformation.

Because the sliding window's multiple overlapping sequences produce an equal amount of overlapping outputs, its average defines the predicted RUL value for each timestep. Large deviations in the output serve as an indicator for model uncertainty, so a confidence interval can be constructed by calculating the standard deviations of these overlapping predictions at a single timestep.

3.4.2. Positional Encoding

Without recurrence or convolution mechanisms in the model, like in LSTMs or CNNs, information on the order of a time series should be manually introduced into the model to properly capture temporal context [246]. Traditionally, self-attention considers a fixed positional embedding (PE) $P = (p_1, ...p_L)$, which is added to the input embedding as $x_i = x_i + p_i$ [162, 163, 166]. Here, transformers apply a combination of sine and cosine functions at different frequencies to obtain a numeric description of the position of a sequence.

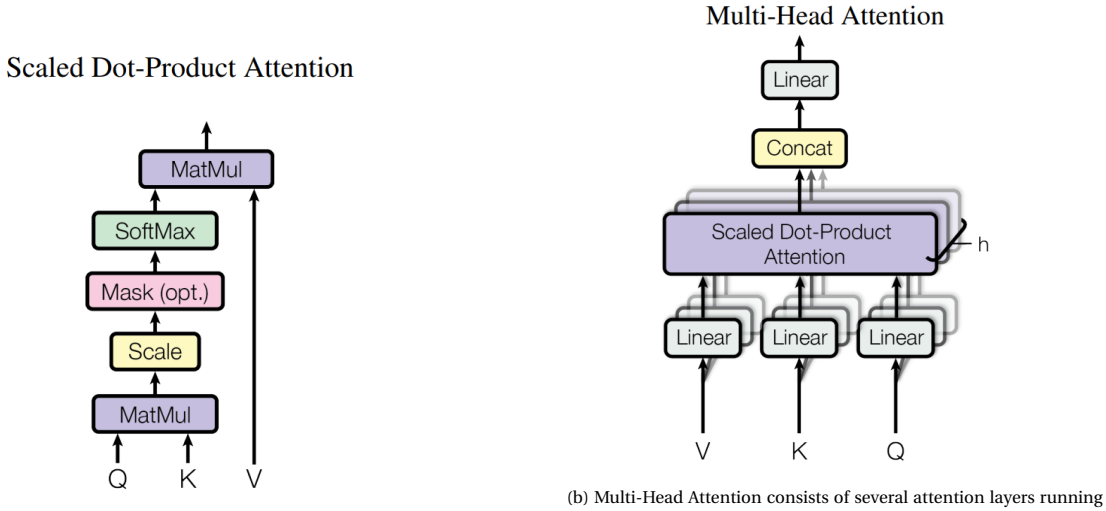
Foumani et al. and Liu et al. conclude that the existing absolute PE method is ineffective for time series data, advocating for a learnable position vector instead [246, 247], such as applied in [167, 248]. In this case, the learnable vector is initialized as a tensor of zeros of size sequence length m and model width d, tweaked during training to obtain a position vector $P \in \Re^{1\times m\times d}$.

3.4.3. Multi-Head Attention

The Multi-Head Attention modules of the transformers deploy a parallelization of multiple self-attention blocks. Self-attention computes the dependencies of different sequence elements by passing a time step's feature vector through three learnable weight matrices W_q, W_k, W_v. This returns variables query (Q) describing which past sensor readings are related to the RUL, key (K) representing past sensor readings, and (V) representing the newly added information. Then, if d_k is the key dimension, the attention scores are computed according to the scaled dot-product attention as illustrated in Figure 3.12a:

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{Q\cdot K^T}{\sqrt{d_k}}\right)\cdot V \tag{3.17}$$

Figure 3.12b describes how multiple of these self-attention blocks can be used side-by-side to achieve Multi-Head Attention. By repeating the mechanism multiple times with linear projections of Q, K, and V, each of these 'heads' learns different attention patterns, allowing the model to capture diverse dependencies in the input.



(a) Scaled Dot-Product Attention.

(b) Multi-Head Attention consists of several attention layers running in parallel.

Figure 3.12: Attention mechanisms in the transformer, from [162].

3.4.4. Transformer Training

Based on the large collection of input sensor sequences and associated RUL labels provided by the fault detection model, the Transformer should be trained to match a given input to a target RUL value. This is achieved by minimizing the loss function $\mathcal{L}_p$, which describes the difference between the model output and the true target RUL value as the Mean Squared Error (MSE), defined in Equation 3.2.2.

To enforce auto-regression during training, instead of its own predictions, the decoder is fed true RUL targets of the previous timestep. However, facing complex representations or imbalanced datasets, training with this 'teacher forcing' approach might introduce an exposure bias. To avoid becoming overdependent on the training targets, their influence should be gradually reduced during training, increasing resilience to the absence of input targets during testing [249].

3.4.5. Evaluation: RUL Prediction Metrics

To assess the performance of the Remaining Useful Life (RUL) prediction model, popular metrics are used in many other C-MAPSS RUL studies [75, 143, 250]. This involves an asymmetric scoring function as well as the Root Mean Square Error (RMSE), which provides insight into both the accuracy and the practical impact of prediction errors. The scoring based on different error values of both evaluation metrics is visualized in Figure 3.13.

1. The asymmetric scoring function is designed to penalize late predictions more heavily than early predictions, as late maintenance actions can have more severe consequences and costs. Given weights $a_1 = 13$ and $a_2 = 10$ ([74, 251]), and a difference between the estimated and true RUL values d, the total score $S = \sum_{i=1}^{n} s_i$ can be calculated as the sum of the scoring metric s:

$$s = \begin{cases} e^{-\frac{d}{13}} - 1, & \text{if } d < 0 \\ e^{\frac{d}{10}} - 1, & \text{if } d \geq 0 \end{cases} \tag{3.18}$$

2. The model is also evaluated using Root Mean Square Error (RMSE). As a commonly used metric in regression tasks, with equal weights for early and late predictions, RMSE provides an overall measure of prediction accuracy by averaging the squared differences between predicted and actual RUL values, defined as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d^2}. \tag{3.19}$$

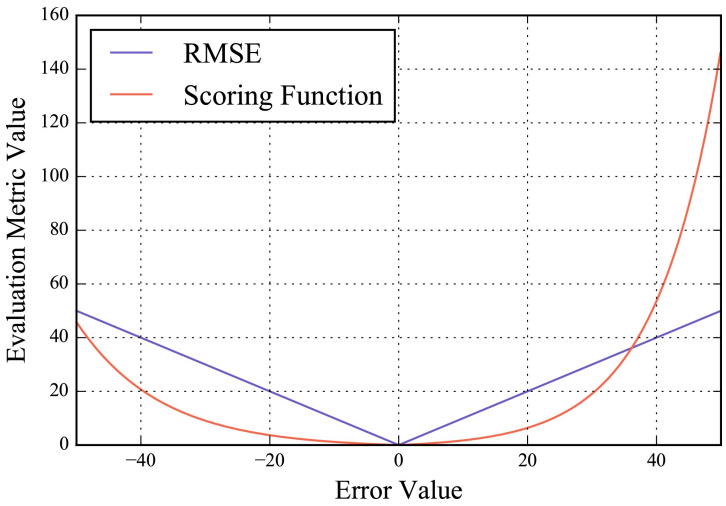


Figure 3.13: different error values and their influence based on the Scoring function and RMSE, from [143].

3.5. Conclusion

Unsupervised deep learning techniques integrate fault detection, diagnosis, and prognosis to enable comprehensive OWT health monitoring using unlabeled SCADA data. This model is implemented in the C-MAPSS and EDP case studies in the next chapters. Certain additional functions[1] were defined specifically considering limitations and challenges associated with SCADA data and will only be used there to reduce implementation complexity and improve comparability of C-MAPSS results.

3.5.1. Anomaly Detection

After preprocessing and appropriately segregating train-, validation-, and testing datasets, an AE with LSTM nodes (Figure 4.11(a)) is trained to reconstruct healthy data with reconstruction loss $\mathcal{L}_r$. A higher reconstruction error (RE) during testing indicates deviations from the learned healthy state, directly implying conditional anomalies.

[1] The adaptive threshold of the fault detection criterion (subsection 3.2.4), Cluster post-processing using (H)DBSCAN in diagnosis subsection 3.3.4, and teacher forcing decay for the Transformer subsection 3.4.4

During training, the AE is forced to select only features relevant to the health of the system. This way, the compressed latent space also provides rich health-related information. Further deciding between RE- or latent-space-based input should be determined in each case study.

After smoothing the RE signal using a moving average, a HI is derived and illustrated in Figure 4.11(b). By considering a fixed or adaptive fault threshold for this curve, an unsupervised means for fault detection is achieved, enabling the extraction of historical RUL labels for training.

3.5.2. Fault Diagnosis

To determine the most likely fault source, the failure-related LSTM-AE output features are clustered based on their degradation behavior in Figure 4.11(c). This ability is enhanced by a contrastive learning approach, where a 1D Convolutional Neural Network (1D-CNN) is trained with a contrastive loss function $\mathcal{L}_c$, to obtain an embedding where samples with similar degradation behavior are encouraged to be closer, while separating different degradation behaviors.

Clusters are assigned based on their proximity in this latent space, and are redefined by DBSCAN to account for complex cluster shapes. These clusters provide fault labels of the historic dataset used to train the Neural Network (NN) in Figure 4.11(d) with respect to classification loss function $\mathcal{L}_{cl}$. After encoding by the CNN, new and unseen data points can be properly assigned to their respective cluster, identifying the most likely failure mode.

3.5.3. Fault Prognosis

Finally, trained on the RUL labels extracted by the LSTM-AE, while using the prediction loss function $\mathcal{L}_p$, a Transformer leveraging a combination of attention mechanisms and feed-forward NNs, auto-regressively identifies the RUL values corresponding to an input sequence (Figure 4.11(e)).

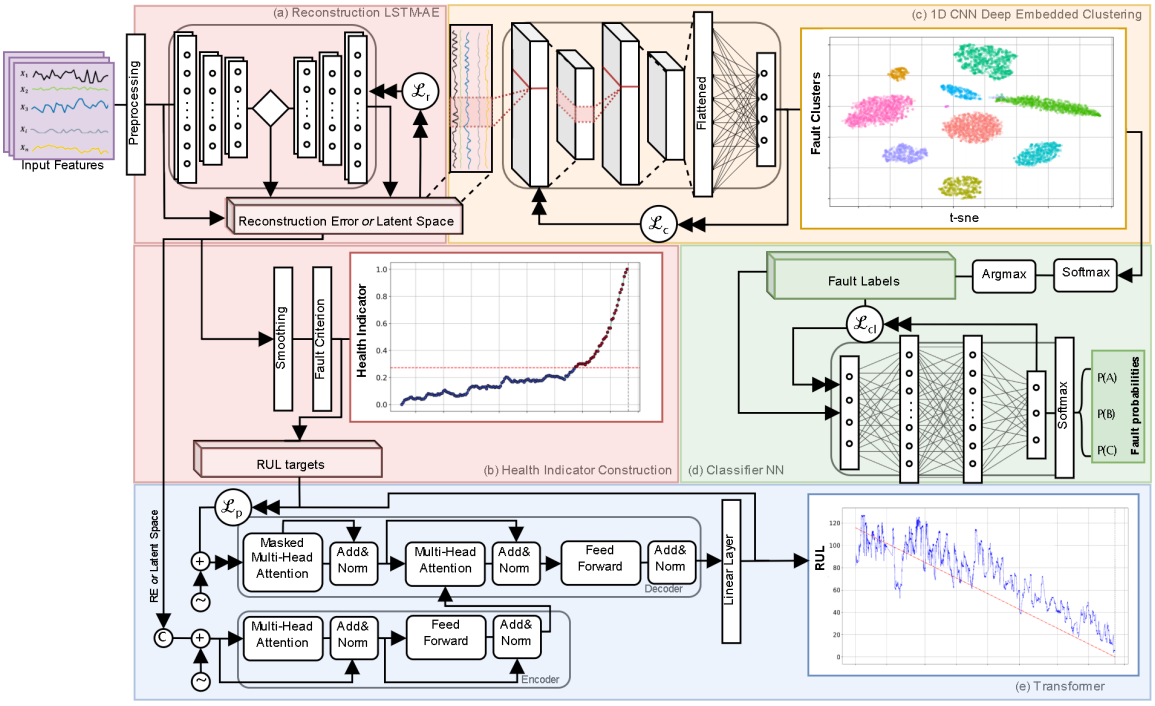


Figure 3.14: Key steps in the framework: (a) anomaly detection, (b) health indicator construction, (c) fault clustering, (d) classification, and (e) RUL estimation.

4
Simulated Engine Case Study

Model Verification

Deep learning offers an advanced approach to extracting health-related information for complex systems involving multiple failure modes and sensor signals, without labor-intensive and non-generalizable signal conditioning steps. However, numerous layers of mathematical operations that capture system behavior introduce a 'black-box' challenge: the intermittent steps between input and output are unknown, making it difficult to interpret predictions or diagnose unexpected model behavior—especially in uncertain, unlabeled environments like wind turbine SCADA data.

To address this, the current chapter applies the proposed data-driven Predictive Maintenance framework to a widely used, simulated benchmark dataset for health monitoring and RUL prediction. This controlled and more predictable environment allows for a focused investigation into the sensitivity and challenges of the proposed model. Moreover, the availability of a large collection of comparable studies allows for verification of the model by comparing results through broadly acknowledged performance metrics. Consequently, this chapter addresses the research question: *How does the performance of the proposed Predictive Maintenance framework compare to state-of-the-art methods on a benchmark dataset in terms of reliability and accuracy?*.

This chapter first introduces the synthetic NASA C-MAPSS engine dataset. After elaborating on preprocessing, the implementation of anomaly detection, diagnosis, and RUL prediction is considered in separate sections. These sections elaborate on training and hyperparameter optimization, which is performed empirically to obtain increased insight into each parameter's behavior and sensitivity. Then, results are demonstrated in the form of figures and relevant evaluation metrics, enabling the quantified comparison of results with related studies. The computational resources used for this case study are given in section B.1.

4.1. NASA C-Mapss Dataset

The simulated turbofan C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) dataset used in this chapter is made available by the NASA Ames Prognostics Center [252]. The dataset includes collections of simulated run-to-failure time series of 4 different simulations. Defined by different settings regarding operating conditions and failure modes, each configuration comprises partial and complete run-to-failure sub-datasets. The complete run-to-failure curves of the datasets that operate at a single operating condition, named 'FD001' and 'FD003', are used in this chapter. Their characteristics are given in Table 4.1.

Dataset	FD001	FD003
Number of engines	100	100
Number of measurements	20,631	24,270
Shortest Lifespan	128	145
Longest Lifespan	362	525
Failure mode	HPC single fault	HPC & Fan mixed fault

Table 4.1: NASA C-MAPSS dataset description, where measurements refer to timesteps across features.

The dataset includes 21 unspecified sensors, whose readings can be interpreted as metrics such as fan speed, pressure, temperature, and fuel flow, given per engine cycle. Additionally, the dataset contains three metrics that describe engine operational settings, such as altitude and temperature, that demonstrate limited variation for a single operating condition.

As each engine runs until failure, the true End-of-Life (EoL) matches the final measurement of each engine dataset, as shown in Figure 4.1. The failure type depends on the simulation. In the subset FD001, High-Pressure Compressor (HPC) degradation is simulated, while FD003 features an unknown combination of HPC

& Fan degradation. By combining both datasets, fault diagnosis involves identifying differences between class labels categorized as HPC single fault and HPC&Fan mixed fault.

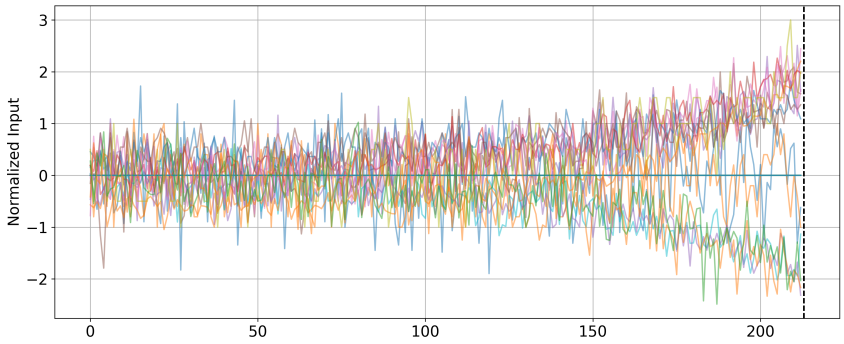


Figure 4.1: Normalized sensor readings of engine #52.

4.2. Preprocessing

The run-to-failure curves of the 200 total engines, numbered as 1-200, are separated per engine, randomized, and divided into a split of 70%, 10%, and 20% for training, validation, and testing data, respectively. The engines in these splits remain the same throughout the different model phases, ensuring that the collection of 'historic' data is consistent for every inference task. The depicted results in this chapter originate from the testing dataset, which is unknown to the model during training and can be seen as 'real-time' operation. During training, the validation dataset is introduced as unseen data to verify that additional training iterations are contributing to the improvement of the model task.

Following the steps introduced in section 3.1, input sensor readings are normalized according to sklearn's "RobustScaler" and plotted for a single engine in Figure 4.1. After normalization, each engine dataset is transformed into sequences according to the sliding-window algorithm, where the window and step sizes are specific to the application and determined per step in their respective sections below.

4.3. Anomaly Detection

To detect faulty behavior without requiring labels, an Auto-Encoder, fitted with LSTM nodes for improved time-specific reasoning, is trained to reconstruct healthy operation. An increasing reconstruction error (RE) will then serve as an indication of system deterioration.

4.3.1. LSTM-AE Hyperparameter Configuration

Healthy cycles are fed into the LSTM-AE during training, while adjusting network weights and biases to minimize training loss. This section explores the effect of various changes to important hyperparameters, including the size and stride of the sliding window, layout or configuration of the hidden nodes, and the size of the latent space. Listed in Table 4.2a, these parameters are changed from their default values, which are highlighted in bold. To enhance readability in the figures, different network configurations are assigned a letter, as shown in Table 4.2b.

Parameter	Values
Window size	18, 24, **36**, 44
Step Size	1, 2, 4, **6**, 8
Hidden Nodes	A, B, C, **D**, E
Latent Size	8, **16**, 32, 64
Training Settings	n = 25, lr = 0.01, γ = 0.94, dropout = 0.5

(a) Studied Model Parameter Values

Option	Node Layout
A	[96,64,32,16,32,64,96]
B	[128, 64, 32, 16, 32, 64, 128]
C	[128, 128, 64,16,64,128,128]
D	[256,144,64,16,64,144,256]
E	[256,256,144,16,144,256,256]

(b) Studied Hidden Node Layouts

Table 4.2: LSTM-AE hyperparameters, where n = n.o. epochs, lr = learning rate, and γ = rate of exponential lr decay.

While parameters such as the duration of training in epochs n, learning rate, learning rate decay γ, and dropout typically provide the best results at their default, they are occasionally adjusted in small increments to improve training robustness. For instance, a smaller model configuration may converge more quickly, making it beneficial to stop training early at an optimal loss. Conversely, larger models are more susceptible to overfitting, which can be mitigated by lowering the learning rate and extending the training duration.

Model hyperparameters are adjusted one at a time, while the others keep their default value. To capture stability, each configuration is tested multiple times to obtain box plots visualizing the sensitivity to LSTM-AE adjustments. These are given in Figure 4.2, displaying the effect of adjusting its value on the validation loss and health indicator performance, defined as the average of the three introduced metrics of subsection 3.2.5.

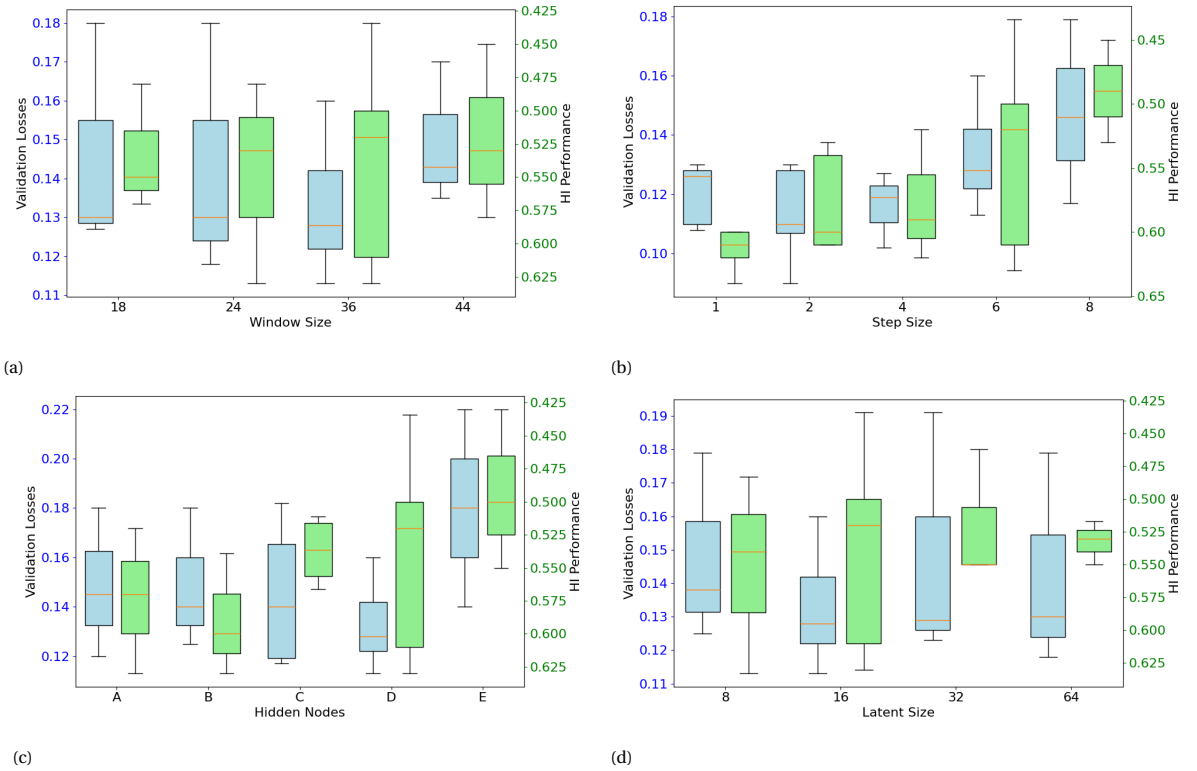


Figure 4.2: Box-plots depicting the sensitivity of validation loss and health indicator performance to differently adjusted LSTM-AE configurations.

Sliding Window

As the number of data points in each engine's life cycle is relatively low, increasing the overlap of windows to increase the density of the training data improves the sensitivity to subtle fault signs. Increased overlap requires more computational resources, possibly leading to redundant information that reduces the effective diversity of the training data. Based on the sensitivity study, Figure 4.2a suggests an optimal window size around 24 or 36, while Figure 4.2b shows that a step size of 1 achieves the best model performance. With this step size, a window size of 24 outperforms 36.

Hidden Layer Size and Distribution

Choosing the layout and size of the hidden layers should avoid overfitting issues due to an overly extensive network, resulting in memorization of the training data instead of learning data relationships. On the other hand, the size should be large enough to capture sufficient information. Figure 4.2c shows that configuration B leads to the best result, where larger configurations impact the ability to generalize, leading to declining validation loss.

Latent Size

A smaller latent size encourages dimensionality reduction, filtering out more information deemed unrelated to describing the engine operation, increasing the observability of deviations from nominal patterns. A too small latent space can also reduce the ability to reconstruct healthy data, thereby impacting the robustness of the relationship between RE and system health. Choosing a larger latent space leaves room for less specific features, but risks weakening the specificity of the reconstruction. Figure 4.2d suggests an optimal latent space size of 16 nodes.

Health indicator Construction

The construction of a reconstruction-error-based health indicator and fault detection follows the smoothing and fault detection algorithms proposed in section 3.2. Based on section B.2, a smoothing factor $\lambda = 0.05$ is applied as well as an arbitrary threshold scaling factor of 0.3, influencing the height of the detection threshold.

4.3.2. Fault Detection Results

Training should be stopped if both training and validation losses have reached a stable minimum, before the validation loss starts to increase, indicative of overfitting. Stable training behavior based on the chosen combination of hyperparameters is shown in Figure 4.3.

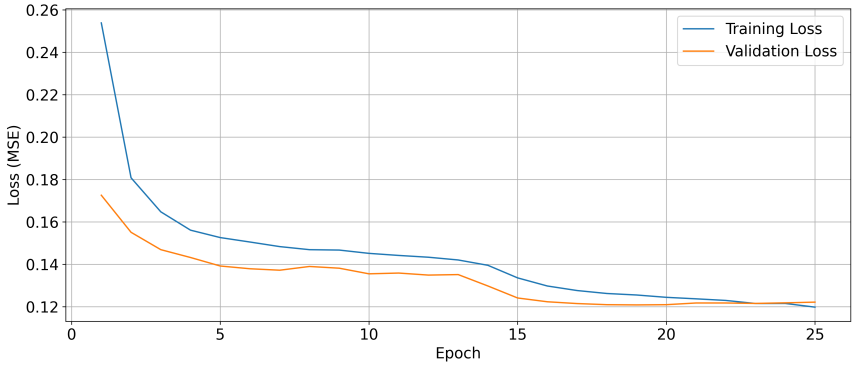
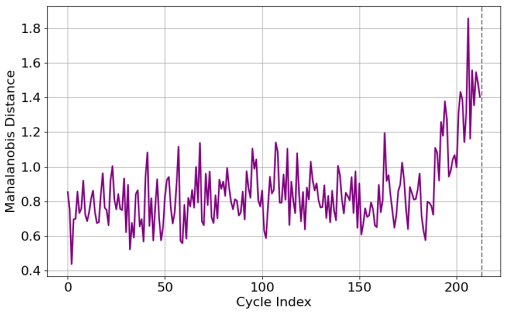


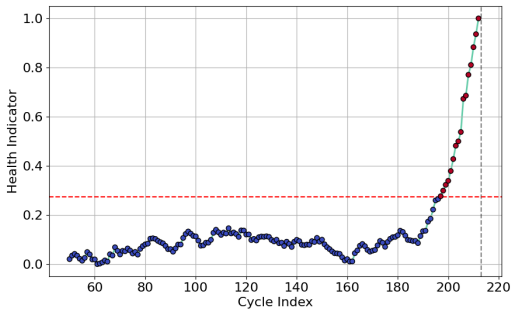
Figure 4.3: Converging training and validation losses.

Then, the model is applied to the testing data to obtain the final performance and results. The model considers each engine separately, giving a RE for every timestep. Figure 4.4a depicts the calculated Mahalanobis Distance (MD) from this LSTM-AE RE of a single testing engine, showing an increasing trend as the engine approaches the EoL.

After smoothing and scaling, Figure 4.4b demonstrates the health indicator of the same engine, where the red markers show successful detection of a defect after crossing the failure threshold.



(a) Mahalanobis distance of the RE



(b) HI-based fault detection, with healthy operation (blue) and detected failure (red)

Figure 4.4: LSTM-AE output of engine #52, until engine EoL indicated by the gray stripes.

The model performance is evaluated using the metrics proposed in subsection 3.2.5, averaged across engines, and listed in Table 4.3. Results are compared with studies that have applied the same dataset and metrics, and an experiment where the RE is calculated in the compressed latent space, as proposed by [133]. As a verification of the model, performance values are also reported for a health indicator directly based on the raw sensor data.

Results show similar performance compared to other AE-based approaches. However, improvements could be made, particularly in monotonicity, through more extensive signal preprocessing and specialized learning methods [222, 253], integration of a more advanced DL approach [131, 254], or improved conditioning of the constructed health indicator [255]. That said, excessive signal conditioning in pursuit of a smooth deterioration curve may lead to the loss of crucial information needed for downstream fault diagnosis and prediction, requiring a careful balance across the entire solution.

Source	Method	Mono.	Trend.	Prog.
This work	Raw sensors data avg.	0.60	0.60	0.40
This work	LSTM-AE	0.33	0.94	0.90
This work	LSTM-AE w/ Latent Space HI	0.37	0.78	0.75
Guo et al. (2018) [255]	CNN w/ trend burr	0.41	0.90	-
Xiao et al. (2021) [256]	LSTM w/ noisy prediction	0.12	0.85	0.72
Yan et al. (2022) [254]	PCA-LSTM-VAE	0.57	0.91	-
Yan et al. (2022) [254]	LSTM	0.26	0.88	-
Yan et al. (2022) [254]	SVM	0.18	0.89	-
González-Muñiz et al. (2022) [133]	deep AE Latent Space	0.46	-	0.99
González-Muñiz et al. (2022) [133]	VAE Latent Space	0.56	-	0.96
Koutroulis et al. (2022) [257]	LSTM-AE	-	0.67	-
Depater & Mitici (2023) [131]	LSTM-AE w/ attention	0.40	0.95	0.90
Huang et al. (2023) [222]	AE	0.25	0.80	0.82
Huang et al. (2023) [222]	LSTM-AE	0.45	0.89	0.84
Jiang et al. (2023) [253]	Hybrid stacked AE	0.88	0.91	-

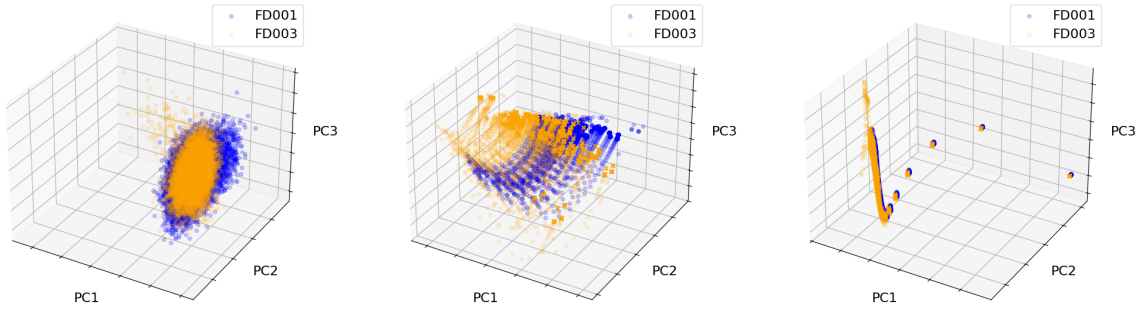
Table 4.3: Monotonocity (Mono.), Trendability (Trend.), and Prognosability (Prog.) of comparable HI studies on C-MAPSS engine FD001, sorted by year, then name.

4.4. Fault Diagnosis

Fault diagnosis involves inferring the failure category of a sample. Unfortunately, evaluating the results is complicated by the fact that engine failure labels are unknown. All engines in the FD001 dataset only demonstrate HPC error, while FD003 contains a mixture of failure modes, so the clustering task is defined as identifying the original dataset of an engine. This way, the performance of the clustering model is expressed as the number of engines of the FD001 dataset (numbered 1-100) assigned to the same cluster, as it is certain these have the same failure mechanism.

Both AE RE and latent space are rich sources of failure-related information suitable for diagnosis tasks. To select the most suitable data source, a visual comparison is made by reduction of the data dimensionality with a Principal Component Analysis (PCA) [181]. Figure 4.5 shows the MD and latent space data structures of the full dataset used in this chapter, which is a combination of NASA's FD001 and FD003 C-MAPSS datasets, in comparison with the raw input features.

Given that both datasets display different degradation mechanisms, a PCA should show differences between the datasets in the reduced spaces. Increased distance between samples in their PCA visualization generally improves the ability to analyze failure-related distinctions. The reduced PCA visualization of the raw data in Figure 4.5a shows no observable difference between datasets FD001 and FD003, while Figure 4.5b shows a more ordered structure with less overlap between datasets. The latent space in Figure 4.5c has significantly reduced the diversity of the dataset, increasing the similarity between datasets, having filtered out most failure-mode-specific features. Further extraction of failure-related information is therefore performed based on the MD of the reconstruction model.



(a) Raw Input Features (b) Mahalanobis Distance (c) Latent space

Figure 4.5: Principal component analyses of three different data sources.

4.4.1. 1D-CNN & FCNN Hyperparameter configuration

DEC parameters should be chosen to avoid under- and overfitting problems. For this clustering problem, underfitting issues result in insufficient separation of clusters. Monitoring of this issue is performed by verifying the proper grouping of FD001 engines. Overfitting overly separates the dataset, leading to cluster fragmentation.

Following subsection 3.3.2, DEC training involves the optimization of Non-Parametric loss first, for global structure organization, followed by incrementally increasing the influence of the pairwise loss, while monitoring the Kullback-Leibler (KL) divergence as validation loss. A stably reducing KL-divergence indicates proper cluster segmentation. As an additional check, the model is initialized with more clusters than the number of failure modes. If the model successfully finds the smaller true number of clusters, this indicates proper class separation.

The configurations used to obtain the experimental results demonstrating functionality of the concept for both the DEC model and the classification network are given in Table 4.4. Included are parameters that define the size and operation of the CNN, where padding, stride, and dilution of the convolutional layers are set to maintain the input length. Also included are the entry point of local contrastive loss (pair-loss) c_e, the introduction rate in epochs η, as well as margin and temperature parameters. The calculation of accuracy is unstable due to the lack of true fault labels, so the sensitivity of diagnosis hyperparameters is considered in the next chapter.

Parameter	Values
CNN Layers	2
Convolution Kernel Size	3
Pooling Kernel Size	2
Encoder Latent Size	8
Training Settings	$n = 15$, lr = 0.005, $\gamma = 0.7$
Loss function Settings	$e_{\text{entry}} = 10$, $\eta = 5$, m=1, T=0.7

(a) DEC 1d-CNN

Parameter	Values
NN Layers	1
NN Size	32
Training Settings	$n = 50$, lr = 0.001

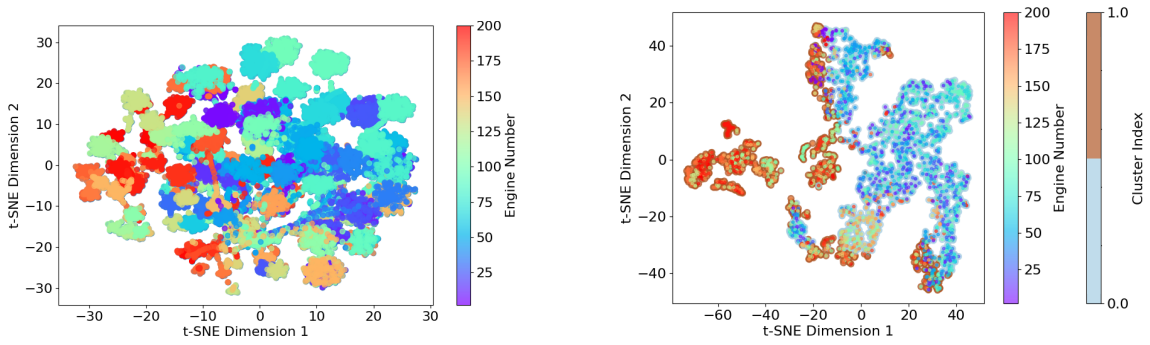
(b) Classification FCNN

Table 4.4: Fault diagnosis hyperparameters.

4.4.2. Clustering Results

The DEC model uses the training engines to learn how it can best identify and separate fault classes. The initial layout of the data is given in Figure 4.6a, which depicts a t-SNE view of the data, mapping data-points on a reduced dimensionality in such a way that minimizes the KL-divergence. Engine numbers are visualized by the color gradient, so that blue tones match FD001, and cyan to red match FD003. After DEC, Figure 4.6b shows the improved structure and assigned clusters indicated by the blue or brown outlines.

The proposed model configuration groups 90% of the engines, numbered 1-100, in the same cluster. As these are assumed to have the same failure mode, the clustering accuracy is approximated as 90%.



(a) Initial clusters

(b) Clusters after contrastive learning

Figure 4.6: t-SNE graphs before and after application of the contrastive approach. Data samples are colored according to their engine number and outlined with the assigned label.

4.4.3. Classification Results

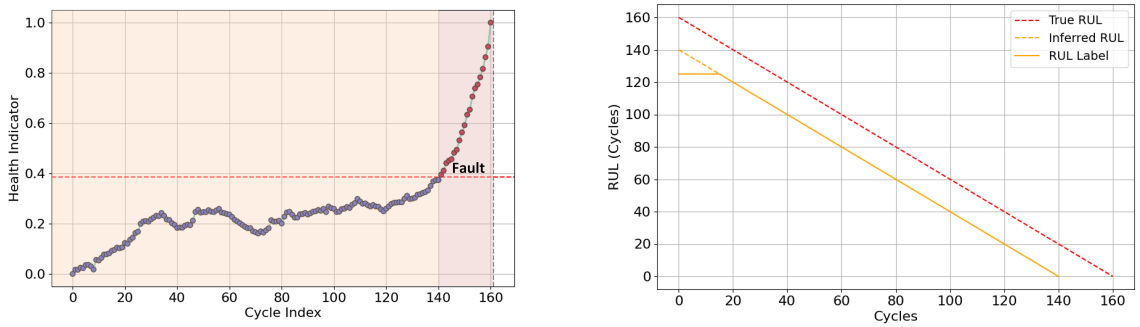
With data labels determined by the DEC model, the testing data can be directed through the model, which maps the data to its clustered embedding. Lacking ground truth labels of failure modes, the classification model is trained based on embeddings and their inferred class labels. Now, engines can be classified sequentially to find their class probability and find out how that probability changes over time. In this case study, the single-layer NN with parameters of Table 4.4b is sufficient for real-time interpretation of DEC output into a fault category that aligns with the learned clusters at least 98% of the time. For the C-MAPSS engine case, the combined accuracy of clustering and classification comes out to 88%.

4.5. Fault Prognosis

The LSTM-AE output signal quantifies the distance of input features from the normal state. By incorporating a fault detection criterion, the moment of failure can be inferred without requiring expert knowledge or labeled data. The transformer model leverages this information to predict RUL.

The number of cycles remaining until the inferred failure point establishes a linear relationship, which serves as the RUL. This process is illustrated in Figure 4.7, where the RUL labels for a test engine are derived based on a detected fault. Measurements recorded after the detected failure are excluded from the analysis, as the RUL has already reached its minimum value.

Following [75, 142, 250, 251], RUL labels undergo preprocessing to enhance training stability. First, the RUL is capped at a maximum of 125 using a piece-wise linear RUL target function. This prevents the model from overestimating RUL, while aligning with a common assumption that system degradation begins after a certain threshold [142]. Next, the RUL values are normalized to the range [0, 1] using min-max normalization (Equation 2), where $y_{min} = 0$ and $y_{max} = 125$. By ensuring that all values fall within a standardized range, the sensitivity to the scale of the data is reduced, improving numerical stability and helping the model to converge faster. After testing, predicted RUL values are rescaled to their original range.



(a) Faulty state according to the extracted health indicator

(b) Inferred RUL label compared to true EoL

Figure 4.7: Labeling process based on the constructed Health Indicator.

4.5.1. Transformer Hyperparameter Configuration

For prognosis, a transformer takes an input sequence and produces a series of output RUL values. During training, the cross-entropy loss is minimized using backpropagation, with a learning rate of 0.0005 over 120 epochs, providing the most stable learning outcomes. Sequences are generated using a sliding window of a specific size and step, where shuffling is avoided, ensuring sequential information is preserved.

The model characteristics are defined by transformer-specific hyperparameters, given in Table 4.5. Optimal model settings are determined by varying one parameter while keeping the others at their default values, highlighted in bold. Performance is measured using RMSE, with the best outcome highlighted in bold. Default values and the range of variations are based on sensitivity studies on Transformer hyperparameters for C-MAPSS RUL prediction in [163, 251, 258]. Boxplots illustrating the validation loss distribution across hyperparameter settings are shown in Figure B.3 in the Appendix.

Parameter	Values	RMSE
Window Size	**16**, 24, 32	30, **25**, 24
Dropout	0.1, 0.25, **0.4**, 0.8	30, **29**, 30, 36
Layers	1,**2**, 3	**25**, 30, 45
Model Dim.	6,10,20,**32**	28, **27**, 28, 30
Heads	1,**2**,4,8	32, 30, 25, **24**
FFNN Dim.	6, **10**, 20	34, **30**, 32

Table 4.5: Model parameter configurations, with default in bold.

Sliding Window

Because of the sequence-to-sequence nature of the model, the sliding window characteristics directly affect the resolution of the output and overall model performance. A larger window size improves the model's ability to capture long-term degradation trends but significantly increases computational cost. Additionally, larger windows reduce responsiveness to sudden changes in the engine health, considerably increasing the risk of late predictions. A smaller step size increases data overlap, improving model responsiveness and enhancing the volume of available data for training. However, this also raises computational complexity, as more overlapping sequences must be processed.

Dropout

Dropout can reduce the risk of overfitting to specific degradation patterns. By randomly masking attention weights of the self-attention mechanism, or neurons in the fully connected layers, dropout helps prevent the model from relying too heavily on specific temporal dependencies. This encourages a more robust feature extraction process, enhancing the model's ability to generalize across different degradation characteristics. However, excessive dropout may hinder learning by preventing the model from capturing critical long-term dependencies necessary for accurate failure prognosis.

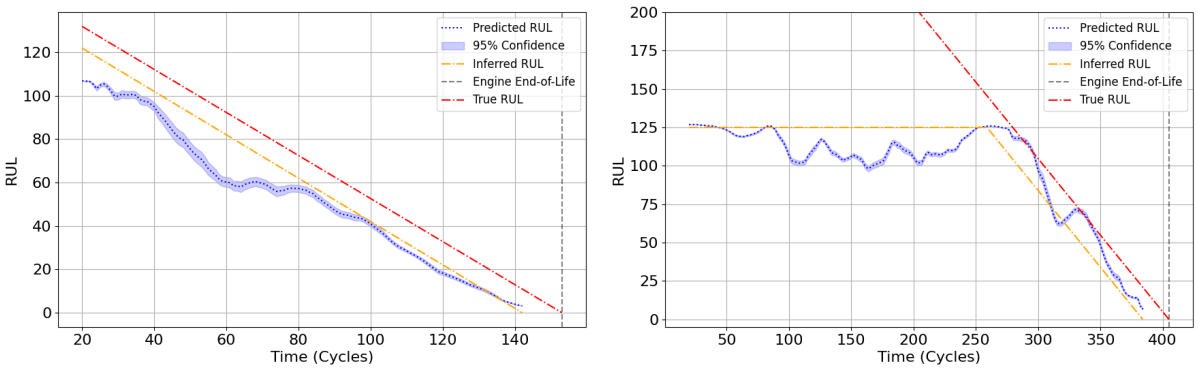
Transformer Dimensions

The depth and width of the transformer architecture significantly influence its learning capacity. Additional encoder or decoder layers allow for deeper feature extraction and improve modeling of complex dependencies. Similarly, a larger FFNN layer or size of the embedding vectors can improve generalization and enhance the model's ability to learn abstract features. However, after a certain point, additional layers or increased size may lead to issues such as overfitting or vanishing gradients, decreasing performance.

An increased number of attention heads enables the model to focus on an increased number of different aspects of the input simultaneously, potentially capturing more diverse patterns in the data. However, too many heads can dilute attention focus, as too many elements are emphasized simultaneously.

4.5.2. RUL Prediction Results

Taking the combination of the best parameters, Figure 4.8 shows the predicted RUL curves for two engines of the testing dataset. Results of eight more engines are included in Figure B.4. The two most inaccurate testing engines are provided in Figure B.5. Figures depict the predicted RUL at each cycle, along with the 95% confidence based on the standard deviation between multiple overlapping windows. Included in the figure



(a) Engine #65

(b) Engine #123

Figure 4.8: The predicted RULs at each cycle as it approaches EoL.

are also the inferred RUL based on the fault detection, together with the true RUL defined by the actual engine EoL. A complete overview of the detection and prognosis process for a single engine is given in Figure 5.17.

The RMSE and score values for the test engines are computed relative to the inferred RUL targets produced by the fault detection model, and given in Table 4.6. For comparison, the same metrics are calculated for a basic LSTM model inspired by Zheng et al. [147], serving as a baseline. Additionally, an experiment is conducted where the transformer is trained and tested directly on raw C-MAPSS input features, bypassing the LSTM-AE.

While literature generally acknowledges the merit of preprocessing data via reconstruction-based methods such as the AE proposed in this thesis, performing RUL inference on the DEC output subjected to contrastive learning methods should be considered. Therefore, an experiment is included in Table 4.6, where the embedded clusters of the 1D CNN form the input for RUL prediction with an LSTM. The LSTM is chosen over the transformer for ease of implementation, better suitability to the smaller embedded dataset, as well as the ability to compare it to [147] and [75].

The performance of these experiments is compared to similar studies on RUL prediction using the C-MAPSS dataset. If separate scores are reported for FD001 and FD003 in these studies, their average is displayed for consistency.

As expected, RMSE and score exhibit a strong correlation, as both metrics quantify deviations between predictions and targets. The RMSE and score of the LSTM method based on raw sensor inputs verify the correctness of the preprocessing and implementation technique, as results are roughly comparable to those of studies with similar methods. Transformer applications display better results overall compared to the LSTM,

Source	Method	RMSE	Score
This work	Transformer w/ LSTM-AE data	13.2 *	835.4 *
This work	LSTM w/ LSTM-AE data	17.1 *	1269 *
This work	Transformer w/ Raw sensor data	16.3	686.8
This work	LSTM w/ Raw sensor data	17.5	1641
This work	LSTM w/ DEC output data	22.9	1949
Babu et al. (2016) [142]	CNN	18.9	1396
Yu et al. (2019) [259]	BiLSTM-AE	16.1	423.5
Zheng et al. (2017) [147]	LSTM	16.2	595
Kim et al. (2020) [250]	Bayesian DNN	13.0	338
Chadha et al. (2022) [163]	STAT-Transformer	11.4	186
Liu et al. (2022) [258]	Double Attention	12.5	244
Fan et al. (2023) [167]	BiLSTM-AE Transformer	11.1	219
Ogunfowora et al. (2023) [251]	Transformer	14.9	333.8
Zhuang et al. (2023) [75]	LSTM	18.5	1226
Zhuang et al. (2023) [75]	Bayesian DNN	12.7	234.9

Table 4.6: RSMEs and scores of other C-MAPSS RUL predictive implementations, sorted by year. * = Calculated relative to inferred RUL labels.

confirming the findings of the literature study that suggest a transformer-based approach allows for a strong interpretation of time-based features.

Notably, prediction accuracy tends to improve when incorporating the LSTM-AE RE. However, the reconstruction-based RUL predictions demonstrate higher score-based errors, which suggests the overall prediction might be more accurate, but at the same time more unstable, as shorter deviations of larger magnitude are penalized exponentially by the scoring function.

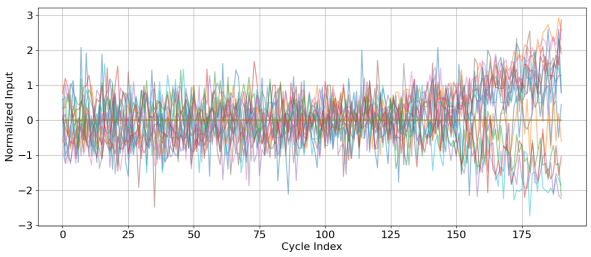
It is important to recognize that the performance metrics for methods that have been trained using the LSTM-AE are computed with respect to the inferred RUL (orange line), rather than the ground truth failure time. Since the ground truth failure time is unknown to the model during training and testing, evaluating a supervised model against an unobservable target would artificially inflate the error. This discrepancy arises because failure detection occurs before the true EoL, depending on the sensitivity of the fault detection criterion. Adjusting the failure threshold can fine-tune this detection timing, impacting both prediction quality and error metrics.

4.6. Concluding Remarks

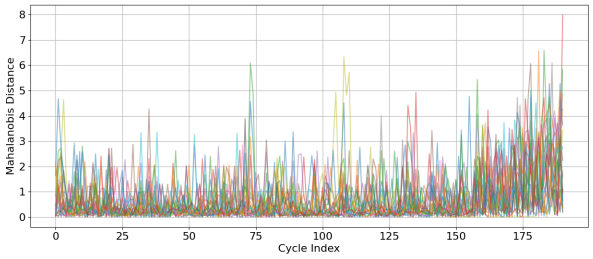
This chapter has demonstrated the feasibility of the proposed unsupervised methodology for fault detection, diagnosis, and RUL prediction using the C-MAPSS dataset. The results demonstrate that the approach effectively identifies anomalies through LSTM-AE-based reconstruction, where the derived HI showed a clear correlation with degradation trends, confirming its validity as a failure-sensitive metric.

The model distinguishes fault patterns using deep embedded clustering, indicating distinct separability between different failure modes, suggesting that the learned latent space captures meaningful fault characteristics, which were easily learned by the classification layer. Solid evaluation of the diagnosis results has proven to be difficult, however, and should receive additional attention in the following chapter.

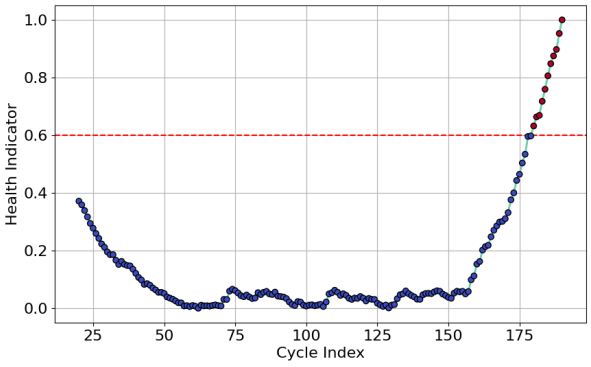
RUL is successfully predicted with a Transformer-based model, where results exhibited a consistent trend with expected degradation, though variations in accuracy were observed across different engines. Comparison of results with similar studies on the same dataset has demonstrated that the approach provides a competitive means of data-based health monitoring, allowing further implementation on Wind Turbines.



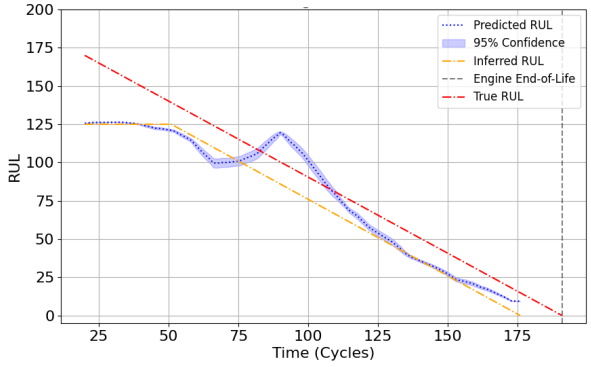
(a) Normalized Input Features



(b) Reconstruction Mahalanobis Distances per sensor of the model



(c) A detected fault based on the Health Indicator



(d) Estimated RULs at every cycle

Figure 4.9: The steps taken to achieve fault detection and an RUL estimate of engine #32.

4.6.1. Discussion & Future Work

The model design proposed in this thesis is primarily based on a review of wind turbine methods, chosen for suitability while considering the scope and workload constraints of this research. Additional health monitoring studies were introduced in this chapter to compare performance, highlighting potential areas for improvement.

Anomaly Detection

Figure 4.9b demonstrates the reconstruction model's susceptibility to early-life anomalies in sensor measurements, which appear in the initial cycles of the raw data. Due to the way the HI is constructed, these early deviations have long-lasting effects, see Figure 4.9c. This reduces fault detection responsiveness, negatively impacting RUL inference—especially for engines with shorter lifetimes. Moreover, noise and inaccuracies in the reconstruction model affect RUL prediction. A comparison between the timing of disturbances in Figure 4.9b and the increased RUL prediction errors in Figure 4.9d suggests a possible correlation between these uncertainties.

So, as the reconstruction model used for HI construction forms the basis for any following inferences, improving its stability and reasoning capacity is expected to yield the most significant overall improvements. Differences in performance compared to related studies can inform areas of improvement: De Pater & Mitici [131] enhance an LSTM autoencoder by incorporating Luong attention and explicitly modeling operating conditions. These modifications improve the model's ability to correctly interpret data structures, leading to more effective learning. Additionally, their study applies a more informed sensor selection process, further enhancing inference effectiveness. Huang et al. [222] improve performance through extensive sensor selection and t-SNE clustering as a preprocessing step. Similarly, Yan et al. [254] apply PCA as a preprocessing step for dimensionality reduction. Jiang et al. [253] employ advanced layer-by-layer training to selectively capture essential state features while suppressing irrelevant variations. By intelligently merging the extracted features, their method produces highly stable outputs.

Diagnosis

As the current approach to diagnosis verification relies on several assumptions, alternative applications of time-series clustering for health monitoring can be explored for this dataset. The proposed DEC method could be extended to datasets with multiple operating conditions, allowing for an evaluation of the 1D CNN's contrastive-learning capability. This aligns with prior studies that have applied clustering techniques to the C-MAPSS dataset [222, 260].

At present, basing RUL prediction on DEC output data yields unstable results, as the 1D CNN significantly reduces the time-series dimensions, which contributes to this instability. Redesigning the CNN with a stronger focus on RUL prediction could provide a means to improve prediction performance, provided that it does not compromise the model's diagnostic capabilities.

Prognosis

While an improved reconstruction will contribute to enhancing the stability of the prognosis, further improvements can be identified between differences in performance as shown in Table 4.6.

Similar to this work, Chadha et al. [163] apply a transformer consisting of both encoder and decoder. Their transformer-based RUL prediction model introduces specialized attention blocks, however, that focus on detecting local degradation patterns. By splitting input features into multiple smaller heads with shared weights, the model significantly reduces the number of trainable parameters. This leads to notable performance improvements, suggesting that structural adaptations that simplify learning complexity can enhance the effectiveness of transformer-based RUL prediction.

This idea is reinforced by reviewing transformer RUL approaches by Ogunfowora et al. [251] based on raw data, and Fan et al. [167] adopting an additional denoising LSTM-AE for preprocessing. Both studies employ an encoder-only transformer structure that outputs a single RUL value per input sequence. This simplified model structure reduces computational complexity, streamlines training, and possibly produces improved results.

As a baseline method, the LSTM experiment of this work should perform similarly to Zheng et al. [147] and Zhuang et al. [75], who applied LSTM networks for RUL prediction using a similar methodology. This difference is explained by dataset usage. Zheng et al. tested on a separate "FD001_test" dataset, where their model estimated RUL using partial ground truth, providing an improved prediction starting point and a reduced prediction horizon. This also frees up the entire "train" dataset as additional training data.

5
Application to Offshore Wind Turbine SCADA data

Results

Implementation of a SCADA-based Predictive Maintenance (PdM) approach provides OWT operators with a tool for maintenance decision-making. SCADA data challenges complicate direct derivation of health-related features, leading to the proposed deep learning framework in chapter 3.

After verifying framework feasibility and studying its sensitivity and limitations on the simulated engine dataset in chapter 4, the model is now is now applied to real-world OWT SCADA data. This transition introduces new challenges and practical considerations. Accordingly, this chapter addresses the research question: *How does the proposed framework perform when tested on offshore wind turbine SCADA data?*.

The chapter begins with an introduction of the dataset, followed by the definition of test cases designed to maximize the utility of available data. The approach to hyperparameter optimization is then presented, with a complete overview of all investigated hyperparameters, their descriptions, and ranges provided in section C.3.

After SCADA-specific preprocessing steps prepare the dataset, the implementation of each model function is described in dedicated sections, covering training, hyperparameter sensitivity, implementation details, results, and discussion. The chapter concludes with a summary and reflection on the key findings.

5.1. EDP Offshore Wind Turbine Case Study

After comparing multiple open-source datasets in section C.1, Portuguese EDP data is selected for its extensive range of SCADA parameters and detailed descriptions of performed maintenance actions [261]. It includes status and error codes, anemometer and turbine SCADA sensor signals from four 2 MW turbines, spanning 2016 and 2017. Maintenance actions given in the dataset are illustrated in Figure 5.1 and detailed in section C.2.

Anemometer readings primarily reflect farm-level environmental conditions, while status and error codes are often inconsistent and incomplete, creating a complex expert systems problem [64, 262]. Therefore, the 81 SCADA sensor signals that provide continuous operational data directly related to turbine performance and condition are used.

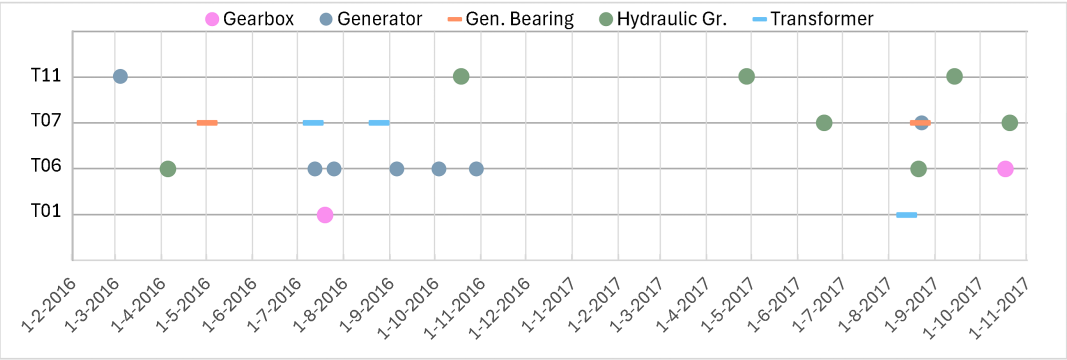


Figure 5.1: Overview of fault occurrences per turbine, according to the failure logbook provided in Table C.2 & Table C.3.

5.1.1. Data splits & Test Cases

To maximize the use of available data while enabling comprehensive testing across all turbines, four different experimental configurations are defined, specifying how the turbines are distributed into training and validation cases.

For testing, the unseen dataset is split into segments corresponding to individual failure events marked by timestamps in the maintenance logs. Logged maintenance actions are assumed as failure occurrences, marking the targets of model inference.

Overlapping or closely spaced failures are grouped into a single testing segment. This applies to three specific scenarios: (1) For the five generator maintenance actions in 2016 on T06, the first and last involve component replacement. Segments are defined leading up to the first replacement, and from the first to the last failure in the group. (2) In T07, the transformer faults from July and August 2016 are grouped together as the first is a temperature alarm, and the second involves replacement. (3) Also in T07, overlapping alarms in August 2017, indicating both generator and generator bearing damage, are treated as a single failure event.

After these adjustments, Table 5.1 details how the data is distributed in each case, specifying the number of failure modes and segments present in the testing dataset.

	Case 1	Case 2	Case 3	Case 4
Training	T01, T06	T06, T07	T07, T11	T01, T11
Validation	T07	T11	T01	T06
Testing	T11	T01	T06	T07
Failure modes	2	2	3	4
Test segments	4	2	5	5

Table 5.1: Test cases for the implementation of the EDP SCADA dataset. The number of faults and fault types is given for the testing subset.

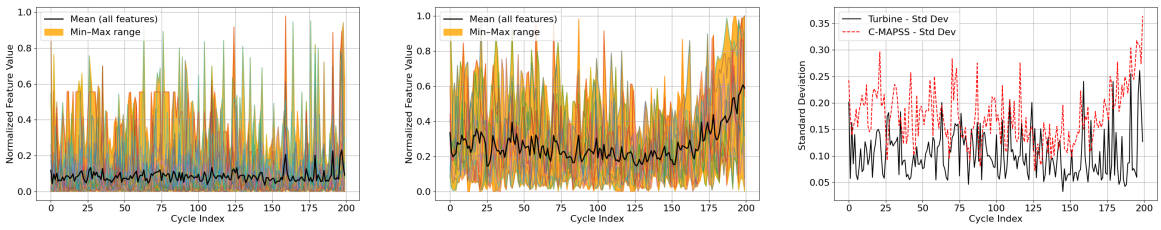
5.1.2. Comparability of C-MAPSS and EDP Datasets

To justify the generalizability of the C-MAPSS findings and highlight the additional challenges posed by SCADA data, similarities and differences between the datasets are analyzed.

Lifespans in the EDP dataset typically range between 9000-30000 measurements (9-30 weeks at 10-minute SCADA intervals), compared to 30–200 cycles for C-MAPSS engines. After resampling to comparable lengths and normalizing input features, the mean and standard deviation of randomly selected engine cycles and wind turbine segments are compared.

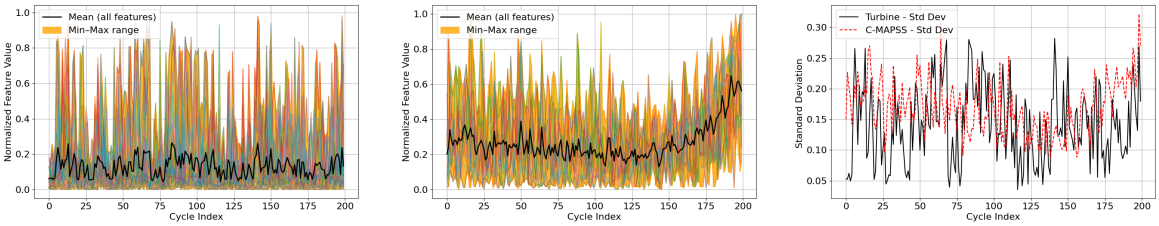
Figure 5.2 and Figure 5.3 display individual feature trends, their mean, and variability across features. Both datasets exhibit substantial short-term fluctuations, reflecting sensor noise and operational variability. In Figure 5.2, the mean trends show an upward drift towards End-of-Life (EoL), particularly subtle in turbines, while the standard deviation increases more clearly with cycle progression towards EoL. The OWT data in Figure 5.3 shows no clear indicators for approaching EoL in the mean and standard deviation curves.

Comparison of both datasets suggests fault features are much more hidden in SCADA data, requiring more advanced model configuration, posing an increasingly difficult hyperparameter optimization problem.



(a) OWT Features and mean (b) C-MAPSS Features and mean (c) Standard deviations

Figure 5.2: Comparison of segment 1 of T01 and engine 60.



(a) OWT Features and mean

(b) C-MAPSS Features and mean

(c) Standard deviations

Figure 5.3: Comparison of segment 1 of T11 and engine 53.

5.1.3. Automated Hyperparameter Optimization

In the previous chapter, the relatively small size of the C-MAPSS dataset and the existence of a search space constrained by the consultation of similar studies allowed for manual hyperparameter optimization based on small adjustments. Facing new conditions and data specifications, this chapter requires a new evaluation of the ideal hyperparameter configuration.

Due to longer computing times and an increasingly complex hyperparameter optimization problem, this task is performed by the Optuna Python package [263]. Optuna is designed to automate the search for optimal model parameters, more efficiently and systematically compared to manual grid or random searches. Leveraging Bayesian Optimization and Tree-structured Parzen Estimators (TPE), optimization involves repeated retraining of the model during a certain number of trials, while aiming for a specified objective, like minimizing validation loss.

For every optimization, a pruning algorithm is included that identifies unpromising trials at the early stages of training and applies early-stopping. Pruner sensitivity is controlled by a number of warmup trials and a number of warmup steps (epochs).

Due to the computational demands of repeated retraining, hyperparameter optimization is conducted exclusively on Case 1. As this case offers a diverse range of fault types in its training and validation data, tuning hyperparameters on this case is expected to yield results that generalize well to other cases. This assumption is tested in section C.7.

5.1.4. Optimizer Configuration

Hyperparameter searches are performed per sub-function of the model, each with its own search space, elaborated upon in the sections below. To achieve the best balance between computing time and optimal solution, the optimization algorithm involves running the optuna optimizer for 100 trials, with 10 warm-up trials and 20 warm-up steps.

The increased data size requires more extensive computational resources, especially when performing repeated model retraining. Therefore, hyperparameter optimization is performed on the DelftBlue supercomputer, provided by the Delft High Performance Computing Centre [264].

DelftBlue consists of multiple compute nodes, including GPU nodes, which are preferable for machine learning tasks. The author's education account provides access to up to 2 GPUs (NVIDIA Tesla V100S 32GB) and 64 CPU (AMD EPYC 7402 24C 2.80 GHz) cores, across a maximum of 2 nodes. These resources are allocated through the SLURM job scheduler, enabling efficient parallel processing of deep learning workloads. The process of setting up DelftBlue for this thesis is outlined in section C.4. To reduce the queue time required before starting a job due to resource allocation on the supercomputer, while allocating sufficient resources, 4 CPU cores of 4GB each, with a single GPU are used in this chapter.

5.2. Preprocessing

Effective preprocessing of the input dataset, including the selection of relevant signals and appropriate signal conditioning, is essential to ensure robust model performance and stable training. Then, datasets are converted into sequences using the sliding window approach introduced in subsection 3.1.2. To properly capture temporal dependencies, the chronological order is preserved when loading the sequences as recommended by [265].

5.2.1. Sensor Selection

Different sensor signals contribute at varying levels of relevance to specific degradation processes in multi-sensor prognostics. Irrelevant sensors may add noise, reducing prediction accuracy and increasing computational burden [134, 266, 267].

Identification of these redundant features is achieved through the application of a correlation analysis, revealing variables with similar temporal effects [206]. Parameters that show high correlation in each turbine dataset are dropped from the dataframe. In this study, this criterion is set as an absolute correlation larger than 95%, removing variables such as minima, maxima, and averages of the same measurements, or comparable temperature measurements. Figure 5.2 lists the remaining 31 SCADA parameters selected for implementation, and Figure 5.4 displays a correlation heat map of these final parameters.

No.	Parameter Description	No.	Parameter Description
0	Gen_RPM_Max	16	Gen_SlipRing_Temp_Avg
1	Gen_RPM_Min	17	Blds_PitchAngle_Avg
2	Gen_RPM_Std	18	Cont_VCP_ChokcoilTemp_Avg
3	Gen_Bear_Temp_Avg	19	Grd_Prod_CosPhi_Avg
4	Hyd_Oil_Temp_Avg	20	Grd_Prod_VoltPhse1_Avg
5	Gear_Oil_Temp_Avg	21	Grd_Prod_VoltPhse2_Avg
6	Nac_Temp_Avg	22	Grd_Prod_VoltPhse3_Avg
7	Amb_WindSpeed_Max	23	Grd_Prod_Pwr_Min
8	Amb_WindSpeed_Avg	24	Grd_Prod_Pwr_Std
9	Amb_WindDir_Relative_Avg	25	Grd_Prod_ReactPwr_Max
10	Amb_Temp_Avg	26	Grd_Prod_ReactPwr_Min
11	HVTrafo_Phase1_Temp_Avg	27	Grd_Prod_ReactPwr_Std
12	HVTrafo_Phase3_Temp_Avg	28	Grd_Prod_PsblePwr_Std
13	Grd_InverterPhase1_Temp_Avg	29	Gen_Bear2_Temp_Avg
14	Cont_Top_Temp_Avg	30	Nac_Direction_Avg
15	Cont_Hub_Temp_Avg		

Table 5.2: The remaining 31 SCADA parameters

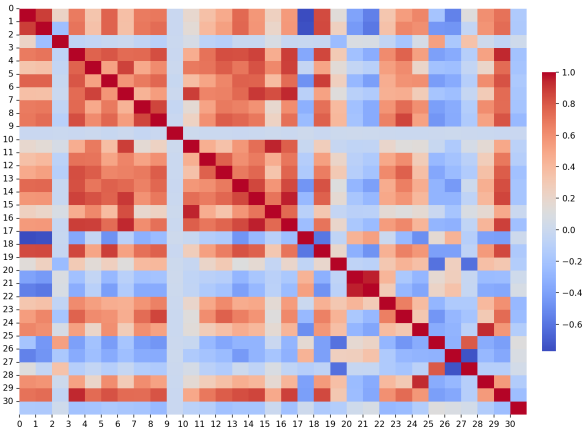


Figure 5.4: Correlation heat map of the parameters

5.2.2. Signal Conditioning

The properties of the EDP dataset bring up some additional data handling steps required for implementation. Missing values should be imputed for the continuity of the time series. Because the missing values in the dataset appear individually or in a small number of consecutive occurrences, linear interpolation is applied to fill in these gaps. An overview of missing value occurrences is given in section C.5.

Initial testing revealed inconsistent model performance across turbines for test segments spanning both years of operation. To mitigate misalignment between years, the first failure segments of 2017 are truncated to begin on January 1st.

To ensure inputs of different ranges are considered equally, "Robust-Scaler", introduced in subsection 3.1.1, reshapes the data structure while preserving essential data characteristics, with minimal influence of outliers. Fitting of the scaler must be performed on either training or validation data to avoid data leakage, involving improper sharing of unseen future data. The final time-series and power curves of the selected parameters are given in section C.6.

5.3. Anomaly Detection

Essential patterns and trends in the time series data collected from OWT SCADA systems are captured through compressing and decompressing the input by an LSTM-AE (Long Short-Term Memory Auto-Encoder). This section covers the selection and sensitivity of LSTM-AE hyperparameters, the construction of the Health Indicator (HI) based on the reconstruction error output, and the application of an adaptive threshold for obtaining fault detection.

5.3.1. LSTM-AE Hyperparameter Configuration

Repeated training iterations determine a configuration that offers the lowest validation loss. To guide this process, parameters are given a suggested search space, bounded by a higher- and lower bound, and, if applicable, a step size to consider when making adjustments given in Table 5.3. Parameter ranges are inspired by the experimental findings of the previous chapter, and supported by [128, 134, 268, 269].

Parameter	Min	Max	Step
Window Size	16	128	16
Step Size	2	24	2
Hidden Size 1	32	256	32
Hidden Size 2	32	256	32
Hidden Size 3	16	126	16
Latent Size	8	64	8
Number of Layers	1	5	1
Dropout	0.1	0.5	0.1
Learning Rate	0.001	0.05	-
Number of Epochs	20	160	20
Gamma (lr decay)	0.8	0.99	-
Gamma Stop	1	101	20

Table 5.3: Tunable LSTM-AE Hyperparameters with the suggested ranges for the optimization algorithm, indicated by upper- and lower bounds, as well as the step size.

By plotting the validation loss of each trial against its corresponding hyperparameter configuration, the boxplots in Figure 5.5 provide insight into both the potential performance of different parameter settings and the stability of their outcomes.

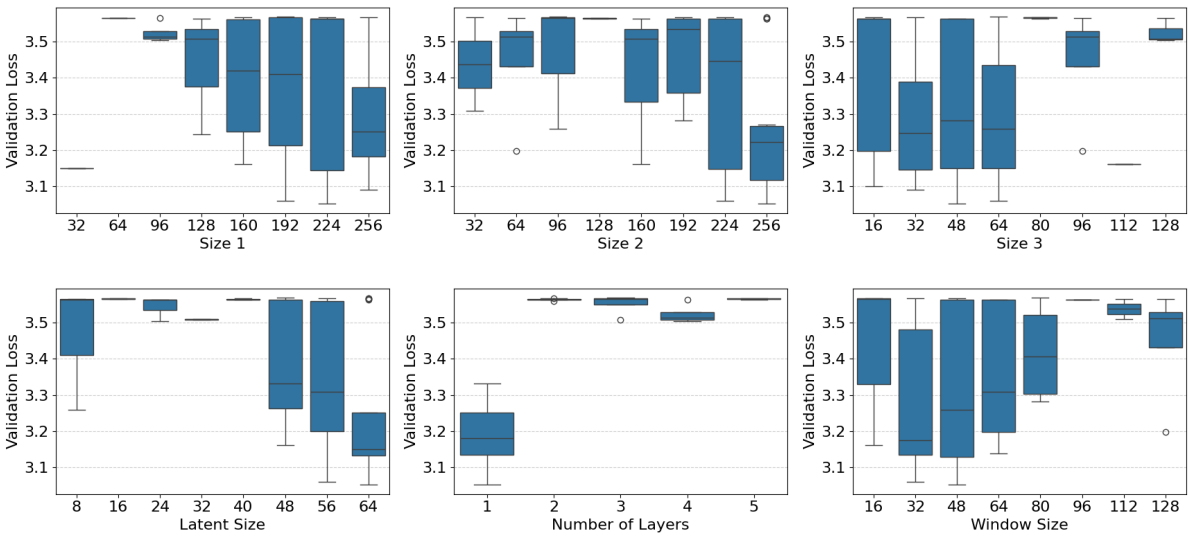


Figure 5.5: Box Plots for LSTM-AE sensitivity after 100 Optuna parameter optimization trials, showing the Validation Loss (MSE) per hyperparameter.

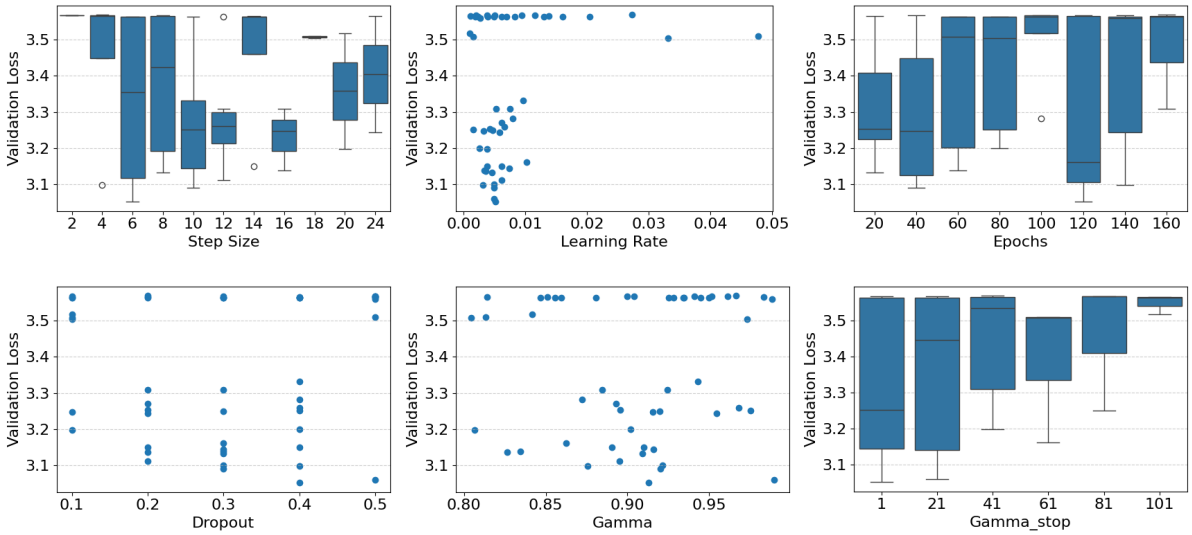


Figure 5.5: Box Plots for LSTM-AE sensitivity after 100 Optuna parameter optimization trials, showing the Validation Loss (MSE) per hyperparameter (continued).

The results indicate that larger autoencoder node sizes yield better performance by reducing compression severity and preserving more information. Similarly, higher latent sizes improve stability by retaining more of the original feature space. Based on these findings, the selected node sizes are 224, 224, 64, and 48 for sizes 1, 2, 3, and the latent size, respectively.

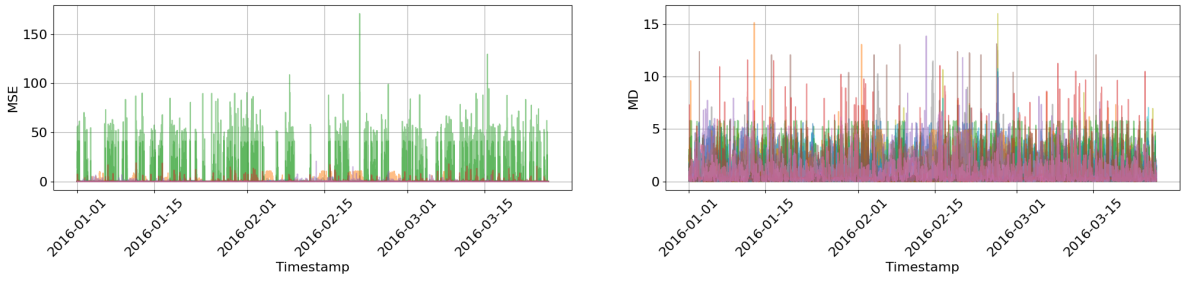
Optimal validation loss is achieved with a single-layer architecture, avoiding overfitting issues and overall intractability of the model. Dropout is ignored as a consequence, as for recurrent networks, dropout must be applied to connections between layers instead of within the same layer, because this risks disrupting temporal dependencies.

For the sliding window, the best results are obtained with a relatively small window size of 48. A step size of 12 provides increased stability and reduces computational time compared to smaller step sizes.

Finding ideal training-related parameters through optimization can be challenging due to strong interdependencies between parameters. Nevertheless, effective results are achieved by training for 40 epochs with a learning rate of 0.005, while disabling learning rate decay.

5.3.2. Health Indicator Configuration

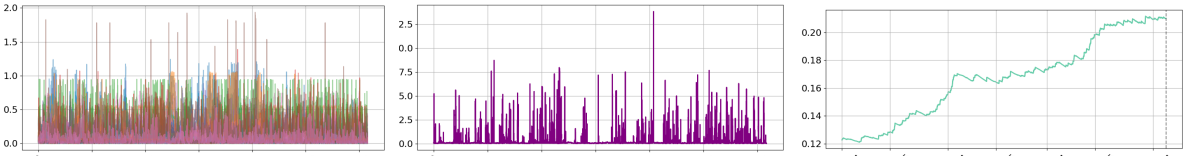
The HI is derived by post-processing of the LSTM-AE output, following the steps introduced in subsection 3.2.3. For SCADA, the most reliably performing HI was found by combining two expressions of the LSTM-AE reconstruction error: Mean Squared Error (MSE) and Mahalanobis Distance (MD), both given in Figure 5.6. As both signals are sensitive to noise in a slightly different way, their average improves the stability of the HI signal.



(a) MSE (b) MD

Figure 5.6: LSTM-AE Reconstruction Error per input feature expressed as Mean Squared Error (MSE) and Mahalanobis Distance (MD).

After applying scaling and normalization, the HI is obtained by smoothing, using $\lambda = 0.0001$. At a certain moment in time, the adaptive failure threshold is based on the mean and standard deviation of a window of historical data. Optimal fault detection behavior is achieved by properly tuning the window size w and responsiveness to the standard deviation k. For this case study, $w = 3000$ and $k = 3.5$ are selected based on tests discussed in section C.8. These measures produce the final HI, as visualized in Figure 5.7.



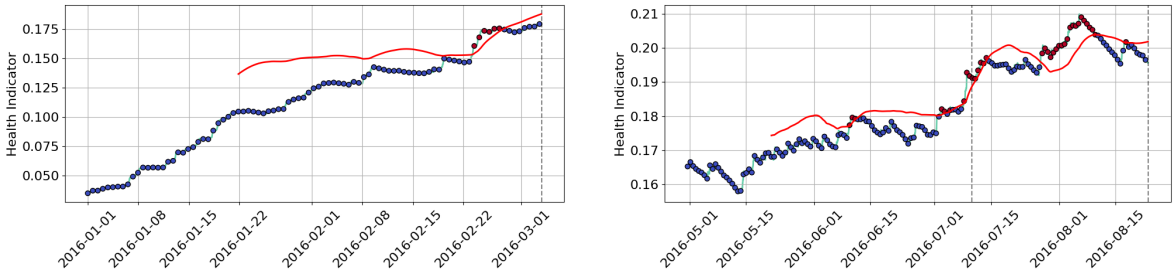
(a) Health signal after Scaling and Normalization (b) Fault Signal Averaged across Features (c) HI after EWMA smoothing

Figure 5.7: Three Steps for obtaining the HI with a health signal defined by the average of the LSTM-AE MSE and MD signals.

5.3.3. Fault Detection Results

After proper configuration of the model, two health indicators and their detected faults are shown in Figure 5.8. All other HI plots with detected faults are given for each case in section C.9 for additional review.

Figure 5.8a demonstrates how a fault is detected using the adaptive threshold. As the EoL approaches, system deterioration drives up the health indicator faster than the threshold is configured to allow, causing a generated alarm. Figure 5.8b shows how varying slopes in the health indicator influence the threshold, causing the generation of alarms at multiple points in the segment. These alarms match two gray dashed lines that indicate two faults contained in this segment: high transformer temperature in July and August, leading to the repair of the Transformer refrigeration.



(a) Case 1, segment 1: T11 Generator Fault (b) Case 4, segment 2: T07 Transformer Fault

Figure 5.8: Health-Indicator-based fault detection. The red markers are generated alarms, the red line indicates the adaptive fault threshold, and the gray dashed line marks segment EoL.

For each case, the HI performance metrics introduced in subsection 3.2.5 are calculated and given in Table 3, along with the share of detected faults. Anomalous behavior is detected thirteen out of sixteen times. Temperature errors and larger replacements score best, while hydraulic group errors, such as oil leakage and brake circuit errors, are sometimes missed if their influence on turbine operation is too subtle for capture.

Fault detection is also implemented based on the full collection of raw input features in section C.10, showing reduced HI performance and fault detection robustness, verifying the sensor selection and preprocessing approach.

Test Case	Mono.	Trend.	Prog.	Detected Faults
Case 1	0.63	0.36	0.87	2/4
Case 2	0.77	0.86	0.94	2/2
Case 3	0.65	0.88	0.84	4/5
Case 4	0.66	0.83	0.82	5/5

Table 5.4: Monotonocity (Mono.), Trendability (Trend.), and Prognosability (Prog.) of the EDP test cases, as well as their share of detected faults.

5.3.4. Discussion

The performance of the HI generation compares well to various C-MAPSS studies in the previous chapter. However, variability in health indicator trajectories causes significant inconsistencies between generated alarms and the actual moment of failure. Improvements can be considered for the AE as well as the fault detection system.

The first generated alarm of some segments is significantly earlier than the actual failure, ranging from a week to 2 months. Improved processing and smoothing of the HI, combined with improving the fault threshold, could be considered to reduce this distance. Additionally, incorporating an extra computational step to interpret alarm severity could enhance alarm informativeness.

Fault detection responsiveness benefits from increasingly reliable health indicators, obtained by improving the reconstruction model. More advanced data interpretation by the AE, increasing its reasoning and resilience to noise, increases HI performance metrics and enhances fault detection reliability.

Various improved AE architectures are available in the literature [213], and their effectiveness in health indicator construction should be researched. For example, Jia et al. demonstrate that a denoising autoencoder can be applied to enhance RE robustness, as the AE captures more stable relationships in the OWT signal during training [270].

5.4. Fault Diagnosis

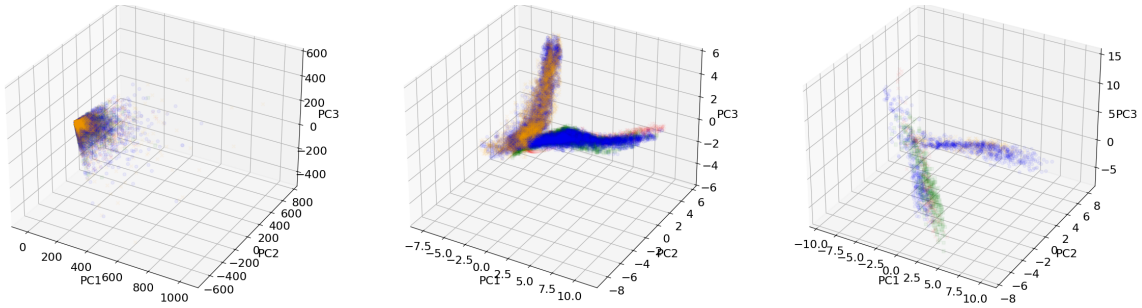
By analyzing the reconstruction-based health signal, the degradation nature can be related to the faulty component. Differences in failure behavior are reflected in the data structure and strengthened by contrastive learning to obtain distinct clusters for each fault.

This section discusses how the reconstruction data is prepared for Deep Embedded Clustering (DEC), before discussing the configurable hyperparameters of the 1D-CNN encoder and clustering results. Then the implementation and results of the classification model are discussed based on these clusters.

5.4.1. Conditioning the AE reconstruction error

While a mixed signal combining MSE and MD provided the best foundation for constructing a stable HI, this signal proved complex for clustering. Figure 5.9a shows how different failure mechanisms are highly overlapping and heavily dispersed, while Figure 5.9b shows more distinct clusters, more suitable for contrastive learning. Differences between clusters remain subtle, however, related to the increased data imbalance of the SCADA dataset and short-lived indicators for faults.

Irrelevant features, unrelated to faults, should be removed to emphasize class distinctions, as shown in Figure 5.9c. First, data related to turbine ambiance is excluded as different turbine operational states have little relation to the location of developing defects. Then, other insensitive features are excluded by calculating a sensitivity metric based on the average value of a feature during failure regions. Of the remaining features, only the sequences that contain alarms are considered to ensure data clusters are learned based on fault-related data.



(a) Mixed Fault Signal (b) AE Latent Space (c) Conditioned AE Latent Space

Figure 5.9: PCA's for three different data sources of Case 1; Blue = Gearbox, Orange = Transformer, Green = Generator, Red = Hydraulics.

5.4.2. 1D-CNN Hyperparameter Configuration

The encoder is configured to maximize clustering accuracy. This involves properly assigning fault types according to the maintenance logs, while avoiding cluster collapse due to loss imbalance, poor initialization, or unstable training dynamics.

A selection of the most influential DEC parameters is presented in Table 5.5, and discussed in this section. Parameter ranges were informed by preliminary testing, insights from the previous chapter, and comparable studies [122, 183]. Boxplots in Figure C.13 validate several design decisions, including the preference for HDBSCAN over DBSCAN, prioritizing optimization of the global (sequence-based) contrastive loss first, and removing sequences that lack alarm events.

Parameter	Min	Max	Step
Number of CNN Layers	1	6	1
Hidden Size	16	256	8
Latent Size	8	64	8
Max Dilation	1	10	2
Convolution Kernel Size	1	12	1
Pooling Kernel Size	1	2	1
Convolution Stride	1	2	1
Loss Temperature	0.2	10	-
Loss Margin	0.2	10	-
Window Size	12	252	12
Step Size	2	24	2
Number of Epochs	5	155	25
Learning Rate	0.0001	0.01	-
Gamma (lr decay)	0.7	0.99	-
Gamma Stop	1	26	5

Table 5.5: Tunable 1D-CNN Hyperparameters with suggested ranges for the optimization algorithm, indicated by upper- and lower bounds, as well as the step size.

Although clustering performance varies only moderately across CNN configurations, the best outcomes were achieved with a relatively large network starting at 184 channels, compressing the input over six layers to a latent dimension of 16. A kernel size of 1 was found to be optimal, as it projects features individually. A dilated convolution rate of 4 allows the network to capture broader temporal dependencies, enhancing the

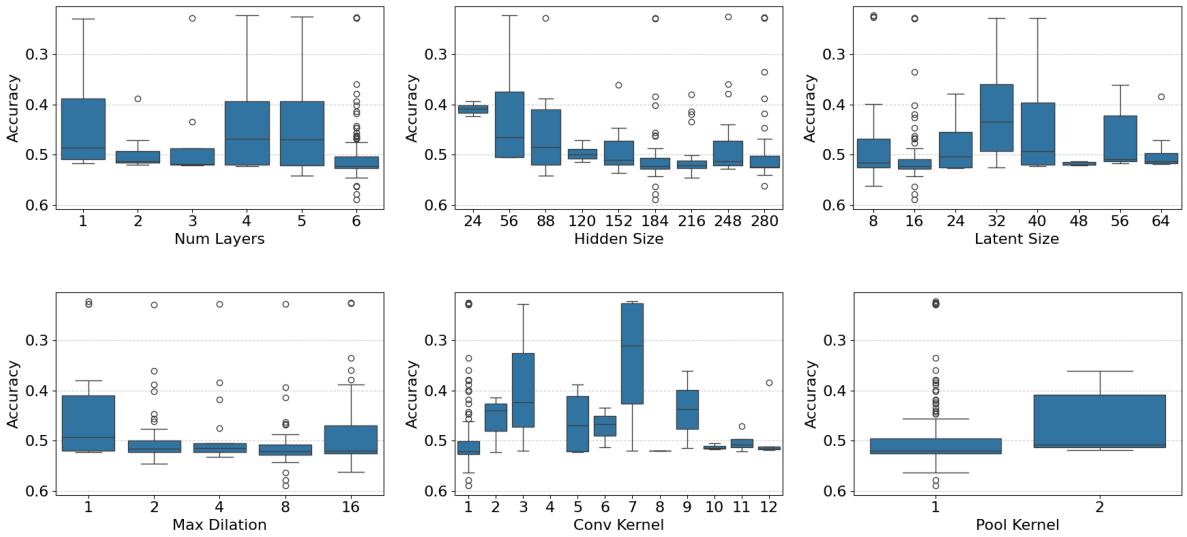


Figure 5.10: Box Plots for 1D-CNN sensitivity after 100 Optuna parameter optimization trials, showing the clustering accuracy per hyperparameter.

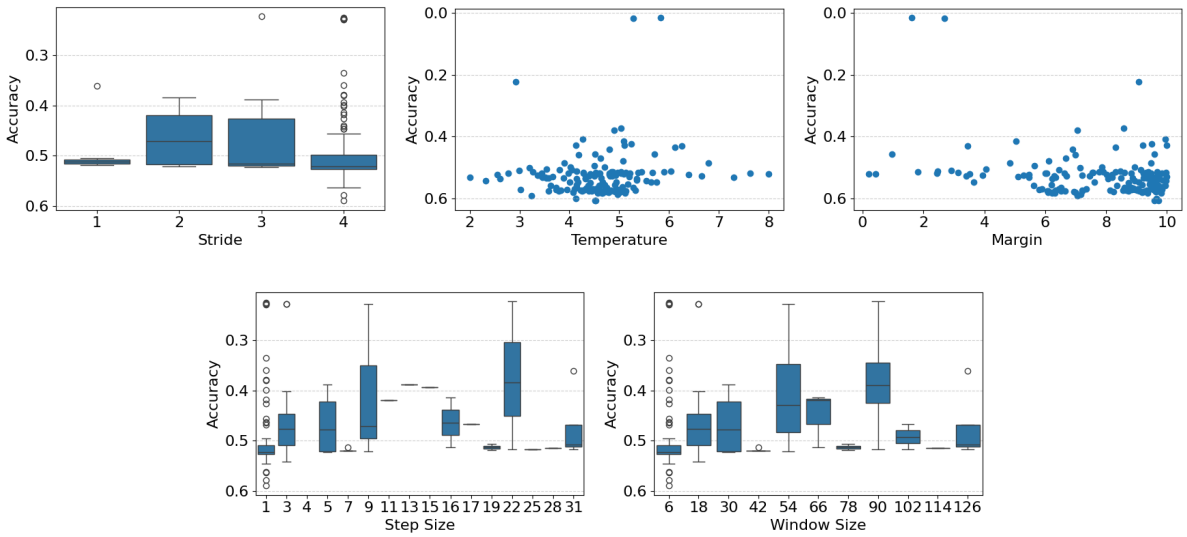


Figure 5.10: Box Plots for 1D-CNN sensitivity after 100 Optuna parameter optimization trials, showing the clustering accuracy per hyperparameter (continued).

modeling of both local and semi-global patterns. A stride of 4 is used to aggressively downsample the input sequence, reducing temporal resolution and yielding more distinct embeddings. To preserve the sequence length and ensure equal attention to edge features, padding is dynamically set to half the kernel size. Finally, flattening operations are adjusted to align with the chosen kernel and stride configurations. Cluster refinement using density-based algorithms (DBSCAN or HDBSCAN) is analyzed in Figure C.11.

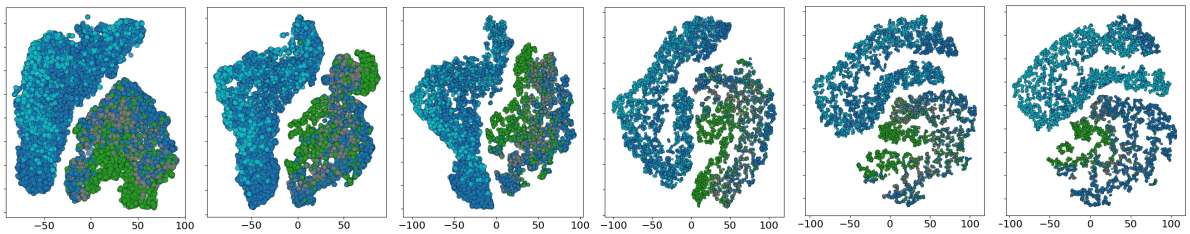
Training settings for the optimizer and learning rate scheduler are shown in Figure C.12, where training with a learning rate of 0.5e-3 for 70 epochs, introducing the local contrastive loss at epoch 62, provides robust results. High values for the temperature and margin parameters of the sequence-based and pairwise contrastive losses ensure significant transformations of the data structure, ensuring increased separation of dissimilar pairs.

Based on the encoded input, repeated training of the FCNN used for classification suggests the optimal balance between interpretation capacity and generalizability is obtained with a single layer of 164 nodes.

5.4.3. Clustering & Classification Results

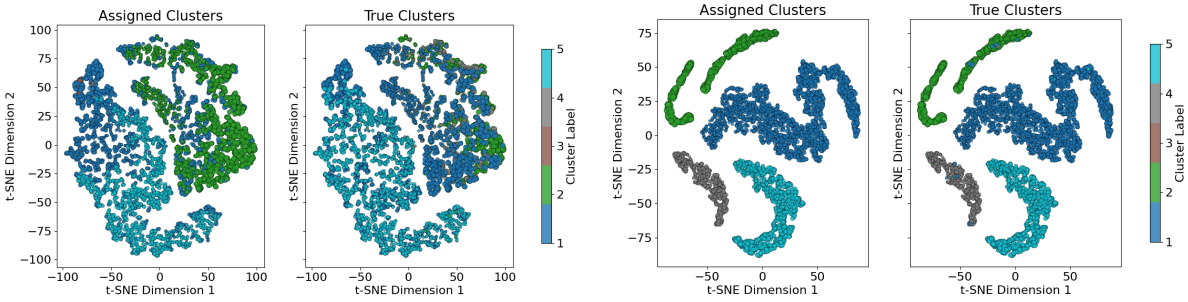
The LSTM-AE signal is directly related to system health. Contrastive learning of this signal is therefore predominantly based on similarity in degradation characteristics and failure mode. Using t-SNE dimensionality reduction, Figure 5.11 demonstrates how samples, marked by their true fault category according to the failure logs, are moved during training epochs to suit the training loss functions. Samples that belong to the different failure modes can be observed to form increasingly well-defined groups over time, first in a global aspect, and then refined locally.

While this shows the model is capable of identifying failure similarity, final label assignment is still required. Due to data nonlinearity, initial k-means-based clusters are recalculated, providing the final cluster



(a) Epoch = 1 (b) Epoch = 10 (c) Epoch = 20 (d) Epoch = 40 (e) Epoch = 60 (f) Epoch = 70

Figure 5.11: t-SNE visualization of 1D-CNN latent space during training of Case 1, using $e_{\text{entry}} = 60$. Blue = Gearbox, Green = Generator, Gray = Hydraulics, Cyan = Transformer.



(a) Fully unsupervised clustering results

(b) Fully supervised clustering results

Figure 5.12: t-SNE visualization of the encoded true clusters compared to the clusters assigned by the model. 1 = Gear., 2 = Gen., 3 = Gen. Bear., 4 = Hydr., 5 = Trans.

assignments. Figure 5.12a compares assigned clusters with their ground-truth, showing cluster separations roughly match true labels. The transformer fault is best defined, as its influence on SCADA sensors varies the most compared to drive-train components that tend to overlap.

Model accuracy, calculated by comparing the inferred label of each sequence to the assumed ground truth, is given in Table 5.6. Both unsupervised and supervised cases are considered. For the latter, data labels are visible to the encoder during training, so that the label guides pair similarity. Naturally, this gives the model much more control over cluster refinement. As shown in Figure 5.12b, if all true cluster labels are known, cluster separation and assignment in the encoder output are close to perfect.

Classification accuracy is given in Table 5.6 for both supervised and unsupervised clustering cases, as well as the accuracy of performing classification directly on the LSTM-AE output, bypassing the 1D-CNN. This shows contrastive learning improves the accuracy of classification, and label availability improves the result.

	Clustering Accuracy		Classification Accuracy		
Case	Unsupervised	Supervised	Unsupervised DEC	Supervised DEC	Bypassed DEC
Case 1	55-61%	92-97%	0-40%	30-42%	10-25%
Case 2	56%	97.9%	39%	48%	31 %
Case 3	60.5%	93.7%	40%	46%	33%
Case 4	68.3%	90%	33%	36%	31%

Table 5.6: Percentage of correctly categorized sequences and the number of inferred clusters, for supervised or unsupervised approaches to each case. The accuracy of case 1 gives the range following n=5 tests.

This relationship between label availability and clustering accuracy is visualized in Figure 5.13, where the outcomes of different hyperparameter optimization studies are plotted and interpolated, indicating a positive relationship between the share of visible labels and clustering accuracy.

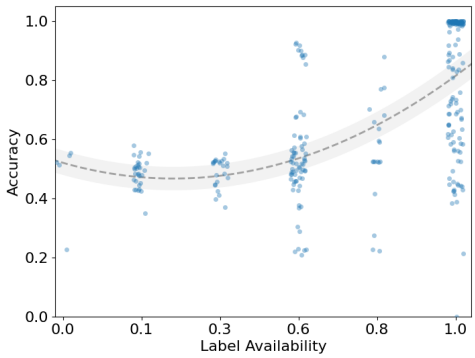


Figure 5.13: Scatterplot of trial accuracy at varying levels of observable labels during training of case 1. Jitter is added to the scatter markers for clarity.

5.4.4. Discussion

While results indicate that DEC of LSTM-AE features leads to a higher proportion of correctly classified test samples, the current application is not yet fully reliable. In particular, the absence of prior knowledge regarding the number of failure modes introduces the risk of cluster collapse and misclassification across large portions of the dataset during encoder training. While these unaligned clusters contribute to the error, they may represent early indicators of novel fault types or alternative degradation paths within the same component, requiring careful consideration.

To improve clustering robustness, adding temporal reasoning in the encoder can introduce increased responsiveness to the dynamic nature of degradation processes. This motivates the use of hybrid architectures—such as CNN-LSTM models—as proposed in studies like [145, 146].

In this work, contrastive losses originally developed for image data have been repurposed for time-series applications. These losses typically rely on distance-based similarity, which might overlook the temporal and contextual complexity of SCADA data. Developing a loss function that accounts for these specific characteristics could enhance the discriminative power of the encoder.

Although encoder improvements enhance classification performance, the current results suggest the classification approach should undergo additional improvements as well. A broader investigation into model architectures or alternative classification strategies may yield more effective solutions for the intended application. Since class distinctions in the latent space are inherently distance-based, a classifier that leverages spatial relationships would be appropriate.

Likely to influence the classification accuracy, the classifier should be sensitive to the confidence of cluster assignments. Due to fault progression and the soft boundaries of similarity-based latent space, many sequences will lie in transitional states that don't clearly belong to a single class, and are often misassigned. Furthermore, an uncertainty-aware system improves resilience to samples with low classification confidence, such as underrepresented failure modes or novel faults not captured by existing clusters.

5.5. Fault Prognosis

RUL prediction involves learning the relationship between historical features and the corresponding time to failure. This section covers the selection and sensitivity of Transformer hyperparameters and the RUL results.

5.5.1. Transformer Hyperparameter Configuration

Properly specifying Transformer parameters, detailed in Table 5.7, involves minimizing the validation loss, ensuring relevant features can be extracted and translated to RUL values, while ensuring it generalizes to unseen inputs.

Due to the quadratic relationship between Transformer complexity and sequence length, longer lifespans in this dataset significantly increase computational demands. During hyperparameter optimization, 30% of the generated sequences are randomly sampled to ensure realizable computation times.

Parameter	Min	Max	Step
Number of Encoder Layers	1	4	1
Number of Decoder Layers	1	4	1
Model Dimension	16	48	8
Number of Heads	6	14	2
Feedforward NN Dim	16	64	8
Window Size	50	140	30
Step Size	40	64	8
Max Rul	14e3	20e3	1e3
Number of Epochs	125	250	25
Learning Rate	1e-4	1e-3	-
Gamma (lr decay)	0.8	0.99	-
Gamma Stop	5	40	5
Dropout	0.01	0.5	-

Table 5.7: Tunable Transformer Hyperparameters with suggested ranges for the optimization algorithm, indicated by upper- and lower bounds, as well as the step size.

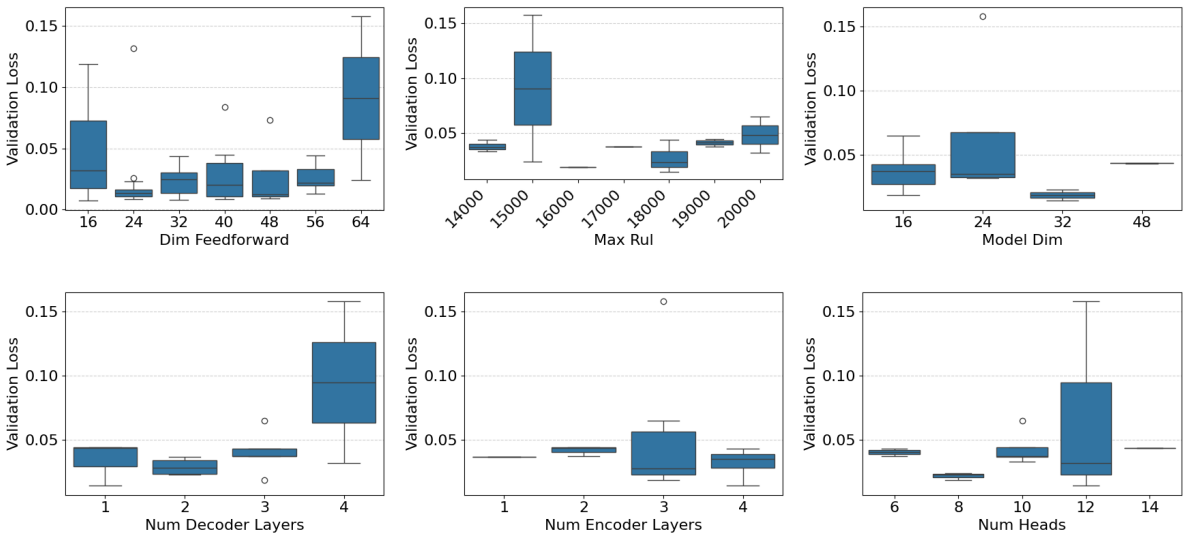


Figure 5.14: Box Plots for Transformer sensitivity after 100 Optuna parameter optimization trials, showing validation loss per hyperparameter.

Due to the subtle nature of degradation in SCADA data, the model is prone to overfitting and exposure bias, limiting the possible size of the optimal transformer architecture. Figure C.14 shows an optimal model width of 32 nodes, comparable to the number of input features. This indicates that the LSTM-AE signal is already informative and normalized, and input data does not benefit from compressing or expanding. Eight attention heads are selected, each operating in a small 4-dimensional subspace (32/8).

Three encoder layers ensure the model is capable of interpreting data relationships without over-complicating training. Two decoder layers suffice for prediction, which involves generating the RUL signal. The optimal feed-forward layer size of 24 further reflects the task's low nonlinearity and suggests that the attention layers extract clear, well-defined EoL indicators.

After studying model sensitivity to training parameters, boxplots in section C.12 confirm that precise tuning of the narrow transformer is required to avoid over-dependency on the training case and its input targets. A low learning rate with fast decay and 150 training epochs is optimal to balance convergence and generalization while mitigating overfitting and exposure bias.

While larger windows increase the amount of data for training, they limit batch sizes to fit GPU memory. This reduces the amount of processed data per epoch, which in turn reduces the learning rate. Smaller step sizes increase overlap and prediction confidence, at a higher computational cost. Balancing both capacity and computational limitations, a window size of 136 and a step size of 40 are selected.

RUL labels derived from detected faults, as described in the previous chapter, are capped and normalized to stabilize training. The optimal cap is expected to lie between 14,000 and 20,000 cycles, scaled relative to the increased lifetime lengths of this case study.

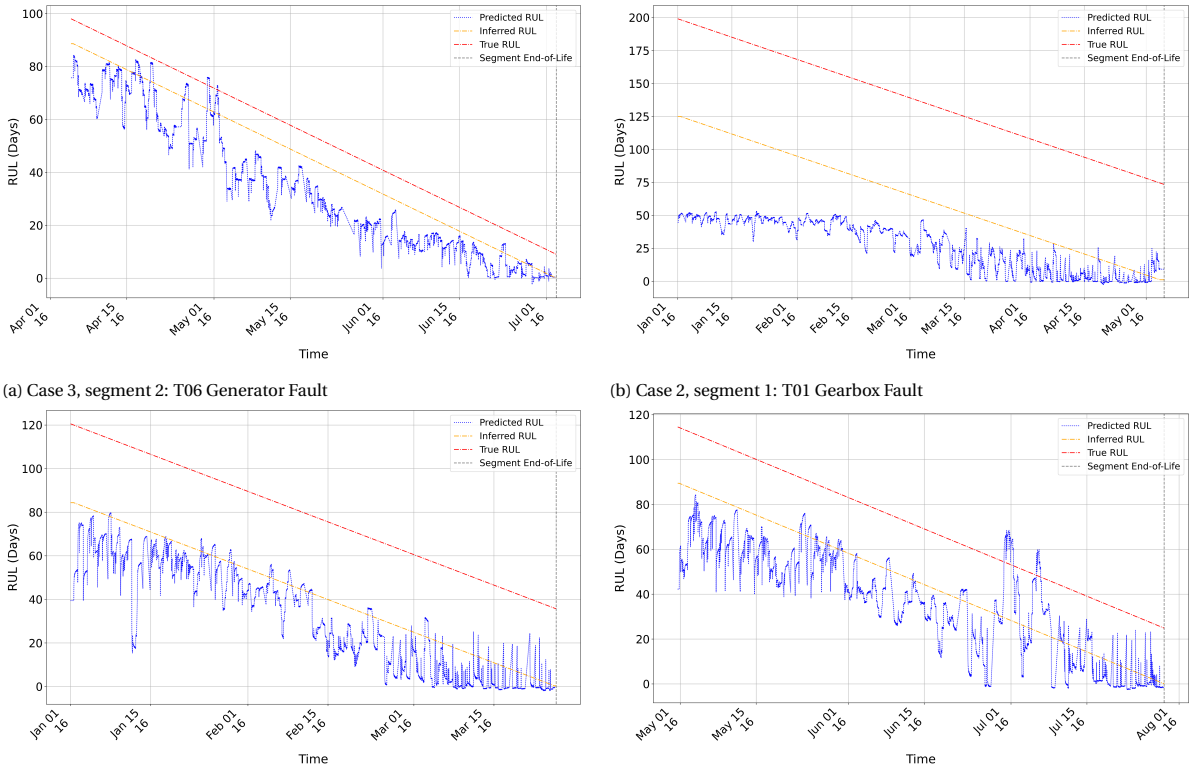
5.5.2. RUL Prediction Results

The Transformer is trained and evaluated across the four fault categories using the LSTM-AE-based health signal as input. Figure 5.15 visualizes predicted RUL—expressed in remaining operational days—for faults in the gearbox, generator, generator bearing, and transformer. A detailed example of the full RUL prediction pipeline is shown in Figure 5.17, highlighting input signals, reconstruction-error-based health estimation, fault detection, and RUL prediction for a hydraulic system failure.

Additional results are provided in section C.12. These suggest hydraulic- and generator faults yield the most consistent predictions. This likely relates to the higher number of fault examples for these components, emphasizing the importance of data availability for effective training.

In contrast, gearbox faults show the least reliable results. This is partly due to data imbalance, as only one gearbox failure remains after exclusion of the undetected T06 gearbox fault, which reduces model sensitivity to gearbox-specific features. This results in a poor prediction illustrated in Figure 5.15b.

Prediction accuracy is quantified using RMSE across timesteps per case and given in Table 5.8. Next to



(a) Case 3, segment 2: T06 Generator Fault

(b) Case 2, segment 1: T01 Gearbox Fault

(c) Case 4, segment 1: T07 Generator Bearing Fault

(d) Case 4, segment 2: T07 Transformer Fault

Figure 5.15: Four RUL prediction examples, showing predicted RUL values (blue), RUL targets based on detected faults (orange), and true segment RUL (red).

the RE-based signal, results are compared across raw input features and LSTM-AE latent space input as well. Each is evaluated both in an unsupervised setting, using RUL labels based on LSTM-AE fault detection, and in a supervised setting based on ground truth annotations from the maintenance logs. A comparative overview is visualized in Figure 5.16.

Case 1 achieves the highest prediction accuracy, likely because the test turbine (T11) contains generator and hydraulic faults, which are better represented in the training data. The combination of LSTM-AE-derived input and labels performs best, while RE-based inputs generally outperform latent space inputs.

The best RMSE values observed in this chapter are comparable to those in the C-MAPSS case study in Table 5.8[1]. This alignment supports the feasibility of SCADA-based RUL prediction.

Interestingly, the raw data performs better when trained on inferred labels than on the maintenance-based ones. This suggests that, in some cases, the fault detection model may identify degradation earlier than the actual intervention by the maintenance team.

	Raw Data		LSTM-AE RE		LSTM-AE Latent Space	
	Unsupervised	Supervised	Unsupervised	Supervised	Unsupervised	Supervised
Case 1	20.99	36.74	18.57	30.90	22.31	38.94
Case 2	35.02	41.76	33.01	46.83	35.01	44.57
Case 3	44.22	35.02	31.87	27.15	28.44	37.64
Case 4	28.24	36.49	25.64	29.32	26.63	22.84

Table 5.8: RUL prediction RMSE in days, for RUL prediction based on raw data and LSTM-AE based RE and Latent Space datasets.

[1] Given that RMSE is reported in days, it can be rescaled to RMSE per measurement and scaled to the C-MAPSS RUL target range using the cutoff RUL values: $e_{EDP} * 144/18000 = e_{NASA}/125$

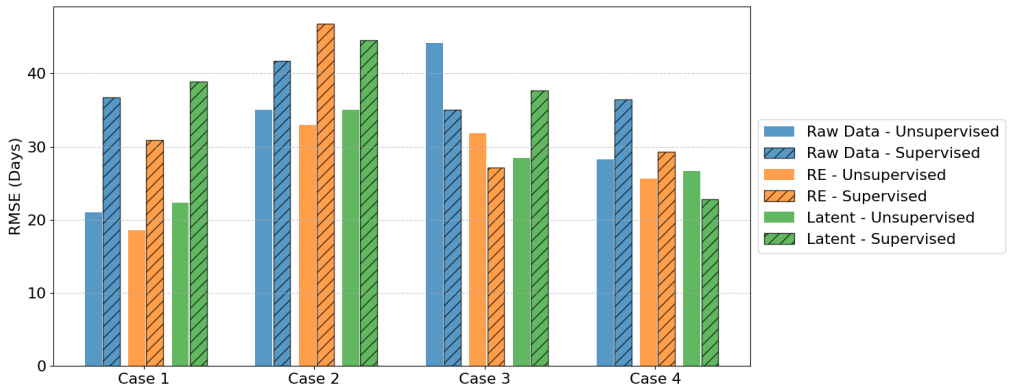


Figure 5.16: Histogram of Transformer Results for each testing case and input dataset.

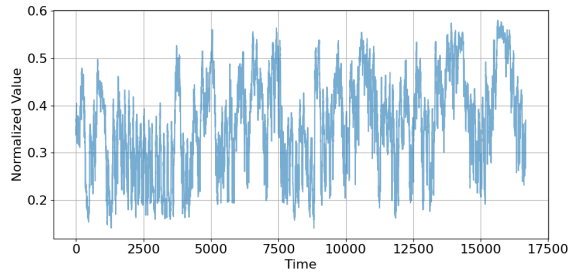
5.5.3. Discussion

While the proposed approach to RUL prediction achieves the highest accuracy, there are still challenges in ensuring reliability. As shown in Figure 5.17b and Figure 5.17d, RUL prediction is highly sensitive to the quality of the RE signal and inferred RUL labels. Inaccurate labels misguide the model, while spikes caused by unrelated disturbances are misinterpreted as early degradation. Insensitivity of the LSTM-AE to actual defects, as in Figure 5.15b, hinders fault detection entirely as no informative features are communicated to the prediction model.

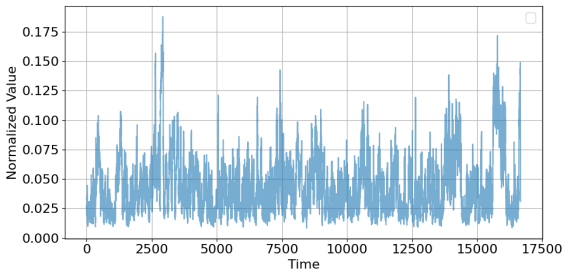
Improved data quality and additional failure examples contribute to achieving competitive results. With an increasingly rich and reliable dataset, more complex Transformer architectures can be considered for improved modeling of features and their temporal relationships.

Alternative architectures, such as CNNs or LSTMs, could also be explored within each feed-forward block to enhance representational power. Here, the implementation of uncertainty quantification models can also be considered to ensure reliable model output for risk evaluation and maintenance scheduling.

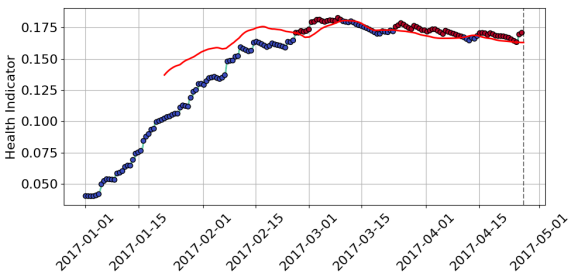
Post-processing of the RUL signal can reduce output variability as well. For instance, applying a weighted moving average reduces fluctuations, improving stability and avoiding premature predictions, such as observed in Figure 5.15d.



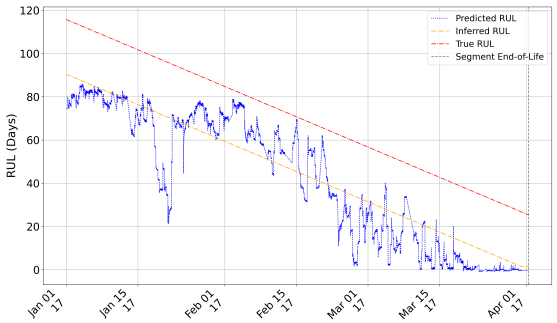
(a) Normalized average of raw inputs



(b) Health Signal based on the LSTM-AE Reconstruction Error



(c) A detected fault based on the Health Indicator



(d) Estimated RULs at every cycle

Figure 5.17: The steps taken to achieve fault detection and an RUL estimate of Case 1, segment 3: Hydraulic Fault

5.6. Concluding Remarks

This chapter has demonstrated the implementation of the proposed framework for anomaly detection, fault diagnosis, and prognosis of real-world OWT SCADA data. Results confirm that the framework is feasibly applicable to such data and capable of generating interpretable outputs.

Compared to the simulated dataset in the previous chapter, the SCADA data presents unique challenges. Short-lived, noisy features that occur in brief peaks create an intricate and unbalanced problem. This causes reduced reliability of the HI, opposing responsiveness of the fault detection, and obscures failure-specific features during early stages of fault development, reducing the accuracy of diagnosis and RUL prediction.

5.6.1. Discussion

The preceding sections identified several directions for potential model improvements, which are summarized here. Then, insights gained from both case studies further inform recommendations for future research.

Opportunities for Model Improvements

Challenges of data imbalance could be mitigated through additional feature extraction or conditioning steps, as well as employing more advanced autoencoder architectures to improve reconstruction accuracy. Refinement of the fault decision criterion should also be considered, as well as the intelligent interpretation of generated alarms.

For the DEC model, incorporating temporal sensitivity would enhance cluster reliability, enable distinction between early and late fault stages, and improve responsiveness to subtle degradation patterns. Further improvement of classification accuracy involves improving the model's responsiveness to spatial features.

Although improvements in the reconstruction error signal and fault detection appear to provide the greatest contribution to RUL performance, larger datasets and advances in Transformer architectures will further increase the stability and practical applicability of prognostic outputs.

Directions for Future Research

Unavailable or unreliable fault labels in SCADA datasets argue for an unsupervised approach. Still, partial label availability can significantly improve diagnostic and prognostic accuracy. Therefore, future work could investigate semi-supervised methods that leverage partial labels, derived from alarms or maintenance logs.

While deep learning models typically reduce data processing demands, intermediate preprocessing between framework stages is essential to ensure the informativeness of each signal. This thesis has shown that even in deep learning pipelines, context-aware data handling can greatly contribute to improved experimental results.

Treating hyperparameter configuration, training, and testing of each sub-model independently improves transparency but increases the risk of information loss and the propagation of uncertainty between stages. For real-world deployment, it is essential to quantify uncertainty both locally and across the full framework, thereby supporting robust O&M decision-making.

In addition, current optimization of individual models may lead to overspecification, limiting generalization to other tasks. Future work could therefore investigate collective or end-to-end training strategies based on global optimization objectives, enabling better coordination across tasks.

This also applies to the present case study, where model design might be overly specified on the EDP dataset, and confirmation bias may occur during post-processing. Future research should validate the framework on diverse case studies to ensure generalizability and account for dataset-specific inaccuracies such as imperfect maintenance records, unreported faults, cross-turbine data transfer issues, and sensor errors. Broader validation would also mitigate the stochastic variability of deep learning training and hyperparameter optimization, thereby increasing confidence in both the results and the underlying design choices.

6
Conclusion

Effective maintenance decision-making requires condition monitoring methods that accurately reflect wind turbine health. Unlike vibration or acoustic emission monitoring, which often requires additional hardware and incurs higher costs, a turbine's SCADA system offers a cost-effective and readily available alternative. However, challenges such as poor data quality and the absence of labeled fault data complicate its direct application. To address this, a data-driven framework for the unsupervised interpretation of Offshore Wind Turbine (OWT) SCADA data has been developed.

The literature review highlights how deep learning methods, capable of capturing the interrelationships in high-dimensional SCADA data, can be combined to achieve anomaly detection, fault diagnosis, and Remaining Useful Life (RUL) estimation.

An Auto-Encoder (AE) with time-sensitive LSTM nodes interprets the raw signal to obtain a refined signal directly related to the health of the system. This signal can be interpreted as a Health Indicator that indicates failure timesteps by applying a fault criterion.

To obtain a diagnosis, contrastive learning is applied with a 1D-Convolutional Neural Network to obtain failure-mode-specific clusters. The relationship between formed clusters and their embedding is learned by a classification neural network, so that the most likely failure mode of newly input data samples can be identified.

Historic faults, detected by the LSTM-AE, are used to determine the RUL values of preceding timesteps to inform a Transformer, which determines the associated RUL values of input sequences to evaluate the maximum available time for maintenance.

To validate the framework under controlled conditions, NASA's C-MAPSS simulated engine dataset was used first. These results indicate the approach provides a competitive and sufficiently reliable means of data-based health monitoring, achieving performance levels comparable to an extensive collection of similar studies.

Application to real-world OWT SCADA data demonstrates the framework's feasibility in practical settings. Compared to the simulated case, SCADA-based challenges introduced an increased data imbalance and less clearly defined features, complicating the training of each model. This results in decreased reliability of the health indicator, reduced responsiveness of the fault detection, and obscuring of failure-specific features during early stages of fault development, which reduces the accuracy of diagnosis and RUL prediction.

Despite these limitations, this study shows that deep learning techniques can extract valuable health information from SCADA data. The proposed framework enables predictive maintenance optimization, with the potential to reduce costs and enhance system reliability in offshore wind operations.

A
Academic Paper

Unsupervised Fault Diagnosis and Remaining Lifetime Estimation for the Predictive Optimization of Offshore Wind Turbine Maintenance.

J.B. Hes, X. Jiang, M. Borsotti
Maritime and Transport Technology, Delft University of Technology, The Netherlands
j.b.hes@student.tudelft.nl

September 8, 2025

Abstract
Effective maintenance decision-making depends on the timely and intelligent anticipation of developing faults. Without requiring the installation of additional sensors, failure-related information can be extracted from the widely available Supervisory Control and Data Acquisition (SCADA) system. This work presents an integrated deep learning framework designed to interpret high-dimensional, unlabeled, and often low-quality SCADA data for anomaly detection, fault diagnosis, and Remaining Useful Life (RUL) estimation.

Its application to real-world offshore wind turbine (OWT) SCADA data demonstrates its practical feasibility. Despite challenges such as data imbalance and obscured features due to SCADA data quality issues, the framework effectively extracts health-related insights, enabling predictive maintenance optimization.

1. Introduction

Wind energy plays a vital role in meeting the rising demand for renewable energy. With Operation & Maintenance (O&M) costs comprising 25%–50% of total energy generation costs, reducing expenses due to equipment failures is critical [4]. Unexpected failures lead to downtime, costs, and safety risks, while physically inspecting wind turbines introduces significant costs [20, 21].

To mitigate these challenges, the wind industry is moving toward data-driven, predictive maintenance (PdM) strategies—where Condition Monitoring (CM) remotely informs the optimization of servicing schedules, ultimately offering a more cost-effective maintenance solution [66, 67].

Without requiring additional hardware, incurring costs and complexity, a turbine's Supervisory Control And Data Acquisition (SCADA) system can provide a cost-effective and readily available means for real-time fault detection and prognosis [57]. A common approach to capture the relationships in these sensor signals is Machine Learning (ML). However, as traditional ML method effectiveness relies heavily on the user's capability of supplying informative and reliable features, deep learning models have seen increased use [104]. Without relying heavily on expert knowledge or manual feature extraction, they excel at extracting high-dimensional and nonlinear patterns for regression, classification, and clustering tasks, outperforming traditional ML, statistical, and physical models, particularly for datasets with minimal or no labels [106–108]. As a result, Deep Learning is increasingly applied in fault diagnosis [109–111] and prognosis [112–114] of mechanical systems.

1.1. Scope of this Thesis

Significant advances have been made on both fronts of offshore wind PdM: A wide range of models, methods, and strategies that are developed to optimize offshore wind O&M decisions [18–21]. At the same time, sensor- or SCADA-based WT CM is a widely discussed topic on the other [40, 43, 57]. However, these areas are often studied in isolation, with limited integration between CM methodologies and the execution of maintenance actions.

Therefore, this thesis contributes to the development of an integrated framework that allows for informed data-driven maintenance decision-making. It bridges the gap between CM and maintenance planning by proposing a data-driven methodology for fault detection, diagnosis, and Remaining Useful Life (RUL) prediction.

Given the complexity of offshore environments and the unpredictable nature of wind turbine faults, the proposed framework is first validated using NASA's C-MAPSS simulated aircraft engine dataset in the full version of this thesis. Results indicate that the approach provides a competitive and sufficiently reliable means of data-based health monitoring, achieving performance levels comparable to an extensive collection of similar studies.

2. Method Selection

Data-driven inference is primarily influenced by the availability of labels, which can serve as failure examples that the model learns to recognize [115]. Incomplete or missing labels, dataset imbalances, and sensors with limited responsiveness to early-stage fault developments necessitate alternative approaches to solve the task, such as unsupervised clustering or reconstruction tasks, or learning an intended purpose through comparative methods [116].

For this reason, the fault or anomaly detection model serves as a feature extraction step. Application

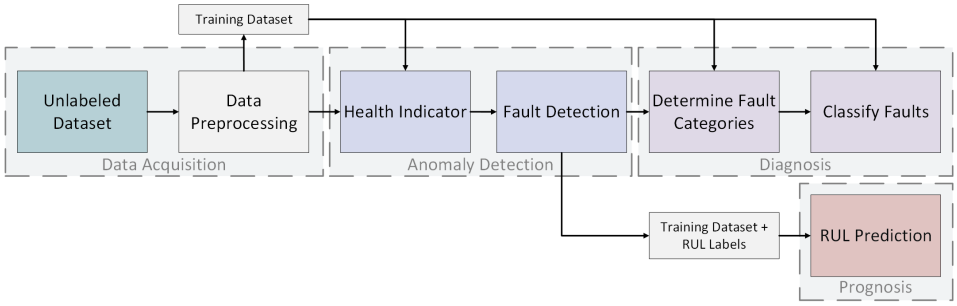


Figure 1: Four characteristic steps of obtaining a fault diagnosis and RUL prediction.

of the fault detection model to historic data provides RUL labels available for prognosis, while feature extraction enriches the signal for diagnosis and prognosis tasks. Lacking fault labels, fault diagnosis requires a clustering step as well, which is used to obtain fault-specific training labels for classification. This way, these tasks are combined as illustrated in Figure 1, allowing for a deeper understanding of system capabilities, reduced complexity of the task, and optimal matching of methods [175].

2.1. Anomaly detection

As an automated feature extraction step, responsive to deviations from a healthy state, implementation of an unsupervised anomaly detection method increases robustness to varying and possibly unprecedented conditions and failure mechanisms [176]. Even in the presence of labeled SCADA data, label reliability is not a guarantee, where unsupervised methods are more robust to possible uncatalogued anomalies or label inaccuracies [177].

Schlechtingen and Santos, as well as Correa et al., demonstrate that multiple variations of a standard neural network can be applied to monitor a variety of WT component failures from SCADA data [65, 121]. Increasing the number of layers of an NN can improve the understanding of representations of extensive SCADA data sets [122]. However, they are limited in handling temporal dependencies or complex, dynamic systems.

Belief networks learn the probability density of a healthy state, capable of automatically extracting degradation features [136]. This makes DBNs well-suited for fault detection through modeling or learning normal behavior of WTs from SCADA data [138–140]. However, belief networks are difficult to implement in multi-variate systems, struggle in evaluating long-term dependencies, and are vulnerable to disturbances and noise [114].

Auto-Encoder

An Auto-encoder (AE) encodes and decodes input data, learning essential parts of the data in the process. By comparing the reconstructed output with the original input, Wang et al., as well as Zhao et al., track this reconstruction error (RE) to derive an indicator for turbine component failure [125, 126]. Renström demonstrates that a single model comprising multiple AE layers is capable of automatically detecting faults in many different components of a WT [127].

One approach is to include LSTM nodes in the structure, improving the extraction of temporal relations as demonstrated by de Pater & Mitici [131]. The LSTM cells allow the retention of important information while discarding irrelevant data [148, 207, 212].

Predictions based on these features often require the integration with time-sensitive approaches [119]. Therefore, AEs are frequently considered as a feature extraction step, improving predictions based on SCADA data for WT blade damage [134] and WT RUL prediction [135]. Through the unsupervised extraction of features from high-dimensional data, AEs are highly valuable for degradation tracking and health monitoring of mechanical systems.

2.2. Fault Clustering

As information on historic failure examples is often unavailable, the degradation behavior of an input signal should be interpreted. Originating from computer vision [182], contrastive learning methods are promising for fault diagnosis of unbalanced and unlabeled SCADA data, beneficial to the accomplishment of classification and scheduling tasks [122, 183, 184].

In time series, an advanced application of contrastive learning is deep embedded clustering (DEC). Through an encoder, data is mapped into a latent space where clustering is more effective by satisfying a discriminative loss function [187–189]

Convolutional Neural Networks

DEC is often performed by Convolutional Neural Networks (CNNs), as the encoder should be adequately capable of the spatial features of each failure mechanism in both long- and local term [144]. Cluster

coherence is improved because the filters (convolutions) can capture both subtle, gradual degradation patterns and sudden changes in the input data, while preserving temporal relationships.

Excelling in feature extraction and spatial pattern recognition, CNNs are highly capable for contrastive learning, while Xiang et al., Kong et al., and Sun et al. demonstrate CNN effectiveness in WT SCADA feature extraction [51, 145, 146].

A set of data points at the same time forms a one-dimensional signal, so a compact 1D-CNN is applied. These offer reduced complexity and computational cost, while demonstrating capable performance on multi-variate applications with limited labeled data and high signal variations [190].

2.3. Fault Classification

As a comparative method, data clustering is complicated by low data quantities, prone to transfer issues when comparing multiple turbines, and sensitive to uncertain or overlapping failure mechanisms. Therefore, inclusion of a final classification layer improves class separation and model robustness by allowing for improved reasoning and interpretation of clusters during real-time implementation.

When the trained encoder embeds future samples, the classification task involves assigning them to the most likely class. Because of the relatively low complexity of this task, a standard feed-forward, fully connected, NN is applied, able to learn the characteristic differences between different clusters in the embedding, and update its training based on new information for diagnosis of new problems [105].

2.4. Prognosis

Prediction of future RUL values involves extracting a relationship between distinctive features and time. Commonly, RUL prediction methods follow supervised learning. By analyzing current and historical run-to-failure data profiles, the degradation trend and corresponding point of failure can be estimated.

Sequential Networks

To model time-dependent behavior in sequential data, Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU) networks have been developed, effective in capturing medium-term dependencies [149–151].

Especially LSTMs are considered well-suited for RUL estimation using sensor data [147], while Hsu et al. demonstrate promising results in SCADA data-driven regression [148]. Combining multiple layers increases the reasoning for advanced time-based feature extraction in RUL prediction [149, 153, 155, 156].

While LSTMs and other recurrent networks are popular for tasks that require temporal sensitivity, sequential processing of data can be slow, limit truly long modeling, and increase sensitivity to data imbalance and overfitting.

Attention-based Models

The concept of attention allows the model to focus on the most relevant parts of the input, proven to enhance model reasoning [157–161].

Through increased computational efficiency of attention, Transformers can handle large computational tasks. Although the primary focus of transformers has been in natural language processing and image recognition, their potential in PdM has recently gained attention [163, 164, 166]. Several studies have explored combining the Transformer with an AE to extract fault information and predict RUL [167–169].

While Transformers have shown promise in power forecasting [170–172], the development of transformers for Wind PdM remains behind. Zhao et al. utilized transformers for gearbox fault detection through predicted temperature [173], while Zheng et al. implemented a semi-supervised anomaly detection method using only a small amount of labeled data [174].

The true potential of Transformers for SCADA analysis still requires research. Still, comparison of initial results shows consistent improvements in terms of training speed, predictive performance, and data requirements, enhancing temporal pattern recognition and improving RUL prediction accuracy.

3. Model Design

Following the sequence of Figure 1, after discussing preprocessing, in-depth descriptions are provided for the selected methods, covering model architecture, training, and implementation to obtain the complete framework architecture depicted in Figure 2.

3.1. Dataset Preprocessing

In data-driven applications, proper conditioning of the multivariate SCADA input signal is a critical step in ensuring optimal model performance and reliable results. After conditioning, the sliding window technique generates overlapping sequences of length m using a stride or step size s, which can be divided into training, testing, and validation datasets [209, 210].

Normalization

To ensure stable and efficient training of machine learning models, data inputs should be normalized to reduce these variations and reduce the risk of scaling and gradient-exploding problems. The choice of normalization method depends on the nature of the data and can significantly impact model performance.

Sklearn's "RobustScaler" centers on the median and scales by the interquartile range (IQR), ensuring that outliers, prevalent in SCADA data, have less influence on the scaling process, while preserving essential data characteristics [208], as follows:

$$x' = \frac{x - \text{median}(x)}{\text{IQR}(x)}$$

3.2. LSTM-AE for Anomaly Detection

To identify behavior that deviates from normal, an Auto-Encoder (AEs) with LSTM nodes (Figure 2(a)) is trained to reconstruct healthy data, such that the reconstruction $\hat{\boldsymbol{x}}$, matches the input signal $\boldsymbol{x}$. When training, model parameters are adjusted by calculating their derivative with respect to the output, which propagates backward throughout the model. Model weights and biases are adjusted to minimize a loss function based on the Mean Squared Error (MSE):

$$\mathcal{L}_{\text{r}}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{n}\sum_i (\boldsymbol{x_i} - \hat{\boldsymbol{x}}_{\boldsymbol{i}})^2$$

Health Indicator Construction

For an AE trained to reconstruct healthy data, the RE is directly linked to defects or other operational anomalies. It can be interpreted as a quantified difference between the in- and output of the AE, measured by methods such as Euclidean distance [128, 130, 215, 216]. Another measure for dissimilarity is the Mahalanobis distance (MD), which considers the covariance structure of the data, ensuring that highly correlated signals do not dominate the results [127, 217, 218].

If C_x is the covariance matrix of the AE reconstruction error $\boldsymbol{r}_i = \boldsymbol{x}_i - \hat{\boldsymbol{x}}_{\boldsymbol{i}}$, the MD for each feature i can be calculated as:

$$MD_i = \sqrt{\boldsymbol{r}_i^T C_x^{-1} \boldsymbol{r}_i}$$

To improve the stability and fault responsiveness of the HI, additional smoothing and normalization is applied, fitted on normal bounds identified by validation data passed through LSTM-AE.

Then an exponentially weighted moving average (EWMA) produces the HI (Figure 2(b)) [125, 127, 215]. Given the weight of newly introduced values λ, a smoothed projection of the MD z is obtained as follows:

$$z_t = \lambda MD_t + (1-\lambda) z_{t-1}$$

Fault Detection

Finally, to decide between acceptable levels of deviations and failure, a fixed fault threshold can be defined, functioning as an alarm decision criterion [126].

To address limitations of traditional constant thresholds, an adaptive thresholding approach is proposed. Inspired by Liu et al., this threshold dynamically adjusts based on recent data trends, captured by the statistical properties of recent observations [220]. At a specific time t, the threshold analyzes a predefined window of historical HI values of size w to find a mean μ_{t-w} and standard deviation σ_{t-w}. Combined with a sensitivity parameter κ that controls the threshold's responsiveness to variations, the threshold value is determined as:

$$T_i = \mu_{t-w} + \kappa \cdot \sigma_{t-w}$$

Fault detection is determined by a decision-criterion based on the smoothed HI value z and the determined fault threshold T, where an alarm is raised if:

$$\text{Alert} = \begin{cases} 1 & \text{if } z_t \geq T, \\ 0 & \text{otherwise.} \end{cases}$$

3.3. 1D-CNN for Fault Clustering

While the RE is very suitable for HI construction, SCADA data imbalance and short-lived fault indicators result in highly overlapping and dispersed fault clusters. Therefore, DEC-based diagnosis of SCADA data involves clustering of the LSTM-AE latent space, which provides rich health-related information [132, 133]. A 1D-CNN-based encoder is defined to obtain a latent interpretation of the input data, where differences in failure characteristics are amplified (Figure 2(c)) [186, 227, 229].

In the process of mapping input data to a lower-dimensional space, the model is trained to iteratively optimize a self-supervised clustering objective of two contrastive loss functions, which allows for the optimization of larger and smaller data structures [228, 229].

A "Non-Parametric loss" ($\mathcal{L}_{seq}$) is calculated between sequences, refining the global structure of the latent space using Softmax-normalized similarity scores [186, 234]. To flexibly learn meaningful representations without needing categorical labels, cosine similarity is computed as the dot product of normalized latent representations $S_{i,j} = \frac{z_i \cdot z_j}{||z_i|| ||z_j||}$, applied between pairs z_i and z_j, scaled by a temperature parameter T, encouraging each sequence to be closer to its most similar counterpart:

$$\mathcal{L}_{\text{seq}} = -\frac{1}{N}\sum_{j=1}^{N} \log \frac{\exp(S_{j,i^*}/T)}{\sum_{i=1}^{N} \exp(S_{ji}/T)}$$

A "Pair loss" (L_{pair}) acts at a finer granularity, ensuring small clusters form around data points that

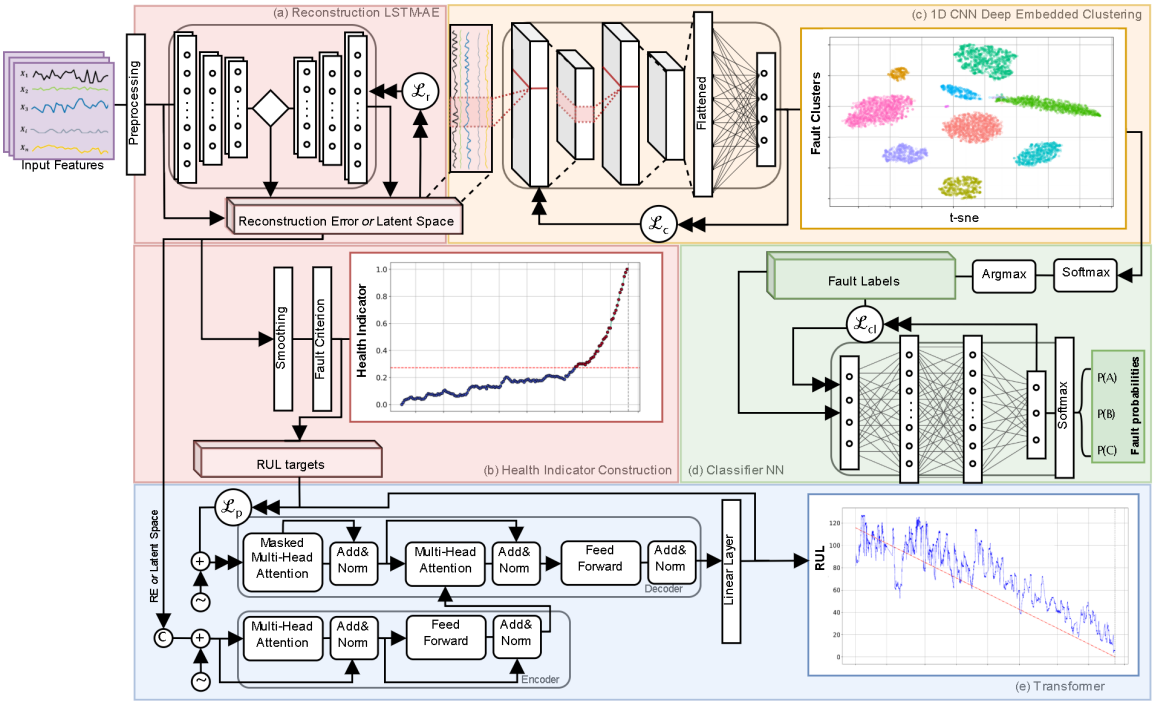


Figure 2: Key steps in the framework: (a) anomaly detection, (b) health indicator construction, (c) fault clustering, (d) classification, and (e) RUL estimation.

share common characteristics. Similarity is calculated by Euclidean distance $D(i,j) = ||i-j||^2$ between pairs i and j in the embedding space. Given a number of samples N, the distance between similar pairs S_{ij} is minimized while the distance between negative pairs is encouraged to be larger than a given margin m:

$$\mathcal{L}_{\text{contrast}} = \frac{1}{N}\sum_{i,j} S_{ij} D_{ij}^2 + (1-S_{ij})\max(0, m - D_{ij})^2$$

During training, at a certain training epoch e_{entry}, the loss shifts from global to the local optimizations during η epochs at a rate $\alpha(e) = \min(1, \frac{e-e_{\text{entry}}}{\eta})$. Considering an optional gain g to balance out the difference in loss range, the contrastive loss is defined as:

$$\mathcal{L}_{\text{c}}(e) = \begin{cases} \mathcal{L}_{\text{seq}} & \text{if } e < e_{\text{entry}} \\ (1-\alpha) g \cdot \mathcal{L}_{\text{pair}} + \alpha \cdot \mathcal{L}_{\text{seq}} & \text{if } e \geq e_{\text{entry}} \end{cases}$$

Embedded Clustering

By calculating the backward gradients of the 1D-CNN, each epoch should reduce the contrastive losses and therefore improve fault class distinctions. Clusters are initialized with k-means clustering on the encoded features, ensuring dense, spherical clusters, suitable for refinement. Based on the updated parameters of the 1D-CNN, these cluster soft-assignments are iteratively updated at each epoch, ensuring smooth clustering behavior and robust adaptation of cluster centers to the data distribution [185, 188, 235].

For a latent representation z_i and cluster centroid μ_k, cluster updating is guided by its soft assignment q_{ik}, which can be interpreted as the probability of assigning a sample i to cluster k. Soft assignments are calculated by a Student's t-distribution, and used to update centroids as follows:

$$q_{ik} = \frac{(1+||z_i-\mu_k||^2)^{-1}}{\sum_j (1+||z_i-\mu_k||^2)^{-1}}, \qquad \mu_k = \frac{\sum_i q_{ik} z_i}{\sum_i q_{ik}}$$

Once centroids are updated after the final training epoch, each sample is assigned to the cluster with the highest probability $c_i = \underset{k}{\text{argmax}}\ q_{ik}$, where c_i is the assigned cluster label for sample z_i.

Cluster Post-Processing

After DEC, the inferred clusters in the encoded space are derived from their initial k-means estimate. SCADA data can contain irregular, non-spherical clusters of an unknown number after training, so clusters should be recalculated using density-based clustering. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) has seen implementations for improving image and data-mining performance [236, 237]. Sporadic applications of WT SCADA clustering for power forecasting [238] and anomaly detection [218, 239]. For complex structures or varying densities, a hierarchical version of

DBSCAN (HDBSCAN) is applied to evaluate clusters at varying comparison levels.

3.4. NN for Fault classification

To learn the relationship between the extracted features of historic data and their failure modes, a Fully Connected Neural Network (FCNN) (Figure 2(d)) minimizes a Cross-Entropy loss function [174, 240]. This loss encourages confidence in correct answers, while implicitly penalizing mistakes. This loss expresses the error between the predicted labels $\hat{c}$ and learned clusters c as follows:

$$\mathcal{L}_{cl} = -\sum_{i=1}^{C} c_i \log \hat{c}_i$$

3.5. Transformer for RUL prediction

To capture long-term dependencies and inter-variable correlations in the attention mechanisms, the Transformer follows an encoder-decoder structure Figure 2(e). First, a linear embedding layer maps the input data to the model dimension. Then, after positional encoding, the embedded input data passes through a self-attention layer of multiple heads, where all the other input data points are taken into context [162, 163].

Before decoding and generating the output, an encoded version of the previously generated output is included as an additional input, enabling auto-regression. The decoder then follows the same operations as the encoder, followed by a single linear transformation to project the model calculation to the desired output.

Transformer Training

By analyzing measurements in the training dataset, the fault detection model provides end-of-life labels $\hat{T} < T$. To recognize the relationship between a given input sequence and matching RUL values, the difference between model output and target RUL values should be minimized. This difference is described by the Mean Squared Error (MSE), given in subsection 3.2.

To train auto-regressive behavior, the decoder is fed true RUL targets of the previous timestep through teacher forcing. Facing complex representations or imbalanced datasets, the model might become overdependent on the training targets and develop exposure bias. During training, the influence of the targets should be gradually reduced to increase reliability in the absence of input targets during testing [249].

4. Case Study Implementation

This work considers the Energia De Portugal (EDP) dataset, which provides detailed descriptions of maintenance actions, which are essential for model validation [261]. The dataset includes status and error codes, anemometer readings, and, most importantly, 81 SCADA sensor signals that provide data directly related to turbine performance and condition.

The dataset covers four 2 MW turbines, spanning two years. To maximize the use of available data and allow for comprehensive testing across all turbines, four different experimental configurations define training, validation, and testing sets, given in Table 1. For testing, the unseen dataset is split into segments corresponding to individual failure events marked by timestamps in the maintenance logs—overlapping or closely spaced failures grouped into a single segment, where final component replacement is often leading.

	Case 1	Case 2	Case 3	Case 4
Training Validation	T01, T06 T07	T06, T07 T11	T07, T11 T01	T01, T11 T06
Testing	T11	T01	T06	T07
Fail Modes	2	2	3	4
Fail Events	4	2	5	5

Table 1: Specified Test cases. Fail Modes and Fail Events refer to the number of fault types and occurrences for the testing subset.

The implementation of each model function is considered in its section. Hyperparameters that control the behavior of each model task are optimized per task, ensuring optimal performance and revealing parameter sensitivity. Optimization is performed by repeated retraining of the model based on Case 1, while aiming for a specified objective, using the optimization package 'Optuna' [263]. Detailed results of the optimization study are available in the full version of this work, including parameter search spaces, sensitivity, and discussion.

4.1. Preprocessing

A series of SCADA-specific preprocessing steps prepares the dataset, conditioning the time series for model interpretation. First, irrelevant measurements, such as minima, maxima, and averages of the same sensor, should be dropped, as they add noise, reducing prediction accuracy and increasing computational burden [134, 266, 267]. Variables with similar temporal effects are identified by a correlation analysis, where parameters with an absolute correlation larger than 95% are dropped, resulting in 31 remaining sensors [206].

Further conditioning involves handling duplicate measurements and linear interpolation of missing values. Normalization is applied as introduced in subsection 3.1, fitted on only training and validation data to avoid data-leakage. After processing, sequences can be generated using the sliding window

approach.

4.2. Anomaly Detection

Essential patterns and trends in the time-series data collected from OWT SCADA systems are captured through compressing and decompressing the input by an LSTM-AE.

LSTM-AE Hyperparameter Configuration

Optimal configuration of the LSTM-AE should correctly capture the most important features indicative of healthy operation, so that the RE closely matches developing defects during testing. Therefore, a parameter configuration is selected that offers the lowest validation loss, given by Table 2.

Parameter	Setting
Window Size	48
Step Size	12
Hidden Size 1	224
Hidden Size 2	224
Hidden Size 3	64
Latent Size	48
Number of Layers	1
Learning Rate	0.005
Number of Epochs	40

Table 2: LSTM-AE Hyperparameter settings

Larger autoencoder node sizes are selected as they yield better performance by reducing compression severity and preserving more information. Similarly, higher latent sizes improve stability by retaining more of the original feature space.

A single-layer architecture avoids overfitting issues and the overall intractability of the model. As a consequence, dropout is disabled, which is applied to connections between layers, instead of within the same layer, because this risks disrupting temporal dependencies.

A small window and step size are selected to increase stability and reduce computational time, while learning was best performed at a high rate.

By averaging Mean Squared Error (MSE) and Mahalanobis Distance (MD) expressions of the LSTM-AE reconstruction error, their varied responses to noise are canceled out. Smoothing with $\lambda = 0.0001$ then gives the HI, while the adaptive failure threshold is properly tuned by selecting window size $w = 3000$ and responsiveness to the standard deviation $k = 3.5$.

Fault Detection Results

After proper configuration of the model, Figure 3a demonstrates how system deterioration drives up the health indicator, causing a generated alarm. Figure 3b illustrates how varying health indicator trends affect the threshold, resulting in alarms at multiple timesteps that match transformer faults in the maintenance logs, as indicated by gray dashed lines.

For each case, frequently used HI performance metrics are calculated and given in Table 3, along with the share of detected faults. Anomalous behavior is detected thirteen out of sixteen times. Temperature errors and larger replacements score best, while hydraulic group errors, such as oil leakage and brake circuit errors, are sometimes missed if their influence on turbine operation is too subtle for capture.

Test Case	Mono.	Trend.	Prog.	Detected
Case 1	0.63	0.36	0.87	2/4
Case 2	0.77	0.86	0.94	2/2
Case 3	0.65	0.88	0.84	4/5
Case 4	0.66	0.83	0.82	5/5

Table 3: Monotonocity (Mono.), Trendability (Trend.), and Prognosability (Prog.) and share of detected faults

4.3. Fault Diagnosis

Due to the compression of the data, health-specific information is stored in the LSTM-AE latent space. Contrastive learning aims to amplify differences in failure behavior reflected in the data structure, which the classification model can then learn.

Only the sequences that contain alarms are considered to emphasize class distinctions. Also, features unrelated to faults should be removed, which can be identified by analyzing their sensitivity during failure regions.

1D-CNN Hyperparameter Configuration

Encoder settings, given in Table 4, are configured to maximize clustering accuracy. This involves properly assigning fault types according to the maintenance logs, while avoiding cluster collapse due to loss imbalance, poor initialization, or unstable training dynamics.

Clustering performance increases with network size. A kernel size of 1 was found to be optimal, as it projects features individually. Dilated convolutions allow the network to capture broader temporal dependencies, enhancing the modeling of both local and semi-global patterns.

The extensive collection of generated input sequences is downsampled by introducing a convolution stride, reducing temporal resolution to improve cluster distinction. Padding is dynamically set to half the kernel size to preserve sequence length.

Parameter	Setting
CNN Layers	6
Hidden Size	184
Latent Size	16
Max Dilation	4
Convolution Kernel Size	1
Pooling Kernel Size	1
Convolution Stride	4
Loss Temperature	4
Loss Margin	9
Window Size	6
Step Size	1
Number of Epochs	70
Learning Rate	0.0005
Gamma (lr decay)	0.85

Table 4: 1D-CNN Hyperparameter settings

During training, high values for the temperature and margin parameters of the sequence-based and pairwise contrastive losses ensure significant transformations of the data structure, ensuring increased separation of dissimilar pairs.

After encoding, cluster assignments are refined using HDBSCAN at a distance value of 12. Training setting results vary since various combinations and timings of loss functions can influence clustering quality. Robust is slow training for 70 epochs, while starting the shift towards local contrastive loss at epoch 62, with a shift rate of 15 epochs.

Based on the encoded input, repeated training of the FCNN used for classification suggests the optimal balance between interpretation capacity and generalizability is obtained with a single layer of 164 nodes.

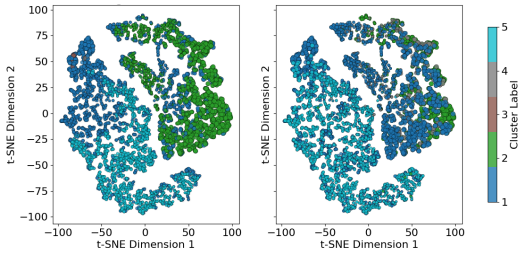
Clustering Results

Using t-SNE dimensionality reduction, Figure 4 demonstrates how samples, marked by their true fault category according to the failure logs, are moved during training epochs to suit the training loss functions. Samples that belong to the different failure modes can be observed to form increasingly well-defined groups over time, first in a global aspect, and then refined locally in the last epoch.

By comparing the inferred label of each sequence to the assumed ground truth, model accuracy is given for supervised and unsupervised contrastive learning in Table 5. Formed clusters, with or without label availability, are visualized in Figure 5a. Inferred or ground-truth clusters marked by color, showing the transformer fault is best defined, likely as its influence on SCADA sensors varies compared to drivetrain components. The generator bearing fault is obscured mainly by the generator and gearbox clusters.

	Clustering Accuracy	
Case	Unsupervised	Supervised
Case 1	55-61%	92-97%
Case 2	56%	97.9%
Case 3	60.5%	93.7%
Case 4	68.3%	90%

Table 5: Clustering accuracy, for supervised or unsupervised approaches to each case. The accuracy of case 1 gives the range following n=5 tests.

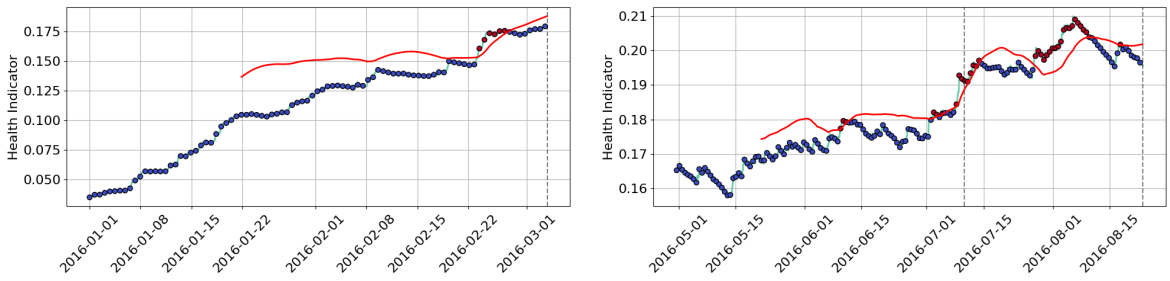


(a) Fully unsupervised clustering results

Figure 5: t-SNE visualization of inferred cluster labels vs. their true label. 1 = Gear., 2 = Gen., 3 = Gen. Bear., 4 = Hydr., 5 = Trans.

Classification Results

The trained encoder compresses unseen testing data before the classification model classifies it. To verify improved accuracy of fault classification based



(a) T11 Generator Fault

(b) T07 Transformer Fault

Figure 3: Health-Indicator and generated alarms (red markers), through adaptive fault threshold (red line), compared to maintenance action (Gray)

on learned contrastive embeddings, the accuracy of classification is given in Table 6 for both supervised and unsupervised clustering cases. Additionally, the table includes the accuracy of performing classification directly on the LSTM-AE output, bypassing the 1D-CNN.

	Classification Accuracy		
Case	Uns. DEC	Sup. DEC	No DEC
Case 1	0-40%	30-42%	10-25%
Case 2	39%	48%	31 %
Case 3	40%	46%	33%
Case 4	33%	36%	31%

Table 6: Classification accuracy following unsupervised DEC (Uns.), Supervised (Sup.), or without DEC (No DEC). The accuracy of case 1 gives the range following n=5 tests.

4.4. Fault Prognosis

RUL prediction involves learning the relationship between the LSTM-AE output and the corresponding time to failure. RUL labels derived from historically detected faults are capped and normalized to stabilize training [75, 142, 250, 251].

Transformer Hyperparameter Configuration

Correctly specifying Transformer parameters involves minimizing the validation loss, ensuring relevant features can be extracted and translated to RUL values, while ensuring it generalizes to unseen inputs. Short-lived fault features in SCADA data quickly infer overfitting and exposure bias, limiting the size of the optimal transformer architecture, defined by hyperparameters given in Table 7.

The model width is comparable to the number of input features, indicating the LSTM-AE signal is already informative and does not benefit from large transformations. Three encoder layers ensure the model is capable of interpreting data relationships without over-complicating training, while two decoder layers perform the simpler task of generating the RUL signal. The optimal feed-forward layer size of 24 further reflects the task's low nonlinearity and suggests that the attention layers extract clear, well-defined end-of-life indicators.

The narrow transformer requires precise training to balance convergence and generalization. Dropout randomly masks attention weights of the self-attention mechanism, or neurons in the fully connected layers, preventing overfitting. Larger windows improve the model's ability to learn temporal dependencies, while smaller step sizes increase overlap and prediction confidence. Both benefits are balanced against computational limitations, however.

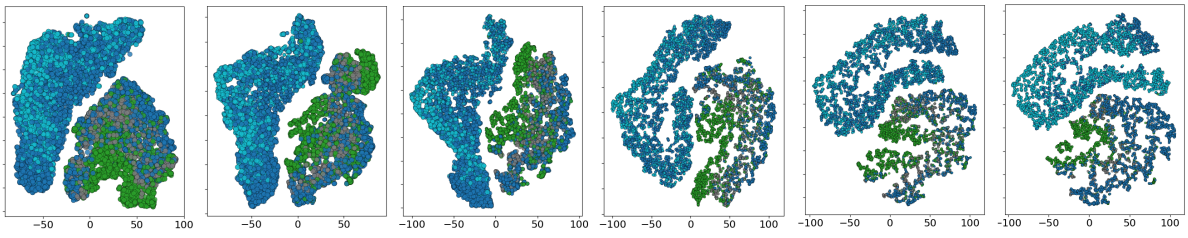
Parameter	Setting
Number of Encoder Layers	3
Number of Decoder Layers	2
Model Dimension	32
Number of Heads	8
Feedforward NN Dim	24
Window Size	136
Step Size	40
Number of Epochs	150
Learning Rate	2e-4
Gamma (lr decay)	0.8
Gamma Stop	5
Dropout	0.25

Table 7: Transformer Hyperparameter settings

RUL Prediction Results

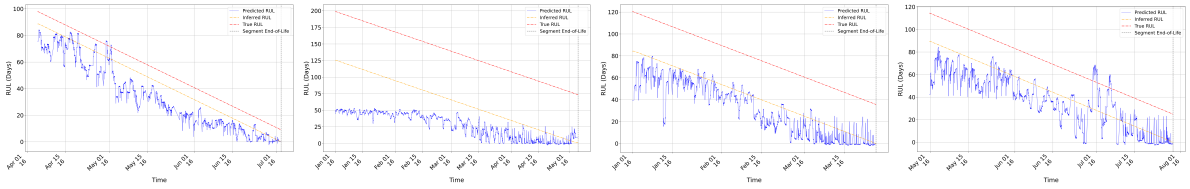
Figure 7 visualizes predicted RUL—expressed in remaining operational days—for faults in the gearbox, generator, generator bearing, and transformer. A detailed example of the full RUL prediction pipeline is shown in Figure 8, highlighting input signals, reconstruction-error-based health estimation, fault detection, and RUL prediction for a hydraulic system failure.

Prediction accuracy is quantified using RMSE across timesteps per case and given in Table 8. It shows Hydraulic- and generator faults yield the most consistent predictions due to the higher number of



(a) Epoch = 1 (b) Epoch = 10 (c) Epoch = 20 (d) Epoch = 40 (e) Epoch = 60 (f) Epoch = 70

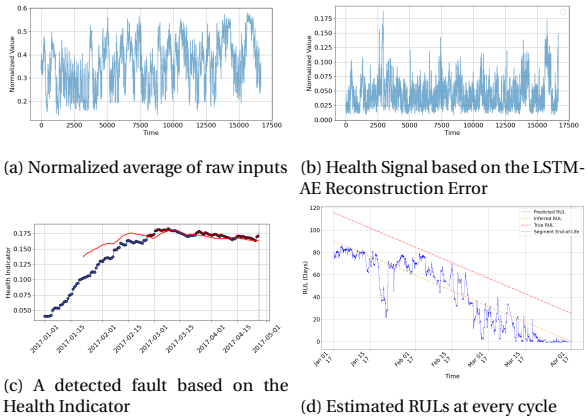
Figure 4: t-SNE visualization of 1D-CNN latent space during training of Case 1, using $e_{\text{entry}} = 60$. Blue = Gearbox, Green = Generator, Gray = Hydraulics, Cyan = Transformer



(a) T06 Generator Fault (b) T01 Gearbox Fault (c) T07 Generator Bearing Fault (d) T07 Transformer Fault

Figure 7: RUL prediction (Blue) based on the LSTM-AE inferred RUL (orange), compared to true RUL (red)

fault examples for these components.



(a) Normalized average of raw inputs (b) Health Signal based on the LSTM-AE Reconstruction Error

(c) A detected fault based on the Health Indicator (d) Estimated RULs at every cycle

Figure 8: Fault detection and an RUL estimate of T11 Hydraulic Fault

	Unsupervised	Supervised
Case 1	22.31	38.94
Case 2	35.01	44.57
Case 3	28.44	37.64
Case 4	26.63	22.84

Table 8: RUL prediction RMSE in days, for RUL prediction based on LSTM-AE Latent Space data.

5. Conclusion

An integrated framework of deep learning methods enables the unsupervised interpretation of Offshore Wind Turbine (OWT) SCADA data for anomaly detection, fault diagnosis, and Remaining Useful Life (RUL) estimation.

To validate the framework under controlled conditions, a simulated case engine dataset is studied. These results indicate the approach provides a competitive and sufficiently reliable means of data-based health monitoring, achieving performance levels comparable to an extensive collection of similar studies.

Application to real-world OWT SCADA data demonstrates the framework's feasibility in practical settings. Compared to the simulated case, SCADA-based challenges introduced increasing data imbalance and less clearly defined features, complicating the training of each model and impacting reliability. This results in decreased reliability of the health indicator, reduced responsiveness of the fault detection, and obscuring of failure-specific features during early stages of fault development, which reduces the accuracy of diagnosis and RUL prediction.

By assessing these limitations, several model improvements and directions for future work are proposed. First, the Autoencoder (AE) could benefit from incorporating Attention Mechanisms, as suggested in recent literature, to better focus on relevant input regions and mitigate the effects of data imbalance. Additionally, intelligent compression of the input could improve anomaly detection by emphasizing key patterns.

To enhance fault diagnosis, the DEC model should account for temporal context. This would allow for distinguishing between early and late fault stages and increase sensitivity to subtle degradation. Further improvement of the diagnosis involves increasing the responsiveness of the classification model to spatial features.

While improvements to the reconstruction error signal and fault detection are likely to have the most significant impact on RUL prediction, further steps such as post-processing and uncertainty quantification are recommended to ensure the stability and real-world viability of the RUL output.

Finally, future work should focus on validating the framework across additional case studies to ensure generalizability and reduce the risk of confirmation bias introduced during post-processing.

Despite these challenges, this study shows that deep learning methods can extract meaningful health information from SCADA data, supporting the predictive optimization of maintenance.

B

C-MAPSS Case Study - Supplementary Material

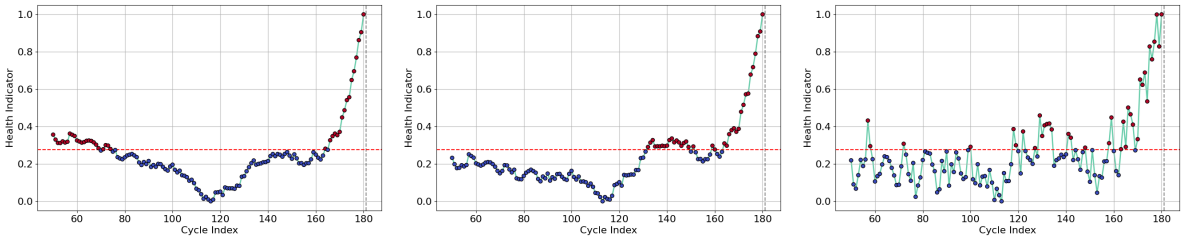
B.1. C-MAPSS Computational Configuration

The proposed machine learning architecture is implemented in Python's PyTorch library. This package provides neural network building blocks, including layers and activation functions, whose dimensions and behavior are specified by user-specified hyperparameters. Network operation is defined by writing a 'forward' function, specifying the flow of information. During training, backward inference is automatically calculated based on this forward structure.

Adaptive moment estimation method (Adam) is one of the most popular optimizers offering high computational efficiency and requiring little tuning (Liu et al., 2021 [271]). When mentioned, learning is supported by an exponential learning decay scheduler, improving the fine-tuning of the model. When computationally possible, model operation runs with a batch size of 256. Construction is performed in Pytorch (2.5.0), in the environment Python (3.13). The computer configuration for training was Intel Core i7-8750H (CPU) @ 2.20GHz, 16 GB 2666 MHz DDR4 (RAM), NVIDIA GeForce GTX 1050 Ti (GPU).

B.2. Elaboration on the selection of EWMA smoothing factor

Figure B.1 demonstrates the effect of different settings of the parameter λ, which influences the weight of newly added values to the weighted moving average, where a higher weight means new values are more influential and the curve responds more quickly. HI performance metrics based on different values of λ are given in Table B.1. While a low weight results in a smoother curve, and thus higher monotonicity, the responsiveness is too low to properly reflect the current health state, and the effect of irrelevant early life deviations is stored in the average for too long, at the cost of trendability.



(a) $\lambda = 0.005$ (b) $\lambda = 0.05$ (c) $\lambda = 0.5$

Figure B.1: Comparison of the effect of different values of λ on engine #35

$\lambda =$	Monotonicity	Trendability	Prognosability
0.005	0.4	0.64	0.88
0.05	0.33	0.94	0.9
0.5	0.08	0.83	0.95

Table B.1: Health Indicator performance metrics based on different values of λ

B.3. Additional Implementation Figures

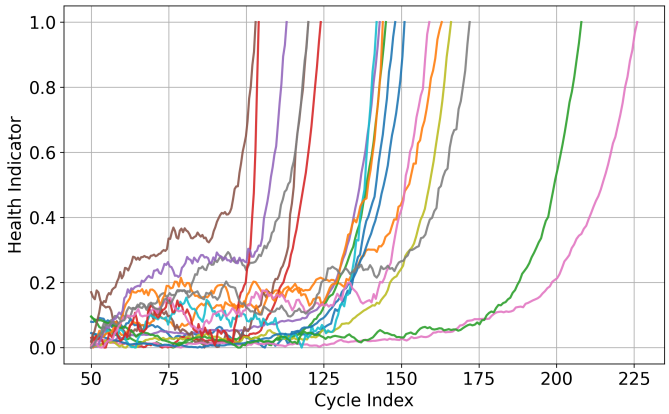
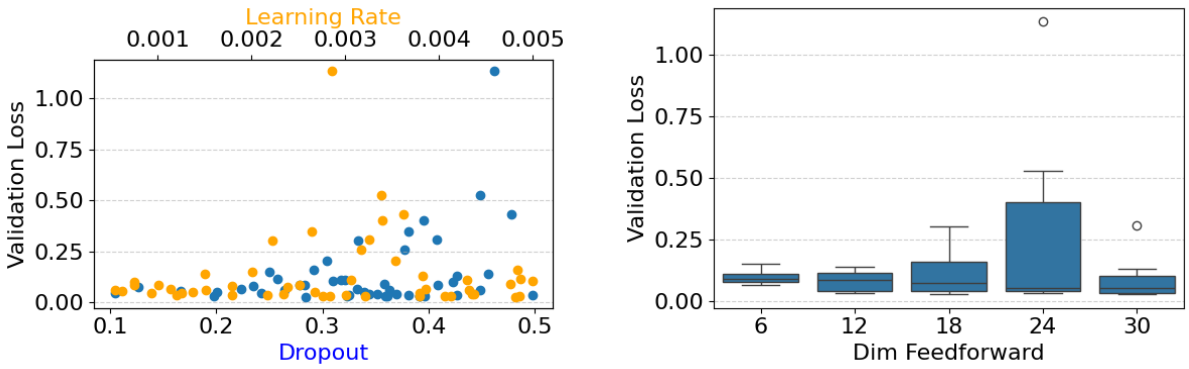


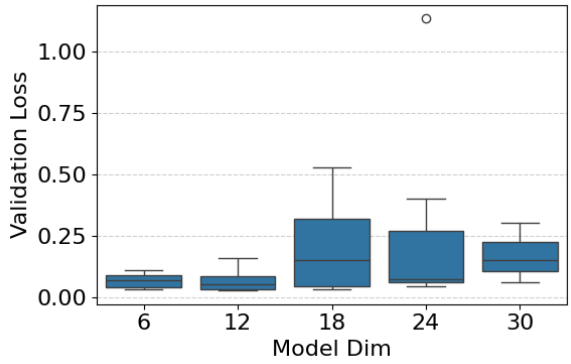
Figure B.2: Multiple constructed health indicators of a randomized test configuration, normalized for generating this plot.

B.4. Transformer hyperparameter sensitivity

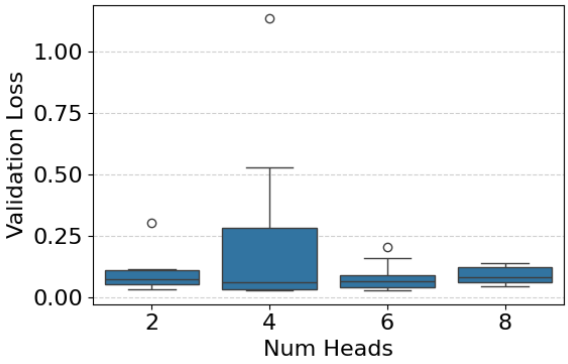


(a) Neuron Dropout

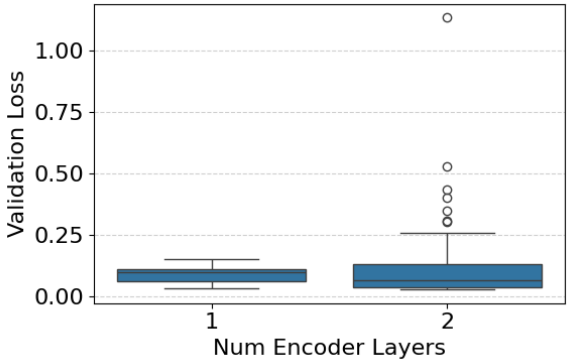
(b) Feed Forward NN Layer Width in Neurons



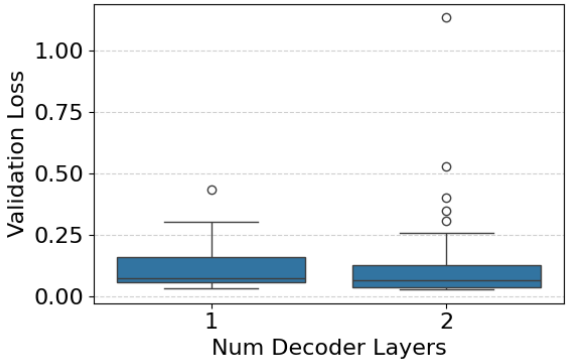
(c) Model Width in Neurons



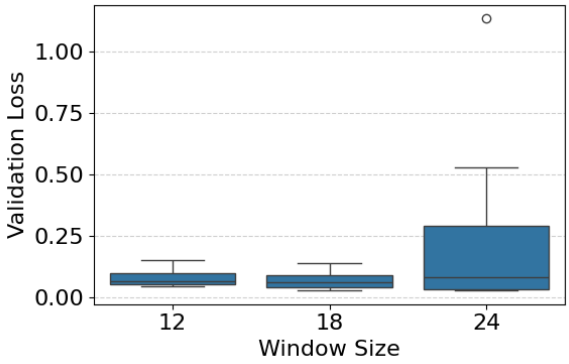
(d) Number of attention heads



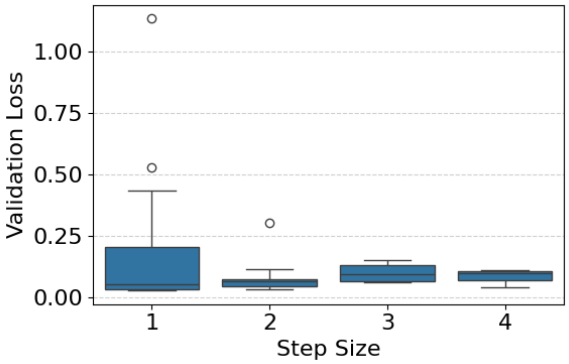
(e) Number of Encoder Layers



(f) Number of Encoder Layers



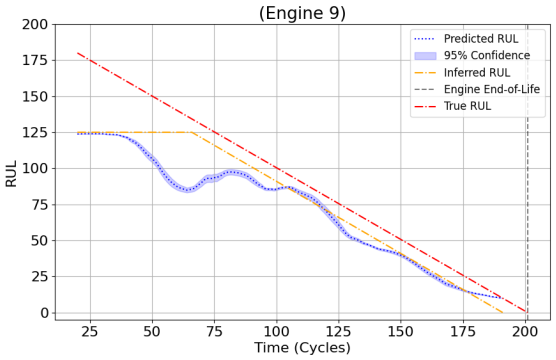
(g) Sliding Window Size



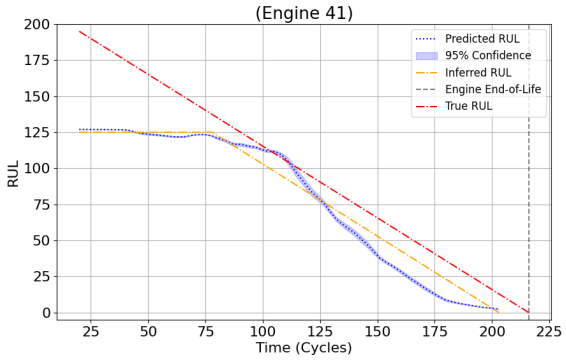
(h) Sliding Window Step Size

Figure B.3: Box Plots for transformer sensitivity after 50 optima parameter optimization trials, showing the Validation Loss (MSE) per hyperparameter.

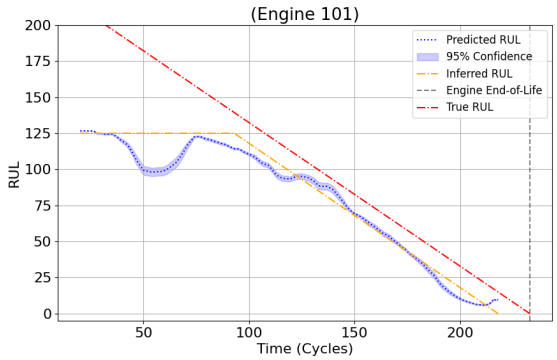
B.5. Additional C-Mapss RUL figures



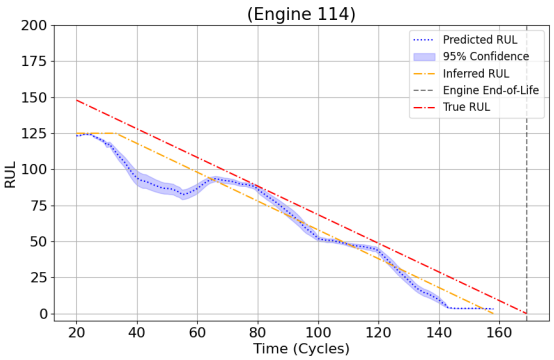
(a) Engine #9



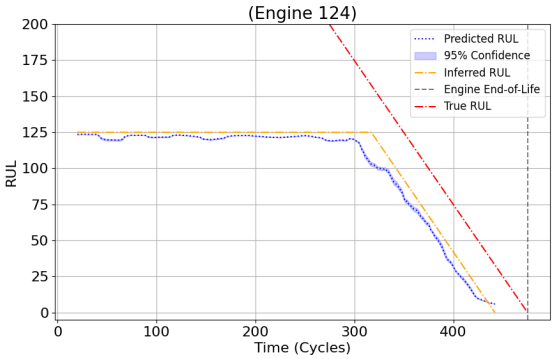
(b) Engine #41



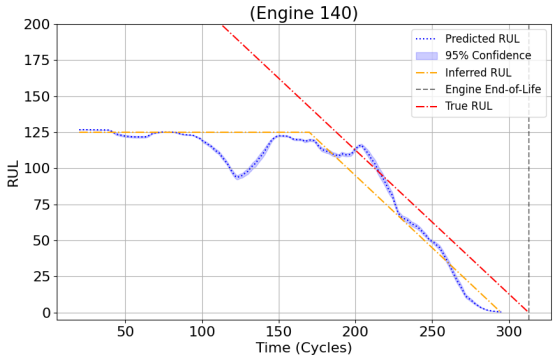
(c) Engine #101



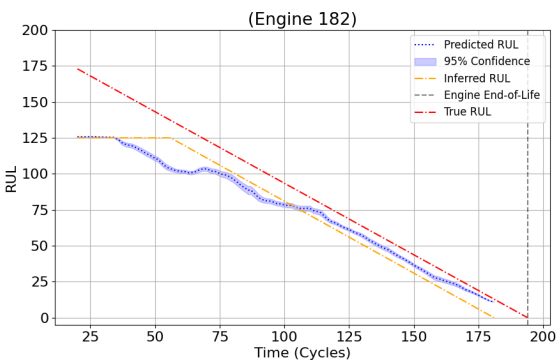
(d) Engine #114



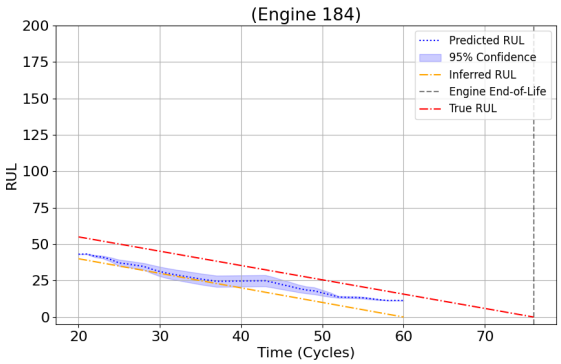
(e) Engine #124



(f) Engine #140



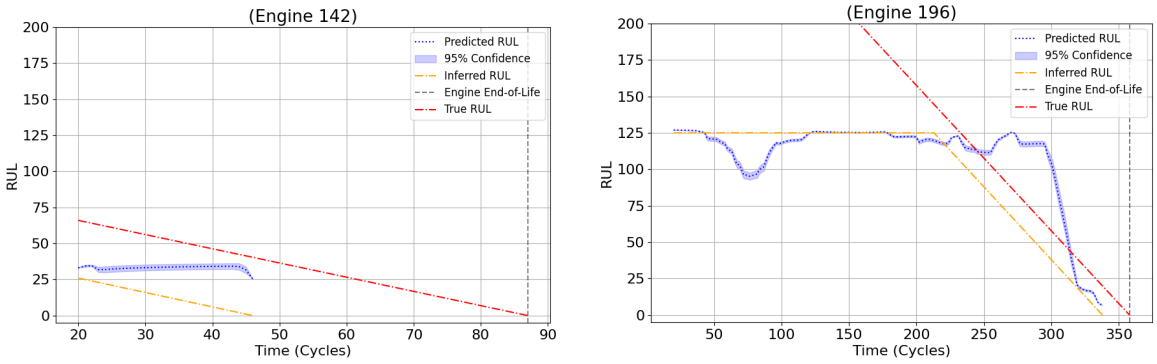
(g) Engine #182



(h) Engine #184

Figure B.4: Figures depicted the predicted RUL values of eight random testing engines with 95% confidence, their targets, the engine's true RUL, and the engine's eol.

B.6. Greatest C-Mapss RUL Error Contributors



(a) Engine #142

(b) Engine #196

Figure B.5: Predicted RUL values of two engines that contributed the most out of the testing set, with 95% confidence, targets, the engine's true RUL, and the engine's eol.

C

EDP Case Study - Supplementary Material

C.1. Overview of Available SCADA Datasets

Overview of available SCADA datasets for investigating data challenges and selecting a suitable case for study.

[59], [272], but most recently [273] and [206], include reviews or overviews on available open-source SCADA datasets. A short overview is given in Table C.1, showing some characteristics of currently accessible datasets relevant for WT CM.

Datasets	Number of WT	Type	Time span	Reported information	# of Parameters
Beberibe [274]	32	Onshore	1 year	SCADA	43
EDP [261]	4	Offshore	2 years	SCADA, metmast, (failure) logs	121
DSWE I1 [275]	4	Onshore	1 year	SCADA (Wind), metmast	7
DSWE I2 [275]	2	Onshore	1 year	SCADA (Wind), metmast	8
DSWE O1 [275]	2	Offshore	1 year	SCADA (Wind), metmast	9
DSWE O2 [275]	2	Offshore	1 year	SCADA (Wind), metmast	8
PCWG [276]	3	-	8, 6, 2 months	SCADA (Wind) metmast	17
Kelmarsh [277]	6	Onshore	5 years	Static, SCADA, metmast, logs	303
Penmanshiel [278]	14	Onshore	5 years	Static, SCADA, metmast, logs	303
Ørsted [279]	Unknown	Offshore	2 years	SCADA	5
USP [280]	1	Onshore	5 years	SCADA	40
Vestas [281]	1	Onshore	14 years	SCADA	22
Yakova [282]	1	Onshore	1 year	Wind-, power parameters	4

Table C.1: Available open-source SCADA datasets. Sampling rate of 10 min. SCADA data: i.e., temperatures, power parameters, and speeds. Metmast: meteorological tower measured Wind Speed, Direction, and other ambient measurements. Logs: status descriptions, such as component activations and failure descriptions. Static: Hub Height, Rated Power, etc.

C.2. Wind turbine failures of EDP dataset

Failure logs as provided by the dataset.

Turbine ID	Component	Timestamp	Remarks
T01	Gearbox	18.07.2016	Gearbox pump damaged
T06	Generator	11.07.2016	Generator replaced
T06	Generator	24.07.2016	Generator temperature sensor failure
T06	Generator	04.09.2016	High temperature generator error
T06	Generator	27.10.2016	Generator replaced
T06	Generator	02.10.2016	Refrigeration and temp. sensors in the generator replaced
T06	Hydraulics	04.04.2016	Error in pitch regulation
T07	Generator bearing	30.04.2016	High temperature in generator bearing (replaced sensor)
T07	Transformer	10.07.2016	High temperature transformer
T07	Transformer	23.08.2016	High temperature transformer. Transformer refrigeration repaired
T11	Generator	03.03.2016	Electric circuit error in generator
T11	Hydraulics	17.10.2016	Hydraulic group error in the brake circuit

Table C.2: EDP wind turbine failures 2016

Turbine ID	Component	Timestamp	Remarks
T01	Gearbox	18.7.2016	Gearbox pump damaged
T01	Transformer	11.08.2017	Transformer fan damaged
T06	Gearbox	17.10.2017	Gearbox bearings damaged
T06	Hydraulics	19.08.2017	Oil leakage in Hub
T07	Generator bearing	20.08.2017	Generator bearings damaged
T07	Generator	21.08.2017	Generator damaged
T07	Hydraulics	17.06.2017	Oil leakage in Hub
T07	Hydraulics	19.10.2017	Oil leakage in Hub
T11	Hydraulics	26.04.2017	Brake circuit error
T11	Hydraulics	12.09.2017	Brake circuit error

Table C.3: EDP wind turbine failures 2017

C.3. List of all studied hyperparameters

Model	Parameter	Description	Research Scope
LSTM-AE	Window Size	Length of input sequence	16–128 (step=16)
	Step Size	Step size of sliding window	2–24 (step=2)
	Window Size	Size of the sequence window input	12–24 (step=6)
	Step Size	Step between sequence windows	1–4 (step=1)
	Size 1	Dimensionality of first LSTM layer	32–256 (step=32)
	Size 2	Dimensionality of second LSTM layer	32–256 (step=32)
	Size 3	Dimensionality of third LSTM layer	16–128 (step=16)
	Latent Size	Bottleneck dimension of encoder output	8–64 (step=8)
	Num Layers	Number of stacked LSTM layers	1–5
	Window Size	Size of the sequence window input	12–24 (step=6)
	Step Size	Step between sequence windows	1–4 (step=1)
	Learning Rate	Effect backpropagation has on Nodes with the optimizer	5e−4 to 1e−2 (log scale)
	Epochs	Number of training epochs	20–160 (step=20)
	Dropout	Probability that individual nodes might be excluded	0.1–0.5 (step=0.1)
	Gamma	Learning rate decay factor ($lr = lr * e^{-(\text{gamma} * \text{epoch})}$)	0.8–0.99 (log scale)
	Gamma Stop	Epoch after which learning rate decay stops	1–101 (step=20)
1D CNN	Window Size	Size of the sliding window	6–126 (step=12)
	Step Size	Stride of the sliding window	1–24
	Conv Kernel Size	Size of the convolutional kernel	1 - min(12, window_size/4)
	Stride	Step size of the convolution operation	1–4
	Pool Kernel	Size of pooling kernel (MaxPooling)	1, 2
	Max Dilation	Max dilation rate for dilated convolutions	1, 2, 4, 8, 16
	Number of Layers	Number of convolutional layers (combined Conv+Pooling)	1–6
	Hidden Size	Size of the intermediate dense layer after convolutions	24–280 (step=32)
	Latent Dim	Size of the latent feature space	8–64 (step=8)
	Epochs	Number of training epochs	5–155 (step=25)
	Learning Rate	Learning rate for optimizer	1e−5 to 1e−3 (log scale)
	Gamma	Exponential decay factor for learning rate	0.7–0.99
	Stop	Epoch after which LR decay is halted	1–26 (step=5)
	Drop Threshold	Minimum feature sensitivity threshold	-0.3 to 0.5 (step=0.1)
	e Entry	Epoch after which second loss function starts (seq or pair loss)	5–155 (step=15)
	Seq Loss Ratio	Weight factor of the Seq Loss as it is often higher than the pair loss	0.1–1.0 (step=0.1)
	New Clusters Ratio	Weight of new cluster assignments after a training epoch	0.1–1.0 (step=0.1)
	Loss Grad	Epochs needed to shift from one loss to another after e_{entry} calculate loss gradient	5–25 (step=5)
	Margin	Margin parameter used in contrastive loss	0.1–10.0
	Temperature	Temperature scaling for sequence loss	0.1–8.0
	Drop Healthy	Whether to exclude healthy sequences during clustering	True, False
	First Global	Whether to apply global- (Seq) before local (Pair) refinement	True, False
	Adaptive Margin	Whether the adaptive margin used in the pair loss during training	True, False
	DB Scaler	Whether to scale DBSCAN input distances	True, False
	DB Type	Type of density-based clustering used (DBSCAN or HDBSCAN)	`db`, `hdb`
	DB Dist	Distance threshold for DBSCAN	2–30 (step=2)
	Eps Percentile	Percentile for selecting neighborhood distance for calculating DBSCAN eps	5–95 (step=10)
Transformer	Window Size	Size of the sequence window input	50–140 (step=30)
	Step Size	Step between sequence windows	40–64 (step=8)
	Num Heads	Number of attention heads in multi-head attention	2–16 (step=2)
	Model Dim	Width of the transformer layers & Embedding dimension (adjusted to multiple of num_heads)	16–96 (step=8)
	Encoder Layers	Number of Transformer encoder layers	1–4
	Decoder Layers	Number of Transformer decoder layers	1–4
	Feedforward Dim	Size of the feedforward network within each Transformer block	16–64 (step=8)
	Learning Rate	Learning rate for optimizer	5e−4 to 5e−3 (log scale)
	Epochs	Number of training epochs	125–250 (step=25)
	Dropout	Dropout rate applied to Transformer layers	0.01–0.5
	Gamma	Learning rate decay factor	0.8–0.99 (log scale)
	Gamma Stop	Epoch after which learning rate decay stops	5–40 (step=5)
	Max Rul	RUL value used for label clipping	14e3–20e3 (Step=1e3)

Table C.4: Hyperparameter tuning space across different model architectures.

C.4. Delft Blue Workflow

This section outlines some of the steps taken to set up Delft Blue and run the files of this thesis. For a more extensive outline, the reader is referred to the Delft Blue Documentation [283].

C.4.1. Delft Blue Access

With a direct or VPN connection to the TU Delft network, Delft Blue Access requires logging in with a user account via SSH:

```
ssh jhes@login.delftblue.tudelft.nl
```

C.4.2. Setting Up the Environment.

Since DelftBlue uses module-based package management, load the required modules before running scripts: Set up / load modules:

```
module load slurm
module load 2024r1
module load cuda/12.4 # 12.4 has compatibility with PyTorch
module load miniconda3 # For setting up conda environment
```

C.4.3. Creating a Conda Environment

Create Anaconda environment Set up PyTorch and Optuna in the accessible Anaconda environment. With a compatible Python version:

```
conda create –p /scratch/jhes/torch_env python=3.10
conda activate /scratch/jhes/torch_env
pip install pytorch   # Pip is recommended to avoid compatibility issues
pip install optuna
conda install pandas numpy # Ensure compatible versions
```

C.4.4. Job Submission

Upload the required files and create a SLURM batch script (.sh file) for submitting jobs. This file defines the task and required resources, which are activated in the cmd via:

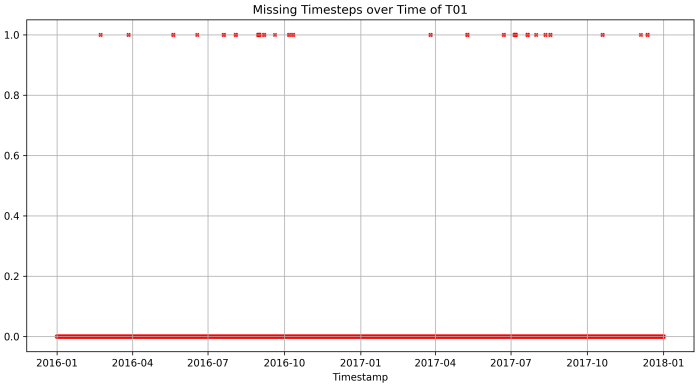
```
sbatch run_script.sh
```

Below is an example Slurm script:

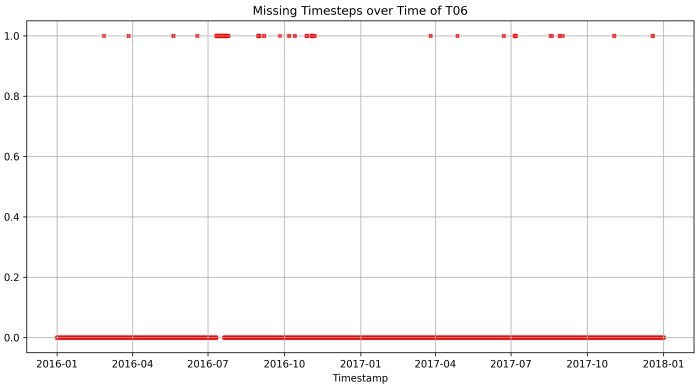
```
#!/bin/sh
#SBATCH --job–name=example_test
#SBATCH --output=test_output.txt
#SBATCH --error=test_error.txt
#SBATCH --partition=gpu
#SBATCH --account=education–me–msc–me
#SBATCH --time=02:00:00
#SBATCH --ntasks=1
#SBATCH --cpus–per–task=1
#SBATCH --gpus–per–task=1
#SBATCH --mem–per–cpu=1GB

module load 2024r1
module load cuda/12.5
module load miniconda3

conda activate /scratch/jhes/my_env
srun python DelftBlue_example.py --optuna_trials 50 > pi.log
conda deactivate
```

C.5. EDP Missing Values

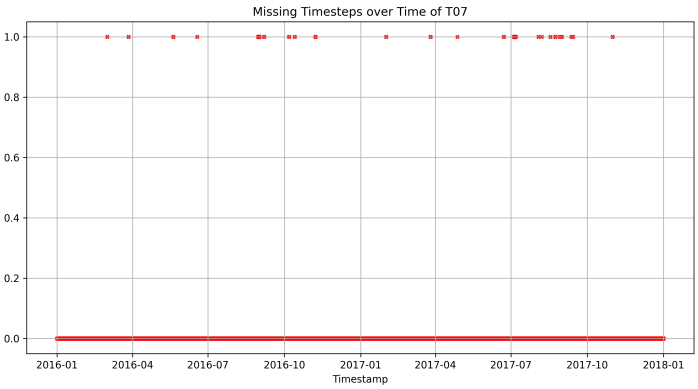
Occurrences of missing values in each turbine dataset.



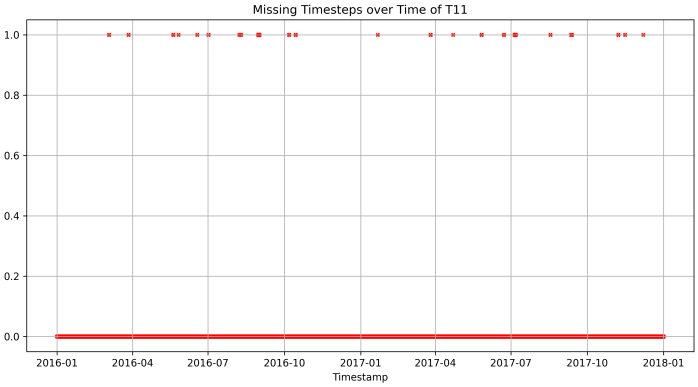
(a) T01



(b) T06



(c) T07



(d) T11

Figure C.1: Missing timestamps of EDP dataset, where 1 indicates a missing row and 0 indicates a present row

C.6. Preprocessed EDP SCADA Inputs

This section depicts time series and power curves of the selected 31 sensors after preprocessing.

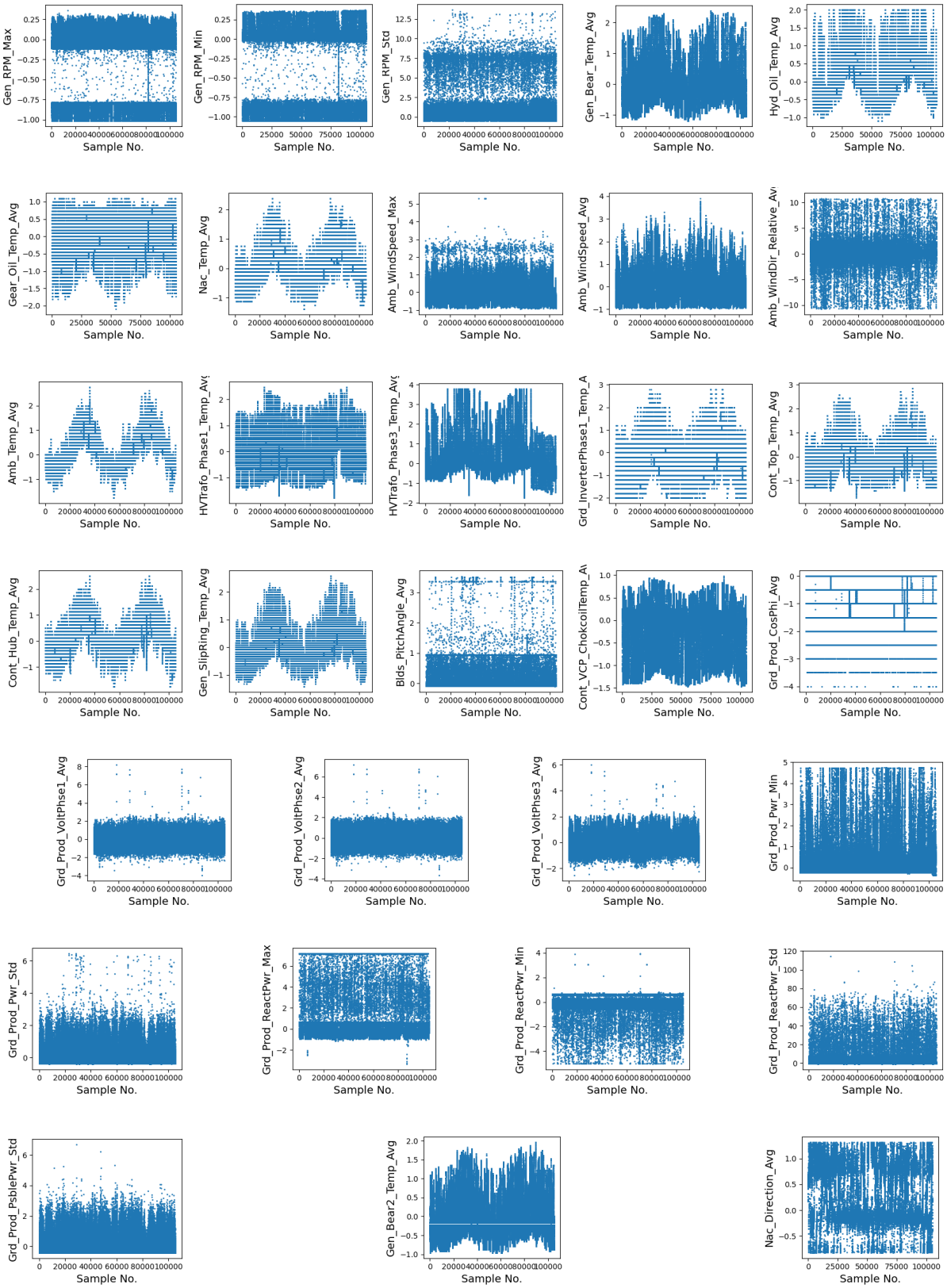


Figure C.2: Time-series of the 31 normalized & conditioned input features, at a measurement frequency of 10 minutes

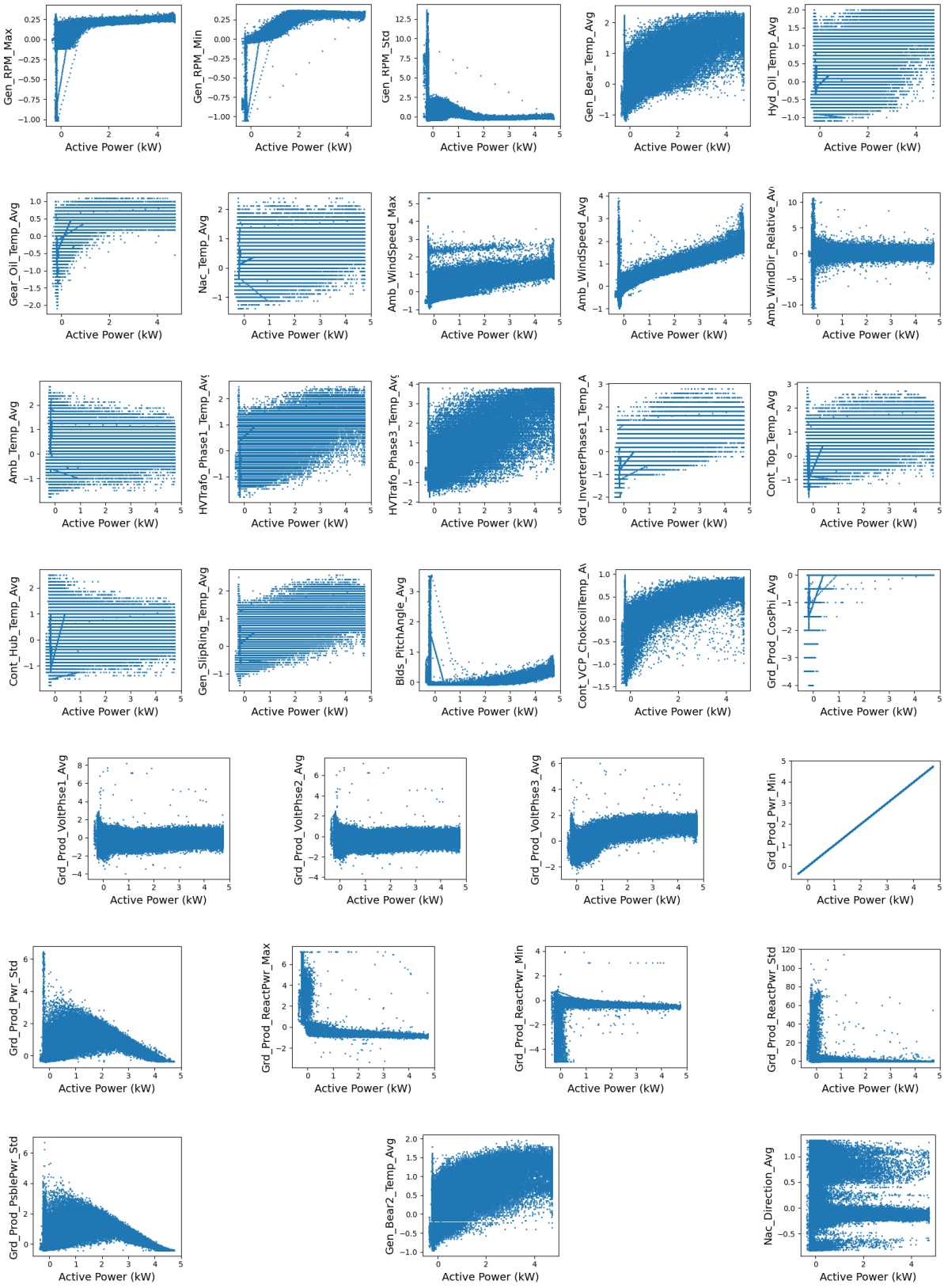


Figure C.3: Power curves plotting Average active power against the 31 normalized & conditioned input features, at a measurement frequency of 10 minutes

C.7. Study of Optuna Robustness

Ideally, a robust optimization model would yield the same outcome for different study cases and when applied to the same case repeatedly. To verify the model's robustness in this manner, the proposed method for hyperparameter optimization is applied to each study case (Table 5.1) and an additional time to cases 1 and 2. The results are outlined in Table C.5 below.

It can be observed that increasing the number of trials improves the model outcome. This is only a relatively small increase, however, so a further increase of the number of trials (and therefore a drastic increase of the required computational time) is not considered. When reviewing the optimal hyperparameters, some similarities can be seen, but also a lot of variations. To assess the differences and similarities across studies, case 1 (R) and case 2 (R) are compared in Figure C.4.

	Trials	Pruned Trials	Optimal Val. Loss	Proposed Hyperparameters
Case 1	50	27	3.305	*size_1*: 128, *size_2*: 128, *size_3*: 32, *latent_size*: 40, *num_layers*: 1, *lr*: 0.0014, *num_epochs*: 140, *window_size*: 80, *step_size* : 20, *dropout*: 0.30, *gamma*: 0.822, *gamma_stop* : 1
Case 1 (R)	100	52	3.24	*size_1*: 256, *size_2*: 224, *size_3*: 128, *latent_size*: 64, *num_layers*: 1, *lr*: 0.001437707373457665,*num_epochs*: 140, *window_size*: 128, *step_size*: 20, *dropout*: 0.4, *gamma*: 0.9326386759583164, *gamma_stop*: 1
Case 2	50	39	3.052	*size_1*: 224, *size_2*: 256, *size_3*: 48, *latent_size* : 64, *num_layers*: 1, *lr*: 0.005, *num_epochs*: 120, *window_size*: 48, *step_size* : 6, *dropout*: 0.4, *gamma*: 0.912, *gamma_stop*: 1
Case 2 (R)	100	54	2.985	*size_1*: 128, *size_2*: 96, *size_3*: 64, *latent_size* : 40, *num_layers*: 1, *lr*: 0.002, *num_epochs*: 100, *window_size*: 96, *step_size* : 18, *dropout*: 0.2, *gamma*: 0.919, *gamma_stop*: 1
Case 3	50	10	3.581	*size_1*: 192, *size_2*: 224, *size_3*: 96, *latent_size*: 56, *num_layers*: 1, *lr*: 0.0057, *num_epochs*: 160, *window_size*: 16, *step_size*: 2, *dropout*: 0.30, *gamma*: 0.946, *gamma_stop* : 81
Case 4	50	42	3.607	*size_1*: 256, *size_2*: 192, *size_3* : 128, *latent_size*: 8, *num_layers*: 1, *lr*: 0.0015, *num_epochs*: 120, *window_size*: 16, *step_size*: 2, *dropout*: 0.1, *gamma*: 0.90, *gamma_stop*: 1

Table C.5: Repeated optimization studies for each study case. Case 1 is run a second time for double the amount of time.

When comparing the two cases, several key differences emerge regarding the impact of hyperparameters:

- **Validation Loss**: Case 1 exhibits a higher validation loss overall, likely due to a greater presence of failures in the validation dataset.
- **Hidden Node Size**: Case 2 requires a larger number of hidden nodes to effectively capture the increased number of failure modes and occurrences in the training set.
- **Latent Size**: Higher latent sizes improve stability in both cases by reducing compression severity and preserving more original feature information.
- **Number of Layers**: Optimal validation loss is achieved with a single-layer architecture.
- Both cases their optimal Learning Rate: Both cases converge best with a learning rate between 0.002 and 0.005, though this varies based on other factors such as the number of epochs and learning rate decay settings.
- **Epochs**: Determining an optimal number of epochs is challenging, as additional epochs beyond convergence do not impact validation loss but influence the variability seen in the results. However, a minimum of 100 epochs is identified as sufficient for both cases.

- **Window Size**: Similar for both cases, with not much variation, with the best validation loss stability occurring at 128.
- **Step Size**: Optimal for both cases, around 16-20.
- **Dropout**: Since most trials were conducted with a single layer, PyTorch ignores dropout in such configurations, making dropout settings irrelevant in this context.
- **Learning Rate Decay**: The best performance in both cases is achieved by setting the stopping epoch for LR decay at the first epoch, effectively disabling learning rate decay.

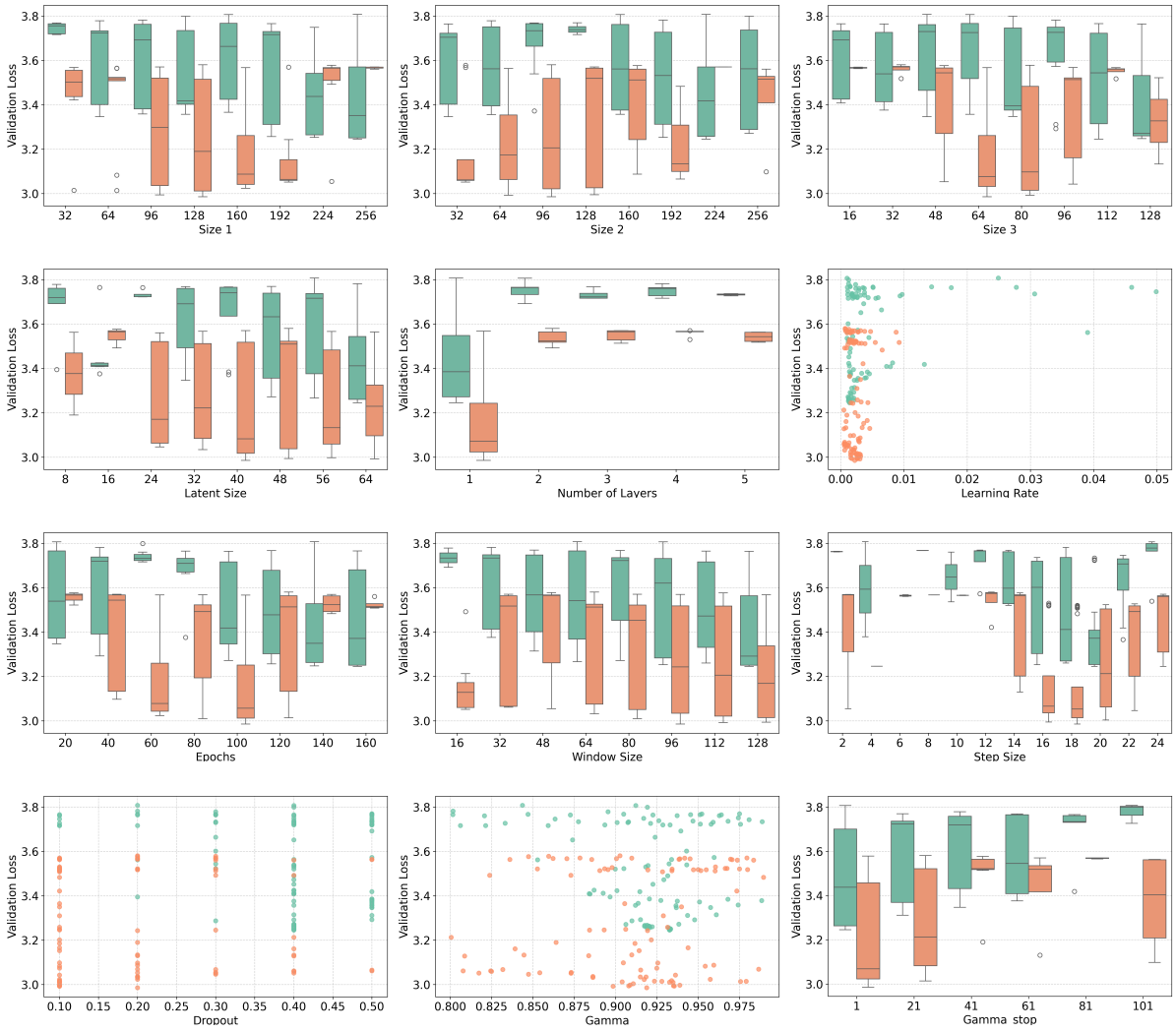


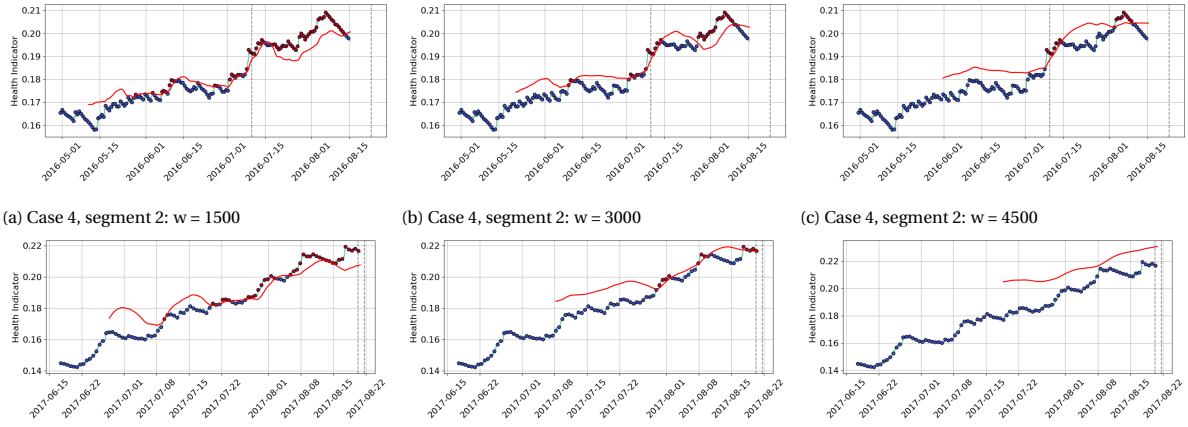
Figure C.4: Box Plots for LSTM-AE sensitivity after 100 optima parameter optimization trials for Case 1 (Blue) and 2 (Orange), showing the Validation Loss (MSE) per hyperparameter.

C.8. Adaptive Threshold

This section justifies the chosen adaptive threshold parameters. Two different case segments are selected for their informativeness in this section. The same approach to parameter selection was applied to all the other cases to find the eventual chosen values $w = 3000$ and $k = 3.5$.

Figure C.5 shows that for two different cases, the window size influences the height and smoothness of the threshold. A short window increases the responsiveness to the current HI value, risking oversensitivity to short and sudden HI increases like in Figure C.5a and Figure C.5d. A longer window size will leave more time for deciding the initial threshold value, but waiting too long will underestimate slower deterioration behavior like in Figure C.5f.

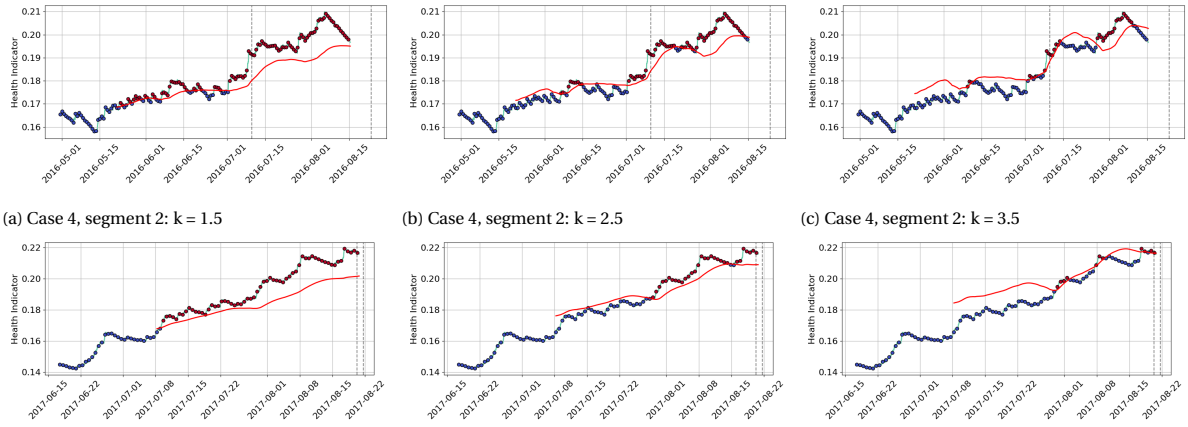
Figure C.6 demonstrates how k influences the responsiveness or sensitivity of the threshold to a changing standard deviation of the health indicator. More responsiveness to changes in the HI trend in this case improves the timing of the detected fault, with an optimum of 3.5. Increasing k further will result in more missed faults, as the HI curve will no longer intersect the threshold.



(a) Case 4, segment 2: w = 1500 (b) Case 4, segment 2: w = 3000 (c) Case 4, segment 2: w = 4500

(d) Case 4, segment 4: w = 1500 (e) Case 4, segment 4: w = 3000 (f) Case 4, segment 4: w = 4500

Figure C.5: Adaptive threshold sensitivity to window size, chosen value is w = 3000



(a) Case 4, segment 2: k = 1.5 (b) Case 4, segment 2: k = 2.5 (c) Case 4, segment 2: k = 3.5

(d) Case 4, segment 4: k = 1.5 (e) Case 4, segment 4: k = 2.5 (f) Case 4, segment 4: k = 3.5

Figure C.6: Adaptive threshold sensitivity to varying responsiveness to standard deviation, the chosen value is k=3.5

C.9. Additional LSTM-AE Result Plots

Result figures are given for each case.

Case 1

Figures showing Case 1 output, where T11 is the testing turbine.

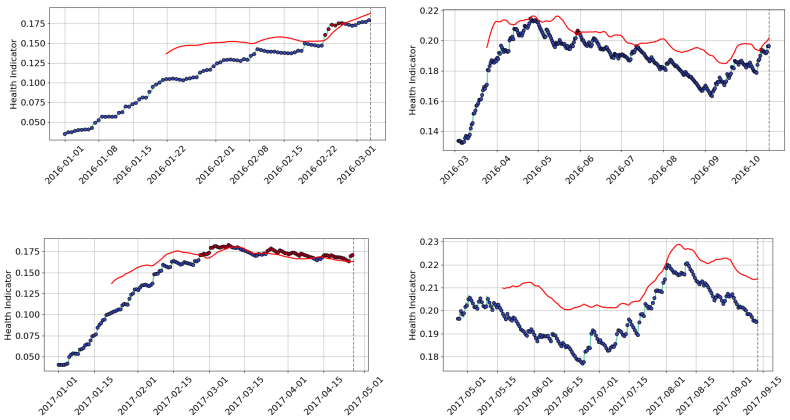


Figure C.7: Case 1

Case 2

Figures showing Case 2 output, where T01 is the testing turbine.

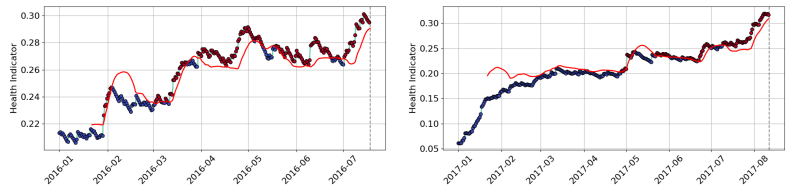


Figure C.8: Case 2

Case 3

Figures showing Case 3 output, where T06 is the testing turbine.

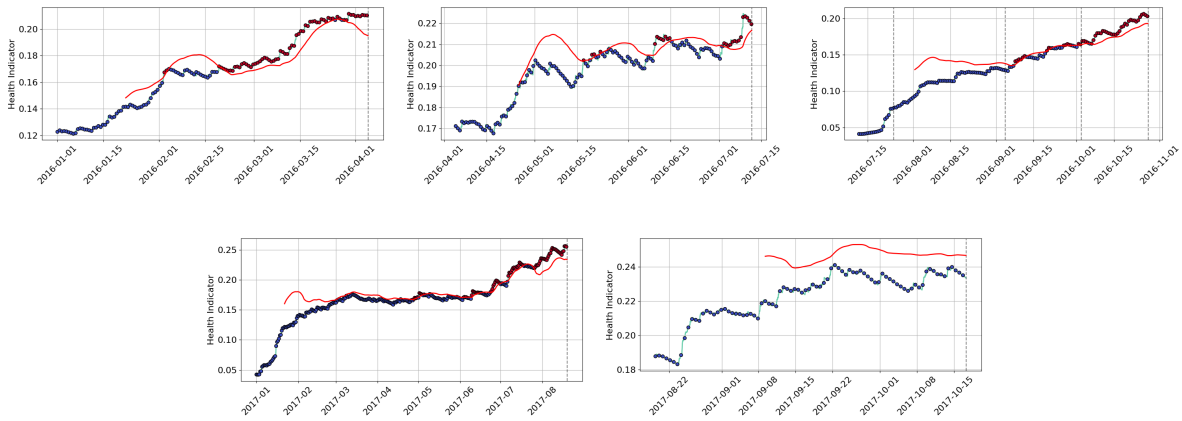


Figure C.9: Case 3

Case 4

Figures showing Case 4 output, where T07 is the testing turbine.

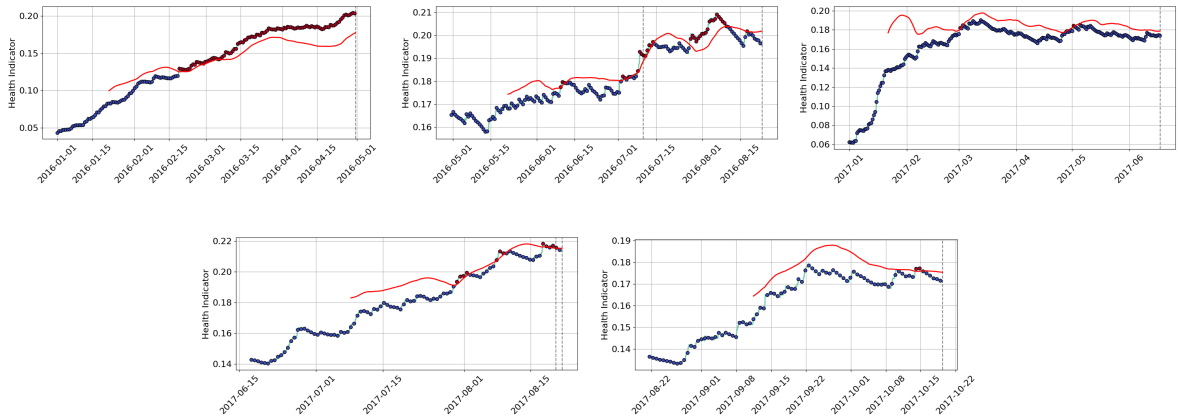


Figure C.10: Case 4

C.10. Verification of sensor selection method

Test Case	Mono.	Trend.	Prog.	Detected Faults
Case 1	0.51	0.16	0.78	1/4
Case 2	0.74	0.44	0.82	0/2
Case 3	0.58	0.53	0.72	3/5
Case 4	0.52	0.80	0.91	3/5

Table C.6: Monotonocity (Mono.), Trendability (Trend.), and Prognosability (Prog.) of the EDP test cases, as well as their share of detected faults, when the complete SCADA sensor dataset is used as input.

C.11. Supplementary Boxplots for DEC

Additional Boxplots of configurable parameters switching certain model functions, training-related parameters, or DSCAN parameters.

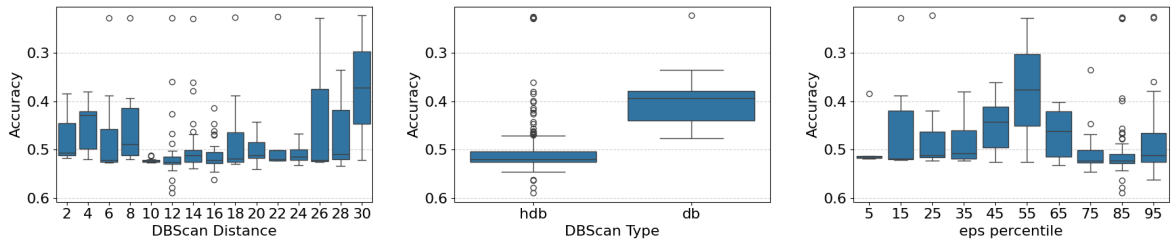


Figure C.11: DBSCAN Distance and Scaler decision

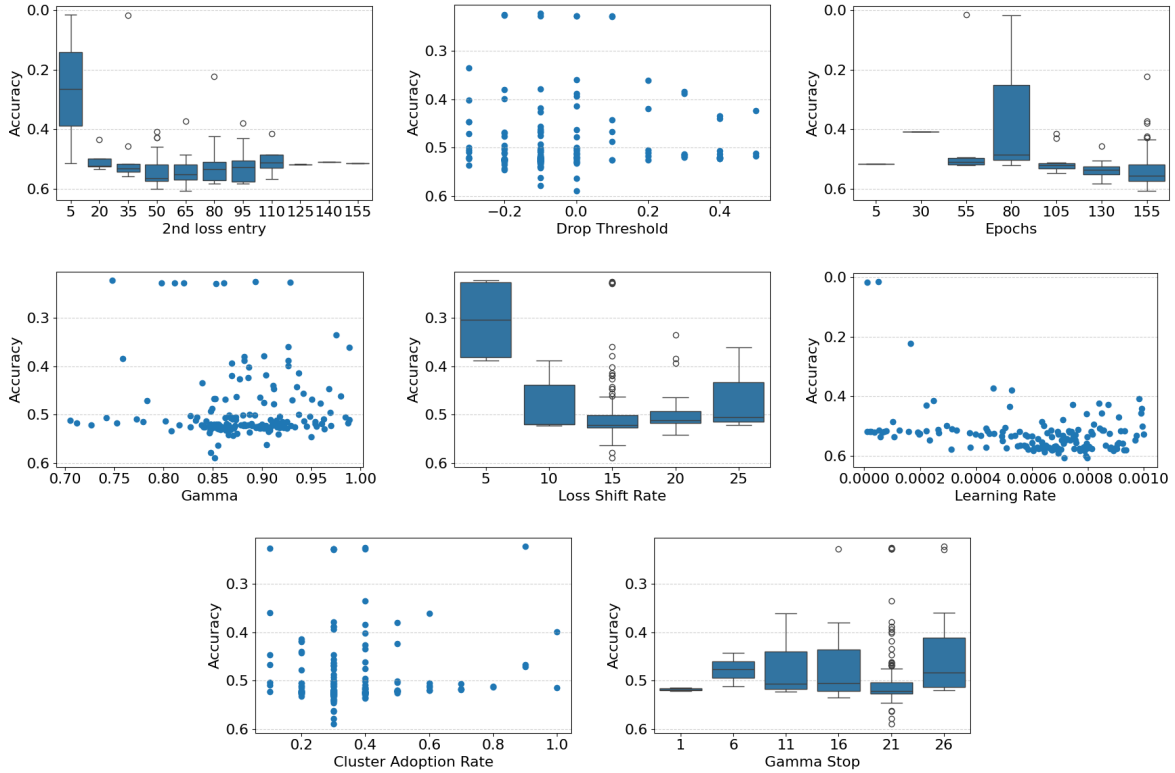


Figure C.12: CNN Training Parameter Sensitivity

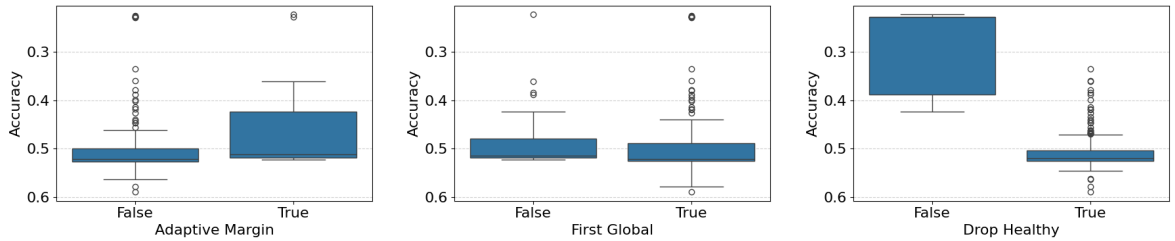


Figure C.13: Method Decision Booleans

C.12. Supplementary Boxplots for Transformer

Additional box plots of configurable parameters related to the training process of the Transformer.

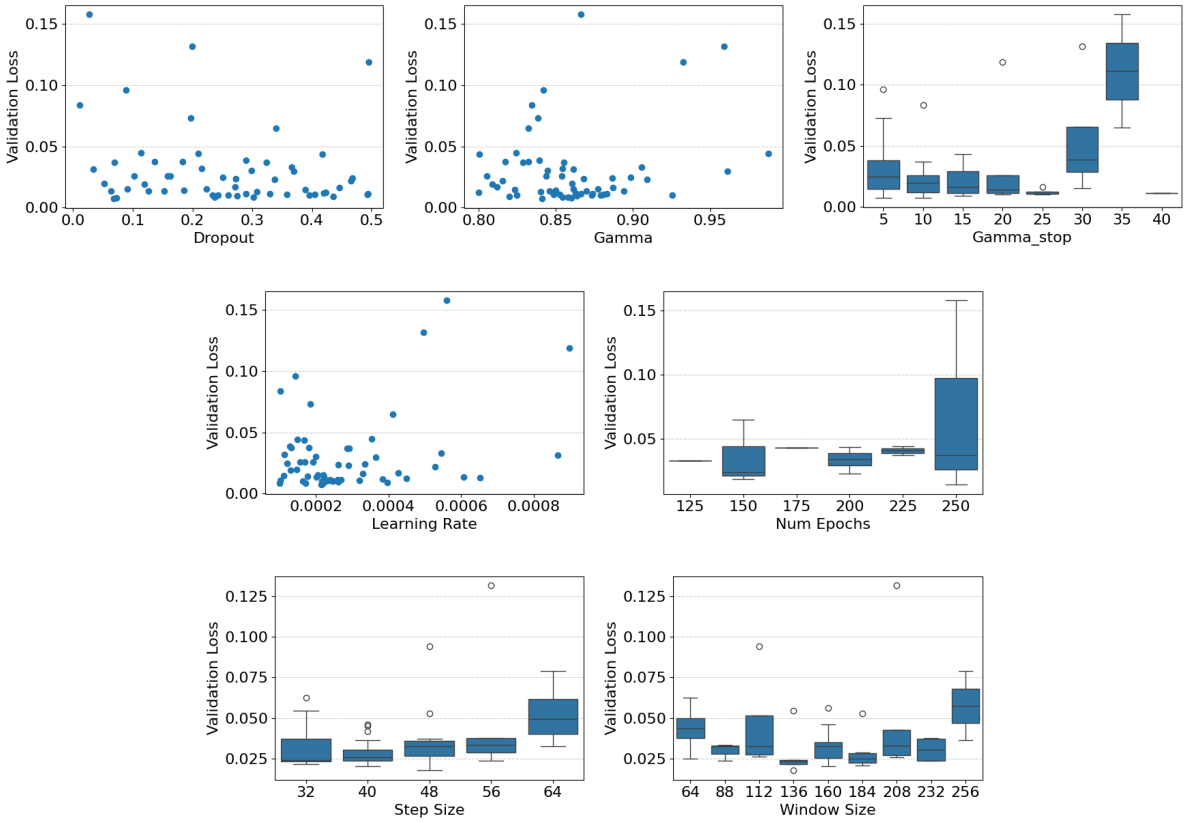


Figure C.14: Box Plots for Transformer sensitivity after 100 Optuna parameter optimization trials, showing validation loss per hyperparameter.

C.13. Additional Transformer Result Plots

Result figures are given for each case.

Case 1

Figures showing Case 1 output, where T11 is the testing turbine.

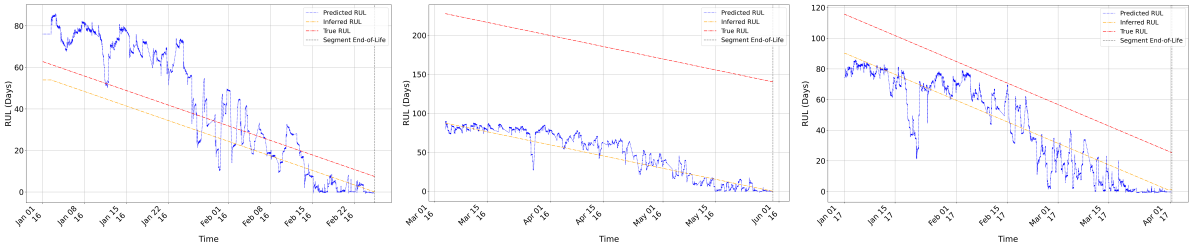


Figure C.15: Case 1

Case 2

Figures showing Case 2 output, where T01 is the testing turbine.

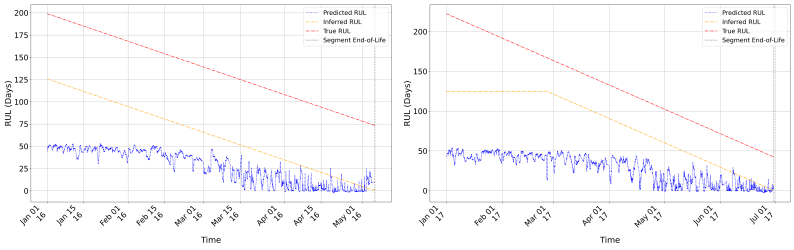


Figure C.16: Case 2

Case 3

Figures showing Case 3 output, where T06 is the testing turbine.

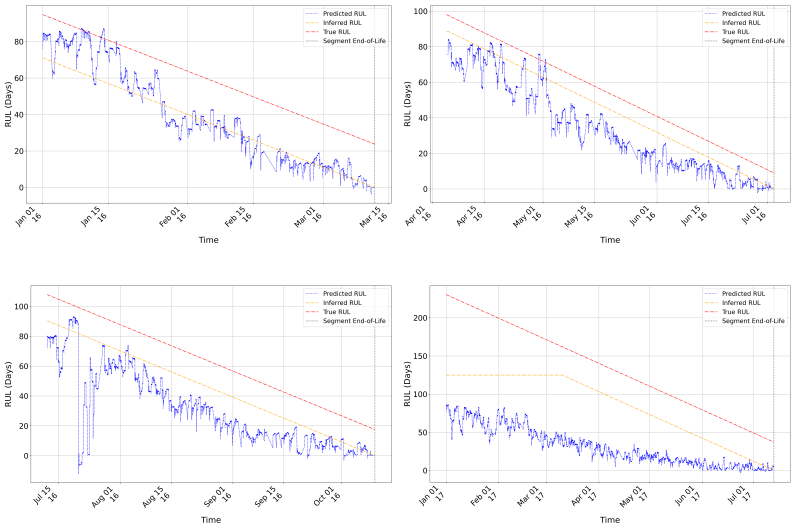


Figure C.17: Case 3

Case 4

Figures showing Case 4 output, where T07 is the testing turbine.

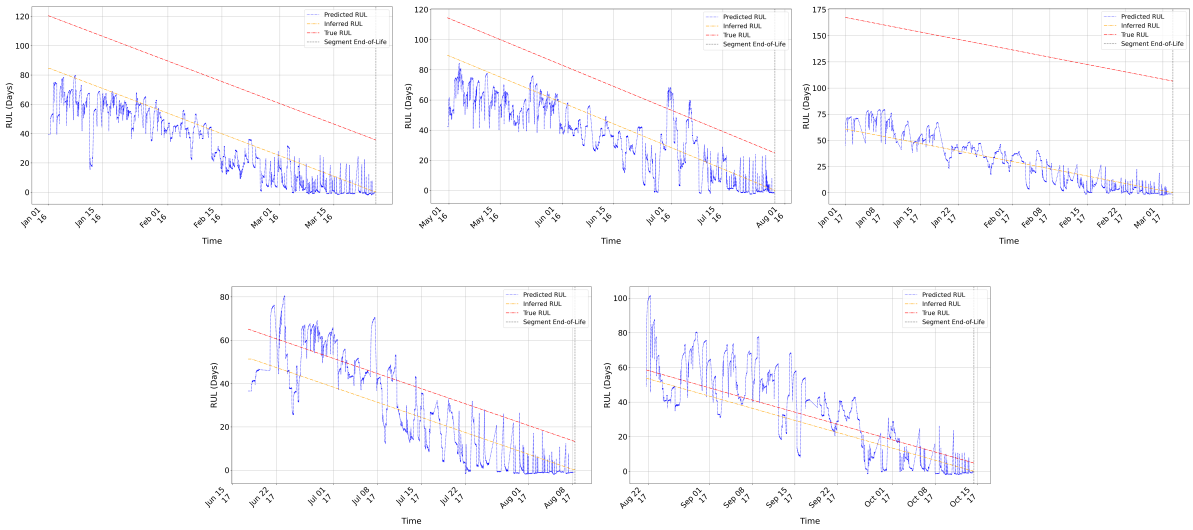


Figure C.18: Case 4

Bibliography

[1] Mehmet Bilgili and Hakan Alphan. “Global growth in offshore wind turbine technology”. In: *Clean Technologies and Environmental Policy* 24.7 (2022), pp. 2215–2227.

[2] Caichao Zhu and Yao Li. “Reliability analysis of wind turbines”. In: IntechOpen London, UK, 2018, pp. 169–186.

[3] Tyler Stehly Philipp Beiter Patrick Duffy. “2019 Cost of Wind Energy Review”. In: *NREL* NREL/TP-5000-78471 (2020).

[4] JinJiang Li et al. “A review on development of offshore wind energy conversion system”. In: *International Journal of Energy Research* 44.12 (2020), pp. 9283–9297.

[5] Iain Allan Dinwoodie and David McMillan. “Operational strategies for offshore wind turbines to mitigate failure rate uncertainty on operational costs and revenue”. In: *IET Renewable Power Generation* 8.4 (2014), pp. 359–366. DOI: https://doi.org/10.1049/iet-rpg.2013.0232. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-rpg.2013.0232. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-rpg.2013.0232.

[6] María Isabel Blanco. “The economics of wind energy”. In: *Renewable and sustainable energy reviews* 13.6-7 (2009), pp. 1372–1382.

[7] Yashwant Sinha and John A Steel. “A progressive study into offshore wind farm maintenance optimisation using risk based failure analysis”. In: *Renewable and Sustainable Energy Reviews* 42 (2015), pp. 735–742.

[8] James Carroll, Alasdair McDonald, and David McMillan. “Failure rate, repair time and unscheduled O&M cost analysis of offshore wind turbines”. In: *Wind Energy* 19.6 (2016), pp. 1107–1119. DOI: https://doi.org/10.1002/we.1887. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.1887. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1887.

[9] James Carroll et al. “Availability, operation and maintenance costs of offshore wind turbines with different drive train configurations”. In: *Wind Energy* 20.2 (2017), pp. 361–378.

[10] Gustavo Oliveira et al. “Continuous dynamic monitoring of an onshore wind turbine”. In: *Engineering Structures* 164 (2018), pp. 22–39.

[11] WY Liu et al. “The structure healthy condition monitoring and fault diagnosis methods in wind turbines: A review”. In: *Renewable and Sustainable Energy Reviews* 44 (2015), pp. 466–472.

[12] Bryant Le and John Andrews. “Modelling wind turbine degradation and maintenance”. In: *Wind Energy* 19.4 (2016), pp. 571–591. DOI: https://doi.org/10.1002/we.1851. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.1851. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.1851.

[13] FaroWind. *Blade Inspeciton*. URL: https://farowind.com/blade-inspection/.

[14] Rafael Gouriveau, Kamal Medjaher, and Noureddine Zerhouni. *From prognostics and health systems management to predictive maintenance 1: Monitoring and prognostics*. John Wiley & Sons, 2016.

[15] P. Zhou and P.T. Yin. “An opportunistic condition-based maintenance strategy for offshore wind farm based on predictive analytics”. In: *Renewable and Sustainable Energy Reviews* 109 (2019), pp. 1–9. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2019.03.049. URL: https://www.sciencedirect.com/science/article/pii/S1364032119301893.

[16] Kai Schenkelberg, Ulrich Seidenberg, and Fazel Ansari. “Analyzing the impact of maintenance on profitability using dynamic bayesian networks”. In: *Procedia CIRP* 88 (2020). 13th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 17-19 July 2019, Gulf of Naples, Italy, pp. 42–47. ISSN: 2212-8271. DOI: https://doi.org/10.1016/j.procir.2020.05.008. URL: https://www.sciencedirect.com/science/article/pii/S2212827120303231.

[17] Thi-Anh-Tuyet Nguyen, Shuo-Yan Chou, and Tiffany Hui-Kuang Yu. "Developing an exhaustive optimal maintenance schedule for offshore wind turbines based on risk-assessment, technical factors and cost-effective evaluation". In: *Energy* 249 (2022), p. 123613. ISSN: 0360-5442. DOI: https://doi.org/10.1016/j.energy.2022.123613. URL: https://www.sciencedirect.com/science/article/pii/S0360544222005163.

[18] Alexander Karyotakis. "On the optimisation of operation and maintenance strategies for offshore wind farms". In: *PhD thesis* (2011).

[19] Mahmood Shafiee. "Maintenance logistics organization for offshore wind energy: Current progress and future perspectives". In: *Renewable Energy* 77 (2015), pp. 182–193. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2014.11.045. URL: https://www.sciencedirect.com/science/article/pii/S0960148114007605.

[20] Alan Turnbull and James Carroll. "Cost Benefit of Implementing Advanced Monitoring and Predictive Maintenance Strategies for Offshore Wind Farms". In: *Energies* 14.16 (2021). ISSN: 1996-1073. DOI: 10.3390/en14164922. URL: https://www.mdpi.com/1996-1073/14/16/4922.

[21] Md Imran Hasan Tusar and Bhaba R. Sarker. "Maintenance cost minimization models for offshore wind farms: A systematic and critical review". In: *International Journal of Energy Research* 46.4 (2022), pp. 3739–3765. DOI: https://doi.org/10.1002/er.7425. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/er.7425. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.7425.

[22] David W. Coit Petros Papadopoulos and Ahmed Aziz Ezzat. "STOCHOS: Stochastic opportunistic maintenance scheduling for offshore wind farms". In: *IISE Transactions* 0.0 (2023), pp. 1–15. DOI: 10.1080/24725854.2022.2152913. eprint: https://doi.org/10.1080/24725854.2022.2152913. URL: https://doi.org/10.1080/24725854.2022.2152913.

[23] Md Imran Hasan Tusar and Bhaba R Sarker. "Spare parts control strategies for offshore wind farms: A critical review and comparative study". In: *Wind Engineering* 46.5 (May 2022), pp. 1629–1656. DOI: 10.1177/0309524x221095258.

[24] Chandra Ade Irawan et al. "Optimisation of maintenance routing and scheduling for offshore wind farms". In: *European Journal of Operational Research* 256.1 (2017), pp. 76–89.

[25] Iraklis Lazakis and Shahroz Khan. "An optimization framework for daily route planning and scheduling of maintenance vessel activities in offshore wind farms". In: *Ocean Engineering* 225 (2021), p. 108752. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2021.108752. URL: https://www.sciencedirect.com/science/article/pii/S0029801821001876.

[26] Fangfang Ding and Zhigang Tian. "Opportunistic maintenance for wind farms considering multi-level imperfect maintenance thresholds". In: *Renewable Energy* 45 (2012), pp. 175–182. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2012.02.030. URL: https://www.sciencedirect.com/science/article/pii/S0960148112001802.

[27] Magnus Stålhane et al. "Optimizing vessel fleet size and mix to support maintenance operations at offshore wind farms". In: *European Journal of Operational Research* 276.2 (2019), pp. 495–509. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2019.01.023. URL: https://www.sciencedirect.com/science/article/pii/S0377221719300426.

[28] Mingxin Li et al. "An opportunistic maintenance strategy for offshore wind turbine system considering optimal maintenance intervals of subsystems". In: *Ocean Engineering* 216 (2020), p. 108067. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2020.108067. URL: https://www.sciencedirect.com/science/article/pii/S002980182031009X.

[29] M Borsotti, RR Negenborn, and X Jiang. "Model predictive control framework for optimizing offshore wind O&M". In: *Advances in Maritime Technology and Engineering*. CRC Press, 2024, pp. 533–546.

[30] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[31] Wasim Ahmad et al. "A reliable technique for remaining useful life estimation of rolling element bearings using dynamic regression models". In: *Reliability Engineering & System Safety* 184 (2019), pp. 67–76.

[32] Rene Jaros et al. “Advanced Signal Processing Methods for Condition Monitoring”. In: *Archives of Computational Methods in Engineering* 30.3 (Oct. 2022), pp. 1553–1577. DOI: 10.1007/s11831-022-09834-4.

[33] Wei Qiao and Dingguo Lu. “A Survey on Wind Turbine Condition Monitoring and Fault Diagnosis—Part II: Signals and Signal Processing Methods”. In: *IEEE Transactions on Industrial Electronics* 62.10 (2015), pp. 6546–6557. DOI: 10.1109/TIE.2015.2422394.

[34] Weiting Zhang, Dong Yang, and Hongchao Wang. “Data-Driven Methods for Predictive Maintenance of Industrial Equipment: A Survey”. In: *IEEE Systems Journal* 13.3 (2019), pp. 2213–2227. DOI: 10.1109/JSYST.2019.2905565.

[35] Olga Fink. “Data-driven intelligent predictive maintenance of industrial assets”. In: *Women in Industrial and Systems Engineering: Key Advances and Perspectives on Emerging Topics* (2020), pp. 589–605.

[36] Chen Xiongzi et al. “Remaining useful life prognostic estimation for aircraft subsystems or components: A review”. In: *Ieee 2011 10th international conference on electronic measurement & instruments*. Vol. 2. IEEE. 2011, pp. 94–98.

[37] Tomasz Barszcz. *Vibration-based condition monitoring of wind turbines*. Springer, 2019. Chap. 3, pp. 96–100.

[38] Meik Schlechtingen, Ilmar Ferreira Santos, and Sofiane Achiche. “Wind turbine condition monitoring based on SCADA data using normal behavior models. Part 1: System description”. In: *Applied Soft Computing* 13.1 (2013), pp. 259–270. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2012.08.033. URL: https://www.sciencedirect.com/science/article/pii/S1568494612003821.

[39] Ying Du et al. “Damage detection techniques for wind turbine blades: A review”. In: *Mechanical Systems and Signal Processing* 141 (2020), p. 106445.

[40] Zhengru Ren et al. “Offshore wind turbine operations and maintenance: A state-of-the-art review”. In: *Renewable and Sustainable Energy Reviews* 144 (2021), p. 110886. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2021.110886. URL: https://www.sciencedirect.com/science/article/pii/S1364032121001805.

[41] C.J. Crabtree, D. Zappala, and P.J. Tavner. *Survey of commercially available condition monitoring systems for wind turbines*. Technical Report. DU, May 2014. URL: http://dro.dur.ac.uk/12497/.

[42] Z. Hameed et al. “Condition monitoring and fault detection of wind turbines and related algorithms: A review”. In: *Renewable and Sustainable Energy Reviews* 13.1 (2009), pp. 1–39. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2007.05.008. URL: https://www.sciencedirect.com/science/article/pii/S1364032107001098.

[43] J.M.P. Pérez and F.P.G. Márquez. “11 - Condition monitoring and fault diagnosis in wind energy systems”. In: *Eco-Friendly Innovation in Electricity Transmission and Distribution Networks*. Ed. by Jean-Luc Bessède. Oxford: Woodhead Publishing, 2015, pp. 221–241. ISBN: 978-1-78242-010-1. DOI: https://doi.org/10.1016/B978-1-78242-010-1.00011-2. URL: https://www.sciencedirect.com/science/article/pii/B9781782420101000112.

[44] Wei Qiao and Dingguo Lu. “A survey on wind turbine condition monitoring and fault diagnosis—Part I: Components and subsystems”. In: *IEEE Transactions on Industrial Electronics* 62.10 (2015), pp. 6536–6545.

[45] Pierre Tchakoua et al. “Wind Turbine Condition Monitoring: State-of-the-Art Review, New Trends, and Future Challenges”. In: *Energies* 7.4 (2014), pp. 2595–2630. ISSN: 1996-1073. DOI: 10.3390/en7042595. URL: https://www.mdpi.com/1996-1073/7/4/2595.

[46] Fausto Pedro García Márquez et al. “Condition monitoring of wind turbines: Techniques and methods”. In: *Renewable Energy* 46 (2012), pp. 169–178. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2012.03.003. URL: https://www.sciencedirect.com/science/article/pii/S0960148112001899.

[47] Julia Nilsson and Lina Bertling. “Maintenance Management of Wind Power Systems Using Condition Monitoring Systems—Life Cycle Cost Analysis for Two Case Studies”. In: *IEEE Transactions on Energy Conversion* 22.1 (2007), pp. 223–229. DOI: 10.1109/TEC.2006.889623.

[48] Wenxian Yang et al. “Wind turbine condition monitoring: technical and commercial challenges”. In: *Wind energy* 17.5 (2014), pp. 673–693.

[49] Wind Turbines. “Part 1: Design Requirements, IEC 61400-1”. In: *International Electrotechnical Commission: Geneva, Switzerland* (2005).

[50] A. Zaher et al. “Online wind turbine fault detection through automated SCADA data analysis”. In: *Wind Energy* 12.6 (2009), pp. 574–593. DOI: https://doi.org/10.1002/we.319. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.319. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.319.

[51] Ziqian Kong et al. “Condition monitoring of wind turbines based on spatio-temporal fusion of SCADA data by convolutional neural networks and gated recurrent units”. In: *Renewable Energy* 146 (2020), pp. 760–768. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2019.07.033. URL: https://www.sciencedirect.com/science/article/pii/S0960148119310535.

[52] Xiaohang Jin, Zhuangwei Xu, and Wei Qiao. “Condition Monitoring of Wind Turbine Generators Using SCADA Data Analysis”. In: *IEEE Transactions on Sustainable Energy* 12.1 (2021), pp. 202–210. DOI: 10.1109/TSTE.2020.2989220.

[53] Xingchen Liu, Juan Du, and Zhi-Sheng Ye. “A Condition Monitoring and Fault Isolation System for Wind Turbine Based on SCADA Data”. In: *IEEE Transactions on Industrial Informatics* 18.2 (2022), pp. 986–995. DOI: 10.1109/TII.2021.3075239.

[54] Jeroen Hes. “Predictive Maintenance of Offshore Wind Turbines: A Comprehensive Review of Condition Monitoring Technologies and Maintenance Solutions, Unpublished manuscript”. In: *Delft University of Technology* (2023).

[55] J.L. Godwin, P.C. Matthews, and C. Watson. “Classification and Detection of Electrical Control System Faults Through SCADA Data Analysis”. In: *Chemical engineering transactions.* 1 (2013). EPrint Processing Status: Full text deposited in DRO, pp. 985–990. ISSN: 1974-9791. DOI: 10.3303/cet1333165. URL: https://durham-repository.worktribe.com/output/1430743.

[56] Yingning Qiu, Yanhui Feng, and David Infield. “Fault diagnosis of wind turbine with SCADA alarms based multidimensional information processing method”. In: *Renewable Energy* 145 (2020), pp. 1923–1931. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2019.07.110. URL: https://www.sciencedirect.com/science/article/pii/S0960148119311309.

[57] Hamed Badihi et al. “A Comprehensive Review on Signal-Based and Model-Based Condition Monitoring of Wind Turbines: Fault Diagnosis and Lifetime Prognosis”. In: *Proceedings of the IEEE* 110.6 (2022), pp. 754–806. DOI: 10.1109/JPROC.2022.3171691.

[58] Yang Hu et al. “Prognostics and health management: A review from the perspectives of design, development and decision”. In: *Reliability Engineering & System Safety* 217 (2022), p. 108063. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.108063. URL: https://www.sciencedirect.com/science/article/pii/S0951832021005652.

[59] Kevin Leahy et al. “Issues with Data Quality for Wind Turbine Condition Monitoring and Reliability Analyses”. In: *Energies* 12.2 (2019). ISSN: 1996-1073. DOI: 10.3390/en12020201. URL: https://www.mdpi.com/1996-1073/12/2/201.

[60] Christos Kaidis, Bahri Uzunoglu, and Filippos Amoiralis. “Wind turbine reliability estimation for different assemblies and failure severity categories”. In: *IET Renewable Power Generation* 9.8 (2015), pp. 892–899.

[61] Valerie A Peters, Alistair B Ogilvie, and Cody R Bond. “Continuous reliability enhancement for wind (CREW) database: wind plant reliability benchmark”. In: *Sandia National Laboratories, Energy, Climate, & Infrastructure Security. energy. sandia. gov* (2012).

[62] M D Reder, E Gonzalez, and J J Melero. “Wind Turbine Failures - Tackling current Problems in Failure Data Analysis”. In: *Journal of Physics: Conference Series* 753.7 (Sept. 2016), p. 072027. DOI: 10.1088/1742-6596/753/7/072027. URL: https://dx.doi.org/10.1088/1742-6596/753/7/072027.

[63] Elena Gonzalez et al. “Using high-frequency SCADA data for wind turbine performance monitoring: A sensitivity study”. In: *Renewable Energy* 131 (2019), pp. 841–853. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2018.07.068. URL: https://www.sciencedirect.com/science/article/pii/S0960148118308656.

[64] J Tautz-Weinert and SJ Watson. *Using scada data for wind turbine condition monitoring–a review. IET Renewable Power Generation, 11 (4), 382-394*. 2016.

[65] Jorge Maldonado-Correa et al. "Using SCADA Data for Wind Turbine Condition Monitoring: A Systematic Literature Review". In: *Energies* 13.12 (2020). ISSN: 1996-1073. DOI: 10.3390/en13123132. URL: https://www.mdpi.com/1996-1073/13/12/3132.

[66] Julia Walgern, Lennart Peters, and Reinhard Madlener. "Economic evaluation of maintenance strategies for offshore wind turbines based on condition monitoring systems". In: (2017).

[67] Estefania Artigao et al. "Wind turbine reliability: A comprehensive review towards effective condition monitoring development". In: *Applied Energy* 228 (2018), pp. 1569–1583. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2018.07.037. URL: https://www.sciencedirect.com/science/article/pii/S0306261918310651.

[68] Georgios Nikitas, Subhamoy Bhattacharya, and Nathan Vimalan. "16 - Wind Energy". In: *Future Energy (Third Edition)*. Ed. by Trevor M. Letcher. Third Edition. Elsevier, 2020, pp. 331–355. ISBN: 978-0-08-102886-5. DOI: https://doi.org/10.1016/B978-0-08-102886-5.00016-5. URL: https://www.sciencedirect.com/science/article/pii/B9780081028865000165.

[69] Harriet Fox et al. "A Review of Predictive and Prescriptive Offshore Wind Farm Operation and Maintenance". In: *Energies* 15.2 (2022). ISSN: 1996-1073. DOI: 10.3390/en15020504. URL: https://www.mdpi.com/1996-1073/15/2/504.

[70] Vepa Atamuradov Fatih Camci Kamal Medjaher and Ashyrmuhammet Berdinyazov. "Integrated maintenance and mission planning using remaining useful life information". In: *Engineering Optimization* 51.10 (2019), pp. 1794–1809. DOI: 10.1080/0305215X.2018.1552951. eprint: https://doi.org/10.1080/0305215X.2018.1552951. URL: https://doi.org/10.1080/0305215X.2018.1552951.

[71] Wlamir Olivares Loesch Vianna and Takashi Yoneyama. "Predictive Maintenance Optimization for Aircraft Redundant Systems Subjected to Multiple Wear Profiles". In: *IEEE Systems Journal* 12.2 (2018), pp. 1170–1181. DOI: 10.1109/JSYST.2017.2667232.

[72] Khanh T.P. Nguyen and Kamal Medjaher. "A new dynamic predictive maintenance framework using deep learning for failure prognostics". In: *Reliability Engineering & System Safety* 188 (2019), pp. 251–262. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2019.03.018. URL: https://www.sciencedirect.com/science/article/pii/S0951832018311050.

[73] Chuang Chen et al. "Data-driven predictive maintenance strategy considering the uncertainty in remaining useful life prediction". In: *Neurocomputing* 494 (2022), pp. 79–88. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2022.04.055. URL: https://www.sciencedirect.com/science/article/pii/S0925231222004428.

[74] Mihaela Mitici et al. "Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines". In: *Reliability Engineering & System Safety* 234 (2023), p. 109199. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2023.109199. URL: https://www.sciencedirect.com/science/article/pii/S095183202300114X.

[75] Liangliang Zhuang, Ancha Xu, and Xiao-Lin Wang. "A prognostic driven predictive maintenance framework based on Bayesian deep learning". In: *Reliability Engineering & System Safety* 234 (2023), p. 109181. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2023.109181. URL: https://www.sciencedirect.com/science/article/pii/S0951832023000960.

[76] Juseong Lee and Mihaela Mitici. "Deep reinforcement learning for predictive aircraft maintenance using probabilistic Remaining-Useful-Life prognostics". In: *Reliability Engineering & System Safety* 230 (2023), p. 108908. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2022.108908. URL: https://www.sciencedirect.com/science/article/pii/S0951832022005233.

[77] Musavir Hussain et al. "Condition Monitoring and Fault Diagnosis of Wind Turbine: A Systematic Literature Review". In: *IEEE Access* 12 (2024), pp. 190220–190239. DOI: 10.1109/ACCESS.2024.3514747.

[78] Ravi Pandit and Jianlin Wang. "A comprehensive review on enhancing wind turbine applications with advanced SCADA data analytics and practical insights". In: *IET Renewable Power Generation* 18.4 (2024), pp. 722–742. DOI: https://doi.org/10.1049/rpg2.12920. eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/rpg2.12920. URL: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/rpg2.12920.

[79] Simona Bogoevska et al. “A Data-Driven Diagnostic Framework for Wind Turbine Structures: A Holistic Approach”. In: *Sensors* 17.4 (2017). ISSN: 1424-8220. DOI: 10.3390/s17040720. URL: https://www.mdpi.com/1424-8220/17/4/720.

[80] Jinhao Lei, Chao Liu, and Dongxiang Jiang. “Fault diagnosis of wind turbine based on Long Short-term memory networks”. In: *Renewable Energy* 133 (2019), pp. 422–432. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2018.10.031. URL: https://www.sciencedirect.com/science/article/pii/S0960148118312151.

[81] Annalisa Santolamazza, Daniele Dadi, and Vito Introna. “A Data-Mining Approach for Wind Turbine Fault Detection Based on SCADA Data Analysis Using Artificial Neural Networks”. In: *Energies* 14.7 (2021). ISSN: 1996-1073. DOI: 10.3390/en14071845. URL: https://www.mdpi.com/1996-1073/14/7/1845.

[82] Wisdom Udo and Yar Muhammad. “Data-driven predictive maintenance of wind turbine based on SCADA data”. In: *IEEE Access* 9 (2021), pp. 162370–162388.

[83] Gabriel de Souza Pereira Gomes et al. “Wind Turbine Remaining Useful Life Prediction Using Small Dataset and Machine Learning Techniques”. In: *Journal of Control, Automation and Electrical Systems* 35.2 (Mar. 2024), pp. 337–345. DOI: 10.1007/s40313-024-01076-y.

[84] Uwe Lützen and Serdar Beji. “Predictive maintenance for offshore wind turbines through deep learning and online clustering of unsupervised subsystems: a real-world implementation”. In: *Journal of Ocean Engineering and Marine Energy* 10.3 (2024), pp. 627–640.

[85] Xinming Li et al. “Multi-sensor fusion fault diagnosis method of wind turbine bearing based on adaptive convergent viewable neural networks”. In: *Reliability Engineering & System Safety* 245 (2024), p. 109980. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2024.109980. URL: https://www.sciencedirect.com/science/article/pii/S0951832024000553.

[86] Ramakrishna SS Nuvvula et al. “Machine Learning-Driven Predictive Maintenance Framework for Anomaly Detection and Prognostics in Wind Farm Operations”. In: *2024 12th International Conference on Smart Grid (icSmartGrid)*. IEEE. 2024, pp. 284–289.

[87] Lala Rajaoarisoa, Raubertin Randrianandraina, and Moamar Sayed-Mouchaweh. “Predictive maintenance model-based on multi-stage neural network systems for wind turbines”. In: *2024 international conference on artificial intelligence, computer, data sciences and applications (acdsa)*. IEEE. 2024, pp. 1–7.

[88] Syed Shazaib Shah, Tan Daoliang, and Sah Chandan Kumar. “RUL forecasting for wind turbine predictive maintenance based on deep learning”. In: *Heliyon* 10.20 (2024).

[89] Rui He et al. “Predictive maintenance for offshore wind farms with incomplete and biased prognostic information”. In: *Ocean Engineering* 322 (2025), p. 120541. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2025.120541. URL: https://www.sciencedirect.com/science/article/pii/S0029801825002562.

[90] Yi Qin et al. “A prognostic driven dynamic predictive maintenance decision-making model for offshore wind turbine systems”. In: *Ocean Engineering* 338 (2025), p. 122041. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2025.122041. URL: https://www.sciencedirect.com/science/article/pii/S0029801825017470.

[91] Hao Yan et al. “Multiple Sensor Data Fusion for Degradation Modeling and Prognostics Under Multiple Operational Conditions”. In: *IEEE Transactions on Reliability* 65.3 (2016), pp. 1416–1426. DOI: 10.1109/TR.2016.2575449.

[92] Yuning Qian, Ruqiang Yan, and Robert X Gao. “A multi-time scale approach to remaining useful life prediction in rolling bearing”. In: *Mechanical Systems and Signal Processing* 83 (2017), pp. 549–567.

[93] Yaguo Lei et al. “Machinery health prognostics: A systematic review from data acquisition to RUL prediction”. In: *Mechanical systems and signal processing* 104 (2018), pp. 799–834.

[94] Baoping Cai et al. “Remaining Useful Life Estimation of Structure Systems Under the Influence of Multiple Causes: Subsea Pipelines as a Case Study”. In: *IEEE Transactions on Industrial Electronics* 67.7 (2020), pp. 5737–5747. DOI: 10.1109/TIE.2019.2931491.

[95] Samir Khan and Takehisa Yairi. “A review on the application of deep learning in system health management”. In: *Mechanical Systems and Signal Processing* 107 (2018), pp. 241–265. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2017.11.024. URL: https://www.sciencedirect.com/science/article/pii/S0888327017306064.

[96] Adrian Stetco et al. “Machine learning methods for wind turbine condition monitoring: A review”. In: *Renewable energy* 133 (2019), pp. 620–635.

[97] Simon Vollert and Andreas Theissler. “Challenges of machine learning-based RUL prognosis: A review on NASA’s C-MAPSS data set”. In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 2021, pp. 1–8. DOI: 10.1109/ETFA45728.2021.9613682.

[98] Carlos Ferreira and Gil Goncalves. “Remaining Useful Life prediction and challenges: A literature review on the use of Machine Learning Methods”. In: *Journal of Manufacturing Systems* 63 (2022), pp. 550–562. ISSN: 0278-6125. DOI: https://doi.org/10.1016/j.jmsy.2022.05.010. URL: https://www.sciencedirect.com/science/article/pii/S0278612522000796.

[99] Mathias Liewald et al. “Perspectives on data-driven models and its potentials in metal forming and blanking technologies”. In: *Production Engineering* 16.5 (2022), pp. 607–625.

[100] Andrew Clifton et al. “Grand challenges in the digitalisation of wind energy”. In: *Wind Energy Science* 8.6 (June 2023), pp. 947–974. DOI: 10.5194/wes-8-947-2023.

[101] Ravil Muhamedyev. “Machine learning methods: An overview”. In: *Computer modelling & new technologies* 19.6 (2015), pp. 14–29.

[102] Satish Chandra Gupta. *An Engineer’s Trek into Machine Learning*. URL: https://www.ml4devs.com/articles/machine-learning-intro-for-developers/.

[103] Thyago P. Carvalho et al. “A systematic literature review of machine learning methods applied to predictive maintenance”. In: *Computers & Industrial Engineering* 137 (2019), p. 106024. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2019.106024. URL: https://www.sciencedirect.com/science/article/pii/S0360835219304838.

[104] Tiago Zonta et al. “Predictive maintenance in the Industry 4.0: A systematic literature review”. In: *Computers & Industrial Engineering* 150 (2020), p. 106889. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2020.106889. URL: https://www.sciencedirect.com/science/article/pii/S0360835220305787.

[105] Tongda Sun et al. “Fault diagnosis methods based on machine learning and its applications for wind turbines: A review”. In: *Ieee Access* 9 (2021), pp. 147481–147511.

[106] Imad A Basheer and Maha Hajmeer. “Artificial neural networks: fundamentals, computing, design, and application”. In: *Journal of microbiological methods* 43.1 (2000), pp. 3–31.

[107] Alberto Pliego Marugán et al. “A survey of artificial neural network in wind energy systems”. In: *Applied Energy* 228 (2018), pp. 1822–1836. ISSN: 0306-2619. DOI: https://doi.org/10.1016/j.apenergy.2018.07.084. URL: https://www.sciencedirect.com/science/article/pii/S0306261918311048.

[108] Divish Rengasamy, Hervé P. Morvan, and Grazziela P. Figueredo. “Deep Learning Approaches to Aircraft Maintenance, Repair and Overhaul: A Review”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 150–156. DOI: 10.1109/ITSC.2018.8569502.

[109] Duy-Tang Hoang and Hee-Jun Kang. “A survey on Deep Learning based bearing fault diagnosis”. In: *Neurocomputing* 335 (2019), pp. 327–335. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.06.078. URL: https://www.sciencedirect.com/science/article/pii/S0925231218312657.

[110] Purushottam Gangsar, Aditya Raj Bajpei, and Rajkumar Porwal. “A review on deep learning based condition monitoring and fault diagnosis of rotating machinery”. In: *Noise & Vibration Worldwide* 53.11 (2022), pp. 550–578. DOI: 10.1177/09574565221139638. eprint: https://doi.org/10.1177/09574565221139638. URL: https://doi.org/10.1177/09574565221139638.

[111] Chuyue Lou, M. Amine Atoui, and Xiangshun Li. “Recent deep learning models for diagnosis and health monitoring: A review of research works and future challenges”. In: *Transactions of the Institute of Measurement and Control* 46.14 (2024), pp. 2833–2870. DOI: 10.1177/01423312231157118. URL: https://doi.org/10.1177/01423312231157118.

[112] Liangwei Zhang et al. “A Review on Deep Learning Applications in Prognostics and Health Management”. In: *IEEE Access* 7 (2019), pp. 162415–162438. DOI: 10.1109/ACCESS.2019.2950985.

[113] Nikhil M Thoppil, V Vasu, and CSP Rao. “Deep learning algorithms for machinery health prognostics using time-series data: A review”. In: *Journal of Vibration Engineering & Technologies* (2021), pp. 1–23.

[114] Oscar Serradilla et al. “Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects”. In: *Applied Intelligence* 52.10 (Jan. 2022), pp. 10934–10964. DOI: 10.1007/s10489-021-03004-y.

[115] Khadija El Bouchefry and Rafael S. de Souza. “Chapter 12 - Learning in Big Data: Introduction to Machine Learning”. In: *Knowledge Discovery in Big Data from Astronomy and Earth Observation*. Ed. by Petr Škoda and Fathalrahman Adam. Elsevier, 2020, pp. 225–249. ISBN: 978-0-12-819154-5. DOI: https://doi.org/10.1016/B978-0-12-819154-5.00023-0. URL: https://www.sciencedirect.com/science/article/pii/B9780128191545000230.

[116] Patrick Schneider and Fatos Xhafa. “Chapter 3 - Anomaly detection: Concepts and methods”. In: *Anomaly Detection and Complex Event Processing over IoT Data Streams*. Ed. by Patrick Schneider and Fatos Xhafa. Academic Press, 2022, pp. 49–66. ISBN: 978-0-12-823818-9. DOI: https://doi.org/10.1016/B978-0-12-823818-9.00013-4. URL: https://www.sciencedirect.com/science/article/pii/B9780128238189000134.

[117] AD Dongare, RR Kharde, Amit D Kachare, et al. “Introduction to artificial neural network”. In: *International Journal of Engineering and Innovative Technology (IJEIT)* 2.1 (2012), pp. 189–194.

[118] Cheng Cheng et al. “A deep learning-based remaining useful life prediction approach for bearings”. In: *IEEE/ASME transactions on mechatronics* 25.3 (2020), pp. 1243–1254.

[119] Youdao Wang, Yifan Zhao, and Sri Addepalli. “Remaining Useful Life Prediction using Deep Learning Approaches: A Review”. In: *Procedia Manufacturing* 49 (2020). Proceedings of the 8th International Conference on Through-Life Engineering Services TESConf 2019, pp. 81–88. ISSN: 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.06.015. URL: https://www.sciencedirect.com/science/article/pii/S2351978920316528.

[120] Arthur Arnx. *First neural network for beginners explained (with code)*. URL: https://towardsdatascience.com/first-neural-network-for-beginners-explained-with-code-4cfd37e06eaf.

[121] Meik Schlechtingen and Ilmar Ferreira Santos. “Comparative analysis of neural network and regression based condition monitoring approaches for wind turbine fault detection”. In: *Mechanical Systems and Signal Processing* 25.5 (2011), pp. 1849–1875. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2010.12.007. URL: https://www.sciencedirect.com/science/article/pii/S0888327010004310.

[122] Longting Chen et al. “Learning deep representation of imbalanced SCADA data for fault detection of wind turbines”. In: *Measurement* 139 (2019), pp. 370–379. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2019.03.029. URL: https://www.sciencedirect.com/science/article/pii/S0263224119302386.

[123] Fjodor van Veen and Stefan Leijnen. *A mostly complete chart of neural networks*. URL: https://www.asimovinstitute.org/neural-network-zoo/.

[124] Geoffrey Hinton. “A practical guide to training restricted Boltzmann machines”. In: *Momentum* 9.1 (2010), p. 926.

[125] Long Wang et al. “Wind Turbine Blade Breakage Monitoring With Deep Autoencoders”. In: *IEEE Transactions on Smart Grid* 9.4 (2018), pp. 2824–2833. DOI: 10.1109/TSG.2016.2621135.

[126] Hongshan Zhao et al. “Anomaly detection and fault analysis of wind turbine components based on deep learning network”. In: *Renewable Energy* 127 (2018), pp. 825–834. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2018.05.024. URL: https://www.sciencedirect.com/science/article/pii/S0960148118305457.

[127] Niklas Renström, Pramod Bangalore, and Edmund Highcock. “System-wide anomaly detection in wind turbines using deep autoencoders”. In: *Renewable Energy* 157 (2020), pp. 647–659. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2020.04.148. URL: https://www.sciencedirect.com/science/article/pii/S0960148120306856.

[128] Hansi Chen et al. “Anomaly detection and critical SCADA parameters identification for wind turbines based on LSTM-AE neural network”. In: *Renewable Energy* 172 (2021), pp. 829–840. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2021.03.078. URL: https://www.sciencedirect.com/science/article/pii/S0960148121004341.

[129] Younjeong Lee et al. “LSTM-Autoencoder Based Anomaly Detection Using Vibration Data of Wind Turbines”. In: *Sensors* 24.9 (2024). ISSN: 1424-8220. DOI: 10.3390/s24092833. URL: https://www.mdpi.com/1424-8220/24/9/2833.

[130] V Siva Brahmaiah Rama, Sung-Ho Hur, and Jung-Min Yang. “Short-Term Fault Prediction of Wind Turbines Based on Integrated RNN-LSTM”. In: *IEEE Access* 12 (2024), pp. 22465–22478. DOI: 10.1109/ACCESS.2024.3364395.

[131] Ingeborg de Pater and Mihaela Mitici. “Developing health indicators and RUL prognostics for systems with few failure instances and varying operating conditions using a LSTM autoencoder”. In: *Engineering Applications of Artificial Intelligence* 117 (2023), p. 105582. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.105582. URL: https://www.sciencedirect.com/science/article/pii/S0952197622005723.

[132] Youngwoong Choi and Sungmin Yoon. “Autoencoder-driven fault detection and diagnosis in building automation systems: Residual-based and latent space-based approaches”. In: *Building and Environment* 203 (2021), p. 108066. ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2021.108066. URL: https://www.sciencedirect.com/science/article/pii/S0360132321004686.

[133] Ana González-Muñiz et al. “Health indicator for machine condition monitoring built in the latent space of a deep autoencoder”. In: *Reliability Engineering & System Safety* 224 (2022), p. 108482. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2022.108482. URL: https://www.sciencedirect.com/science/article/pii/S0951832022001417.

[134] Ping Wu et al. “Wind Turbine Blade Breakage Monitoring With Mogrifier LSTM Autoencoder”. In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–10. DOI: 10.1109/TIM.2023.3323967.

[135] Chuang Chen et al. “Dynamic Predictive Maintenance Scheduling Using Deep Learning Ensemble for System Health Prognostics”. In: *IEEE Sensors Journal* 21.23 (2021), pp. 26878–26891. DOI: 10.1109/JSEN.2021.3119553.

[136] Linxia Liao, Wenjing Jin, and Radu Pavel. “Enhanced Restricted Boltzmann Machine With Prognosability Regularization for Prognostics and Health Assessment”. In: *IEEE Transactions on Industrial Electronics* 63.11 (2016), pp. 7076–7083. DOI: 10.1109/TIE.2016.2586442.

[137] Jason Deutsch and David He. “Using deep learning based approaches for bearing remaining useful life prediction”. In: *Annual conference of the PHM society*. Vol. 8. 1. 2016.

[138] Haizhou Chen et al. “An integrated approach to planetary gearbox fault diagnosis using deep belief networks”. In: *Measurement science and technology* 28.2 (2016), p. 025010.

[139] Hong Wang et al. “Early Fault Detection of Wind Turbines Based on Operational Condition Clustering and Optimized Deep Belief Network Modeling”. In: *Energies* 12.6 (2019). ISSN: 1996-1073. DOI: 10.3390/en12060984. URL: https://www.mdpi.com/1996-1073/12/6/984.

[140] Xiafei Long et al. “Wind Turbine Anomaly Identification Based on Improved Deep Belief Network with SCADA Data”. In: *Mathematical Problems in Engineering* 2021.1 (2021), p. 8810045. DOI: https://doi.org/10.1155/2021/8810045. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/8810045.

[141] Somayeh Bakhtiari Ramezani et al. “A Survey of HMM-based Algorithms in Machinery Fault Prediction”. In: *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2021, pp. 1–9. DOI: 10.1109/SSCI50451.2021.9659838.

[142] Giduthuri Sateesh Babu, Peilin Zhao, and Xiao-Li Li. “Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life”. In: *Database Systems for Advanced Applications*. Ed. by Shamkant B. Navathe et al. Cham: Springer International Publishing, 2016, pp. 214–228. ISBN: 978-3-319-32025-0.

[143] Xiang Li, Qian Ding, and Jian-Qiao Sun. "Remaining useful life estimation in prognostics using deep convolution neural networks". In: *Reliability Engineering & System Safety* 172 (2018), pp. 1–11. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2017.11.021. URL: https://www.sciencedirect.com/science/article/pii/S0951832017307779.

[144] Markus Ulmer et al. "Early fault detection based on wind turbine scada data using convolutional neural networks". In: *5th European Conference of the Prognostics and Health Management Society, Virtual Conference, 27-31 July 2020*. Vol. 5. 1. PHM Society. 2020.

[145] Ling Xiang et al. "Fault detection of wind turbine based on SCADA data analysis using CNN and LSTM with attention mechanism". In: *Measurement* 175 (2021), p. 109094. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2021.109094. URL: https://www.sciencedirect.com/science/article/pii/S026322412100124X.

[146] Yu Sun et al. "CNN–LSTM–AM: A power prediction model for offshore wind turbines". In: *Ocean Engineering* 301 (2024), p. 117598. ISSN: 0029-8018. DOI: https://doi.org/10.1016/j.oceaneng.2024.117598. URL: https://www.sciencedirect.com/science/article/pii/S0029801824009351.

[147] Shuai Zheng et al. "Long Short-Term Memory Network for Remaining Useful Life estimation". In: *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*. 2017, pp. 88–95. DOI: 10.1109/ICPHM.2017.7998311.

[148] Che-Sheng Hsu and Jehn-Ruey Jiang. "Remaining useful life estimation using long short-term memory deep learning". In: *2018 IEEE International Conference on Applied System Invention (ICASI)*. 2018, pp. 58–61. DOI: 10.1109/ICASI.2018.8394326.

[149] Xinyun Zhang et al. "Remaining Useful Life Estimation Based on a New Convolutional and Recurrent Neural Network". In: *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*. 2019, pp. 317–322. DOI: 10.1109/COASE.2019.8843078.

[150] Zhewen Niu et al. "Wind power forecasting using attention-based gated recurrent unit network". In: *Energy* 196 (2020), p. 117081.

[151] Shuo Zhang, Emma Robinson, and Malabika Basu. "Wind power forecasting based on a novel gated recurrent neural network model". In: *Wind Energy and Engineering Research* 1 (2024), p. 100004. ISSN: 2950-3604. DOI: https://doi.org/10.1016/j.weer.2024.100004. URL: https://www.sciencedirect.com/science/article/pii/S2950360424000044.

[152] Wang Jiujian et al. "Remaining Useful Life Estimation in Prognostics Using Deep Bidirectional LSTM Neural Network". In: Oct. 2018, pp. 1037–1042. DOI: 10.1109/PHM-Chongqing.2018.00184.

[153] Cheng-Geng Huang, Hong-Zhong Huang, and Yan-Feng Li. "A Bidirectional LSTM Prognostics Method Under Multiple Operational Conditions". In: *IEEE Transactions on Industrial Electronics* 66.11 (2019), pp. 8792–8802. DOI: 10.1109/TIE.2019.2891463.

[154] Yizhe Shen et al. "Remaining useful life prediction of rolling bearing based on multi-head attention embedded Bi-LSTM network". In: *Measurement* 202 (2022), p. 111803. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2022.111803. URL: https://www.sciencedirect.com/science/article/pii/S0263224122010041.

[155] Ahmed Zakariae Hinchi and Mohamed Tkiouat. "Rolling element bearing remaining useful life estimation based on a convolutional long-short-term memory network". In: *Procedia Computer Science* 127 (2018), pp. 123–132.

[156] Jun Wu et al. "Data-driven remaining useful life prediction via multiple sensor signals and deep long short-term memory neural network". In: *ISA transactions* 97 (2020), pp. 241–250.

[157] Yuanhang Chen et al. "A novel deep learning method based on attention mechanism for bearing remaining useful life prediction". In: *Applied Soft Computing* 86 (2020), p. 105919. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2019.105919. URL: https://www.sciencedirect.com/science/article/pii/S1568494619307008.

[158] Zhiqiang Xu et al. "Global attention mechanism based deep learning for remaining useful life prediction of aero-engine". In: *Measurement* 217 (2023), p. 113098. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2023.113098. URL: https://www.sciencedirect.com/science/article/pii/S0263224123006620.

[159] Chengying Zhao et al. “A novel remaining useful life prediction method based on gated attention mechanism capsule neural network”. In: *Measurement* 189 (2022), p. 110637. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2021.110637. URL: https://www.sciencedirect.com/science/article/pii/S0263224121015074.

[160] Derya Soydaner. “Attention mechanism in neural networks: where it comes and where it goes”. In: *Neural Computing and Applications* 34.16 (2022), pp. 13371–13385.

[161] Zhaoyang Niu, Guoqiang Zhong, and Hui Yu. “A review on the attention mechanism of deep learning”. In: *Neurocomputing* 452 (2021), pp. 48–62. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.03.091. URL: https://www.sciencedirect.com/science/article/pii/S092523122100477X.

[162] A Vaswani. “Attention is all you need”. In: *Advances in Neural Information Processing Systems* (2017).

[163] Gavneet Singh Chadha et al. “Shared Temporal Attention Transformer for Remaining Useful Lifetime Estimation”. In: *IEEE Access* 10 (2022), pp. 74244–74258. DOI: 10.1109/ACCESS.2022.3187702.

[164] Zhizheng Zhang, Wen Song, and Qiqiang Li. “Dual-Aspect Self-Attention Based on Transformer for Remaining Useful Life Prediction”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–11. DOI: 10.1109/TIM.2022.3160561.

[165] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. “Tranad: Deep transformer networks for anomaly detection in multivariate time series data”. In: *arXiv preprint arXiv:2201.07284* (2022).

[166] Daoquan Chen, Weicong Hong, and Xiuze Zhou. “Transformer Network for Remaining Useful Life Prediction of Lithium-Ion Batteries”. In: *IEEE Access* 10 (2022), pp. 19621–19628. DOI: 10.1109/ACCESS.2022.3151975.

[167] Zhengyang Fan, Wanru Li, and Kuo-Chu Chang. “A Bidirectional Long Short-Term Memory Autoencoder Transformer for Remaining Useful Life Estimation”. In: *Mathematics* 11.24 (2023). ISSN: 2227-7390. DOI: 10.3390/math11244972. URL: https://www.mdpi.com/2227-7390/11/24/4972.

[168] Junjie Wang, Xiaofeng Hu, and Yuwang Yang. “Remaining Useful Life Prediction Method Based on Conv-Transformer Variational Autoencoder”. In: *2024 5th International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*. 2024, pp. 900–908. DOI: 10.1109/AIEA62095.2024.10692508.

[169] Xanthi Bampoula, Nikolaos Nikolakis, and Kosmas Alexopoulos. “Condition Monitoring and Predictive Maintenance of Assets in Manufacturing Using LSTM-Autoencoders and Transformer Encoders”. In: *Sensors* 24.10 (2024). ISSN: 1424-8220. DOI: 10.3390/s24103215. URL: https://www.mdpi.com/1424-8220/24/10/3215.

[170] Lei Wang et al. “M2TNet: Multi-modal multi-task Transformer network for ultra-short-term wind power multi-step forecasting”. In: *Energy Reports* 8 (2022), pp. 7628–7642.

[171] Kai Qu et al. “Short-term forecasting for multiple wind farms based on transformer model”. In: *Energy Reports* 8 (2022), pp. 483–490.

[172] Haoyi Xiao, Xiaoxia He, and Chunli Li. “Probability Density Forecasting of Wind Power Based on Transformer Network with Expectile Regression and Kernel Density Estimation”. In: *Electronics* 12.5 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12051187. URL: https://www.mdpi.com/2079-9292/12/5/1187.

[173] Zi-han Zhao et al. “A state detection method of offshore wind turbines’ gearbox bearing based on the transformer and GRU”. In: *Measurement Science and Technology* 35.2 (Nov. 2023), p. 025903. DOI: 10.1088/1361-6501/ad0956. URL: https://dx.doi.org/10.1088/1361-6501/ad0956.

[174] Minglei Zheng et al. “Semi-supervised multivariate time series anomaly detection for wind turbines using generator SCADA data”. In: *Reliability Engineering & System Safety* 235 (2023), p. 109235. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2023.109235. URL: https://www.sciencedirect.com/science/article/pii/S0951832023001503.

[175] Ying Peng, Ming Dong, and Ming Jian Zuo. “Current status of machine prognostics in condition-based maintenance: a review”. In: *The International Journal of Advanced Manufacturing Technology* 50 (2010), pp. 297–313.

[176] Sarfaraz Natha. “A Systematic Review of Anomaly detection using Machine and Deep Learning Techniques”. In: *Quaid-e-Awam University Research Journal of Engineering, Science & Technology* 20 (June 2022), pp. 83–94. DOI: 10.52584/QRJ.2001.11.

[177] Georg Helbing and Matthias Ritter. “Deep Learning for fault detection in wind turbines”. In: *Renewable and Sustainable Energy Reviews* 98 (2018), pp. 189–198. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2018.09.012. URL: https://www.sciencedirect.com/science/article/pii/S1364032118306610.

[178] Yongjian Xue and Pierre Beauseroy. “Transfer learning for one class SVM adaptation to limited data distribution change”. In: *Pattern Recognition Letters* 100 (2017), pp. 117–123.

[179] Renxiang Chen et al. “Intelligent fault diagnosis method of planetary gearboxes based on convolution neural network and discrete wavelet transform”. In: *Computers in industry* 106 (2019), pp. 48–59.

[180] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. “A systematic review on imbalanced data challenges in machine learning: Applications and solutions”. In: *ACM computing surveys (CSUR)* 52.4 (2019), pp. 1–36.

[181] Cristian Velandia-Cardenas, Yolanda Vidal, and Francesc Pozo. “Wind Turbine Fault Detection Using Highly Imbalanced Real SCADA Data”. In: *Energies* 14.6 (2021). ISSN: 1996-1073. DOI: 10.3390/en14061728. URL: https://www.mdpi.com/1996-1073/14/6/1728.

[182] Tongzhou Wang and Phillip Isola. “Understanding contrastive representation learning through alignment and uniformity on the hypersphere”. In: *International conference on machine learning*. PMLR. 2020, pp. 9929–9939.

[183] Shilin Sun et al. “Matching contrastive learning: An effective and intelligent method for wind turbine fault diagnosis with imbalanced SCADA data”. In: *Expert Systems with Applications* 223 (2023), p. 119891. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2023.119891. URL: https://www.sciencedirect.com/science/article/pii/S0957417423003925.

[184] Zixuan Wang et al. “An imbalanced semi-supervised wind turbine blade icing detection method based on contrastive learning”. In: *Renewable Energy* 212 (2023), pp. 251–262.

[185] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. “Unsupervised Deep Embedding for Clustering Analysis”. In: *CoRR* abs/1511.06335 (2015). arXiv: 1511.06335. URL: http://arxiv.org/abs/1511.06335.

[186] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. “Contrastive Representation Learning: A Framework and Review”. In: *IEEE Access* 8 (2020), pp. 193907–193934. DOI: 10.1109/ACCESS.2020.3031549.

[187] Dino Ienco and Roberto Interdonato. “Deep Multivariate Time Series Embedding Clustering via Attentive-Gated Autoencoder”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Hady W. Lauw et al. Cham: Springer International Publishing, 2020, pp. 318–329. ISBN: 978-3-030-47426-3.

[188] Baptiste Lafabregue et al. “End-to-end deep representation learning for time series clustering: a comparative study”. In: *Data Mining and Knowledge Discovery* 36.1 (2022), pp. 29–81.

[189] Mulin Chen, Bo-Cheng Wang, and Xuelong Li. “Deep Contrastive Graph Learning with Clustering-Oriented Guidance”. In: *International conference on machine learning*. Vol. 38. 10. Association for the Advancement of Artificial Intelligence, 2024, pp. 11364–11372. DOI: 10.1609/aaai.v38i10.29016.

[190] Serkan Kiranyaz et al. “1D convolutional neural networks and applications: A survey”. In: *Mechanical Systems and Signal Processing* 151 (2021), p. 107398. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107398. URL: https://www.sciencedirect.com/science/article/pii/S0888327020307846.

[191] W Wang and A H Christer. “Towards a general condition based maintenance model for a stochastic dynamic system”. In: *Journal of the Operational Research Society* 51.2 (2000), pp. 145–155. DOI: 10.1057/palgrave.jors.2600863. eprint: https://doi.org/10.1057/palgrave.jors.2600863. URL: https://doi.org/10.1057/palgrave.jors.2600863.

[192] GAM Van Kuik et al. “Long-term research challenges in wind energy–a research agenda by the European Academy of Wind Energy”. In: *Wind energy science* 1.1 (2016), pp. 1–39.

[193] Weiwen Peng, Zhi-Sheng Ye, and Nan Chen. "Bayesian deep-learning-based health prognostics toward prognostics uncertainty". In: *IEEE Transactions on Industrial Electronics* 67.3 (2019), pp. 2283–2293.

[194] Moloud Abdar et al. "A review of uncertainty quantification in deep learning: Techniques, applications and challenges". In: *Information Fusion* 76 (2021), pp. 243–297. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.05.008. URL: https://www.sciencedirect.com/science/article/pii/S1566253521001081.

[195] Mingxin Li et al. "Influence of uncertainty on performance of opportunistic maintenance strategy for offshore wind farms". In: *OCEANS 2021: San Diego–Porto*. IEEE. 2021, pp. 1–10.

[196] Milad Rezamand et al. "Critical Wind Turbine Components Prognostics: A Comprehensive Review". In: *IEEE Transactions on Instrumentation and Measurement* 69.12 (2020), pp. 9306–9328. DOI: 10.1109/TIM.2020.3030165.

[197] Jakob Gawlikowski et al. "A survey of uncertainty in deep neural networks". In: *Artificial Intelligence Review* 56.Suppl 1 (2023), pp. 1513–1589.

[198] Dengshan Huang et al. "Bayesian Neural Network Based Method of Remaining Useful Life Prediction and Uncertainty Quantification for Aircraft Engine". In: *2020 IEEE International Conference on Prognostics and Health Management (ICPHM)*. 2020, pp. 1–8. DOI: 10.1109/ICPHM49022.2020.9187044.

[199] Huanjie Wang, Xiwei Bai, and Jie Tan. "Uncertainty Quantification of Bearing Remaining Useful Life Based on Convolutional Neural Network". In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 2893–2900. DOI: 10.1109/SSCI47803.2020.9308463.

[200] Ganjour Mazaev, Guillaume Crevecoeur, and Sofie Van Hoecke. "Bayesian convolutional neural networks for remaining useful life prognostics of solenoid valves with uncertainty estimations". In: *IEEE Transactions on Industrial Informatics* 17.12 (2021), pp. 8418–8428.

[201] Sheng Hong and Zheng Zhou. "Application of Gaussian process regression for bearing degradation assessment". In: *2012 6th International Conference on New Trends in Information Science, Service Science and Data Mining (ISSDM2012)*. IEEE. 2012, pp. 644–648.

[202] Jürgen Herp et al. "Bayesian state prediction of wind turbine bearing failure". In: *Renewable Energy* 116 (2018), pp. 164–172.

[203] Milad Rezamand et al. "An Integrated Feature-Based Failure Prognosis Method for Wind Turbine Bearings". In: *IEEE/ASME Transactions on Mechatronics* 25.3 (2020), pp. 1468–1478. DOI: 10.1109/TMECH.2020.2978136.

[204] n.d. *Numerical data: Normalization*. Accessed on 24.10.2024. n.d. URL: https://developers.google.com/machine-learning/crash-course/numerical-data/normalization.

[205] n.d. *What is the MinMax Scaler?* Accessed on 24.10.2024. n.d. URL: https://databasecamp.de/en/ml/minmax-scaler-en.

[206] Davide Manna. "HEALTH MONITORING FOR WIND TURBINES–DATASETS PROCESSING AND DEVELOPMENT OF RUL PROGNOSTICS". PhD thesis. Politecnico di Torino, 2023.

[207] H.D. Nguyen et al. "Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management". In: *International Journal of Information Management* 57 (2021), p. 102282. ISSN: 0268-4012. DOI: https://doi.org/10.1016/j.ijinfomgt.2020.102282. URL: https://www.sciencedirect.com/science/article/pii/S026840122031481X.

[208] n.d. *RobustScaler*. Accessed on 31.10.2024. n.d. URL: https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.RobustScaler.html#sklearn.preprocessing.RobustScaler.

[209] Amgad Muneer et al. "Data-Driven Deep Learning-Based Attention Mechanism for Remaining Useful Life Prediction: Case Study Application to Turbofan Engine Analysis". In: *Electronics* 10.20 (2021). ISSN: 2079-9292. DOI: 10.3390/electronics10202453. URL: https://www.mdpi.com/2079-9292/10/20/2453.

[210] Yan Liu et al. "A DLSTM-Network-Based Approach for Mechanical Remaining Useful Life Prediction". In: *Sensors* 22.15 (2022). ISSN: 1424-8220. DOI: 10.3390/s22155680. URL: https://www.mdpi.com/1424-8220/22/15/5680.

[211] Julius Venskus, Povilas Treigys, and Jurgita Markevičiūtė. “Unsupervised marine vessel trajectory prediction using LSTM network and wild bootstrapping techniques”. In: *Nonlinear Analysis: Modelling and Control* 26 (July 2021), pp. 718–737. DOI: 10.15388/namc.2021.26.23056.

[212] Bahrudin Hrnjica and Ognjen Bonacci. “Lake Level Prediction using Feed Forward and Recurrent Neural Networks”. In: *Water Resources Management* (May 2019), pp. 1–14. DOI: 10.1007/s11269-019-02255-2.

[213] Shuangshuang Chen and Wei Guo. “Auto-Encoders in Deep Learning—A Review with New Perspectives”. In: *Mathematics* 11.8 (2023). ISSN: 2227-7390. DOI: 10.3390/math11081777. URL: https://www.mdpi.com/2227-7390/11/8/1777.

[214] Yoshua Bengio, Aaron Courville, and Pascal Vincent. “Representation learning: A review and new perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[215] Diehao Kong and Xuefeng Yan. “Industrial process deep feature representation by regularization strategy autoencoders for process monitoring”. In: *Measurement Science and Technology* 31.2 (2019), p. 025104.

[216] Jianbo Yu and Xuefeng Yan. “A new deep model based on the stacked autoencoder with intensified iterative learning style for industrial fault detection”. In: *Process Safety and Environmental Protection* 153 (2021), pp. 47–59.

[217] Pramod Bangalore and Lina Bertling Tjernberg. “An artificial neural network approach for early fault detection of gearbox bearings”. In: *IEEE Transactions on Smart Grid* 6.2 (2015), pp. 980–987.

[218] Génesis Vásquez-Rodríguez and Jorge Maldonado-Correa. “Anomaly-based fault detection in wind turbines using unsupervised learning: a comparative study.” In: *IOP Conference Series: Earth and Environmental Science* 1370.1 (July 2024), p. 012005. DOI: 10.1088/1755-1315/1370/1/012005. URL: https://dx.doi.org/10.1088/1755-1315/1370/1/012005.

[219] Xavier Chesterman et al. “Condition monitoring of wind turbines and extraction of healthy training data using an ensemble of advanced statistical anomaly detection models”. In: *Annual Conference of the PHM Society*. Vol. 13. 1. 2021.

[220] Yirong Liu, Zidong Wu, and Xiaoli Wang. “Research on Fault Diagnosis of Wind Turbine Based on SCADA Data”. In: *IEEE Access* 8 (2020), pp. 185557–185569. DOI: 10.1109/ACCESS.2020.3029435.

[221] Marcia L. Baptista, Kai Goebel, and Elsa M.P. Henriques. “Relation between prognostics predictor evaluation metrics and local interpretability SHAP values”. In: *Artificial Intelligence* 306 (2022), p. 103667. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2022.103667. URL: https://www.sciencedirect.com/science/article/pii/S0004370222000078.

[222] Lin Huang et al. “An Unsupervised Machine Learning Approach for Monitoring Data Fusion and Health Indicator Construction”. In: *Sensors* 23.16 (2023). ISSN: 1424-8220. DOI: 10.3390/s23167239. URL: https://www.mdpi.com/1424-8220/23/16/7239.

[223] Jamie Coble and J Wesley Hines. “Identifying optimal prognostic parameters from data: a genetic algorithms approach”. In: *Annual Conference of the PHM Society*. Vol. 1. 1. 2009.

[224] Omar Serghini et al. “1-D Convolutional Neural Network-Based Models for Cooperative Spectrum Sensing”. In: *Future Internet* 16.1 (2024). ISSN: 1999-5903. DOI: 10.3390/fi16010014. URL: https://www.mdpi.com/1999-5903/16/1/14.

[225] Ilaria Cacciari and Anedio Ranfagni. “Hands-On Fundamentals of 1D Convolutional Neural Networks—A Tutorial for Beginner Users”. In: *Applied Sciences* 14.18 (2024). ISSN: 2076-3417. DOI: 10.3390/app14188500. URL: https://www.mdpi.com/2076-3417/14/18/8500.

[226] Jiaju Wu et al. “Aircraft Engine Fault Diagnosis Model Based on 1DCNN-BiLSTM with CBAM”. In: *Sensors* 24.3 (2024), p. 780.

[227] Ashish Jaiswal et al. “A Survey on Contrastive Self-Supervised Learning”. In: *Technologies* 9.1 (2021). ISSN: 2227-7080. DOI: 10.3390/technologies9010002. URL: https://www.mdpi.com/2227-7080/9/1/2.

[228] Kexin Zhang, Rongyao Cai, and Yong Liu. “Industrial Fault Detection using Contrastive Representation Learning on Time-series Data”. In: *IFAC-PapersOnLine* 56.2 (2023). 22nd IFAC World Congress, pp. 3197–3202. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2023.10.1456. URL: https://www.sciencedirect.com/science/article/pii/S2405896323018645.

[229] Jing Wang and Songhe Feng. “Contrastive and view-interaction structure learning for multi-view clustering”. In: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*. 2024, pp. 5055–5063.

[230] Ching-Yao Chuang et al. “Debiased Contrastive Learning”. In: *CoRR* abs/2007.00224 (2020). arXiv: 2007.00224. URL: https://arxiv.org/abs/2007.00224.

[231] Sumit Chopra, Raia Hadsell, and Yann LeCun. “Learning a Similarity Metric Discriminatively, with Application to Face Verification”. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. CVPR '05. USA: IEEE Computer Society, 2005, pp. 539–546. ISBN: 0769523722. DOI: 10.1109/CVPR.2005.202. URL: https://doi.org/10.1109/CVPR.2005.202.

[232] R. Hadsell, S. Chopra, and Y. LeCun. “Dimensionality Reduction by Learning an Invariant Mapping”. In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[233] Aoxue Li et al. “Boosting Few-Shot Learning With Adaptive Margin Loss”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[234] Zhirong Wu et al. “Unsupervised Feature Learning via Non-parametric Instance Discrimination”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3733–3742. DOI: 10.1109/CVPR.2018.00393.

[235] Laurens van der Maaten. “Learning a Parametric Embedding by Preserving Local Structure”. In: *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*. Ed. by David van Dyk and Max Welling. Vol. 5. Proceedings of Machine Learning Research. Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA: PMLR, 16–18 Apr 2009, pp. 384–391. URL: https://proceedings.mlr.press/v5/maaten09a.html.

[236] Yazhou Ren et al. *Deep Density-based Image Clustering*. 2018. arXiv: 1812.04287 [cs.LG]. URL: https://arxiv.org/abs/1812.04287.

[237] Dingsheng Deng. “DBSCAN Clustering Algorithm Based on Density”. In: *2020 7th International Forum on Electrical Engineering and Automation (IFEEA)*. 2020, pp. 949–953. DOI: 10.1109/IFEEA51475.2020.00199.

[238] Guolian Hou, Junjie Wang, and Yuzhen Fan. “Wind power forecasting method of large-scale wind turbine clusters based on DBSCAN clustering and an enhanced hunter-prey optimization algorithm”. In: *Energy Conversion and Management* 307 (2024), p. 118341. ISSN: 0196-8904. DOI: https://doi.org/10.1016/j.enconman.2024.118341. URL: https://www.sciencedirect.com/science/article/pii/S0196890424002826.

[239] Chenlong Feng et al. “Multivariate anomaly detection and early warning framework for wind turbine condition monitoring using SCADA data”. In: *Journal of Energy Engineering* 149.6 (2023), p. 04023040.

[240] Jiayang Liu et al. “Cross-machine deep subdomain adaptation network for wind turbines fault diagnosis”. In: *Mechanical Systems and Signal Processing* 210 (2024), p. 111151. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2024.111151. URL: https://www.sciencedirect.com/science/article/pii/S0888327024000499.

[241] Feiya Lv et al. “Fault diagnosis based on deep learning”. In: *2016 American Control Conference (ACC)*. 2016, pp. 6851–6856. DOI: 10.1109/ACC.2016.7526751.

[242] Laurens Van Der Maaten, Eric O Postma, H Jaap Van Den Herik, et al. “Dimensionality reduction: A comparative review”. In: *Journal of machine learning research* 10.66-71 (2009), p. 13.

[243] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. “A survey of dimensionality reduction techniques”. In: *arXiv preprint arXiv:1403.2877* (2014).

[244] Mohsena Ashraf et al. “A Survey on Dimensionality Reduction Techniques for Time-Series Data”. In: *IEEE Access* 11 (2023), pp. 42909–42923. DOI: 10.1109/ACCESS.2023.3269693.

[245] Ziqian Kong et al. “A contrastive learning framework enhanced by unlabeled samples for remaining useful life prediction”. In: *Reliability Engineering & System Safety* 234 (2023), p. 109163. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2023.109163. URL: https://www.sciencedirect.com/science/article/pii/S0951832023000789.

[246] Navid Mohammadi Foumani et al. “Improving position encoding of transformers for multivariate time series classification”. In: *Data Mining and Knowledge Discovery* 38.1 (Sept. 2023), pp. 22–48. DOI: 10.1007/s10618-023-00948-2.

[247] Xuanqing Liu et al. “Learning to encode position for transformer with continuous dynamical model”. In: *International conference on machine learning*. PMLR. 2020, pp. 6327–6335.

[248] Jacob Devlin. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[249] Samy Bengio et al. *Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks*. 2015. arXiv: 1506.03099 [cs.LG]. URL: https://arxiv.org/abs/1506.03099.

[250] Minhee Kim and Kaibo Liu. “A Bayesian deep learning framework for interval estimation of remaining useful life in complex systems by incorporating general degradation characteristics”. In: *IISE Transactions* 53.3 (2020), pp. 326–340.

[251] Oluwaseyi Ogunfowora and Homayoun Najjaran. “A Transformer-based Framework For Multi-variate Time Series: A Remaining Useful Life Prediction Use Case”. In: *arXiv preprint arXiv:2308.09884* (2023).

[252] Chris Teubert. *CMAPSS Jet Engine Simulated Data*. Accessed on 25.11.2024. 2024. URL: https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6/about_data.

[253] Wei Jiang et al. “A feature-level degradation measurement method for composite health index construction and trend prediction modeling”. In: *Measurement* 206 (2023), p. 112324. ISSN: 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2022.112324. URL: https://www.sciencedirect.com/science/article/pii/S0263224122015202.

[254] Jianhai Yan, Zhen He, and Shuguang He. “A deep learning framework for sensor-equipped machine health indicator construction and remaining useful life prediction”. In: *Computers & Industrial Engineering* 172 (2022), p. 108559. ISSN: 0360-8352. DOI: https://doi.org/10.1016/j.cie.2022.108559. URL: https://www.sciencedirect.com/science/article/pii/S0360835222005629.

[255] Liang Guo et al. “Machinery health indicator construction based on convolutional neural networks considering trend burr”. In: *Neurocomputing* 292 (2018), pp. 142–150. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2018.02.083. URL: https://www.sciencedirect.com/science/article/pii/S0925231218302583.

[256] Lei Xiao et al. “Remaining useful life prediction based on intentional noise injection and feature reconstruction”. In: *Reliability Engineering & System Safety* 215 (2021), p. 107871. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2021.107871. URL: https://www.sciencedirect.com/science/article/pii/S0951832021003902.

[257] Georgios Koutroulis, Belgin Mutlu, and Roman Kern. “Constructing robust health indicators from complex engineered systems via anticausal learning”. In: *Engineering Applications of Artificial Intelligence* 113 (2022), p. 104926. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.104926. URL: https://www.sciencedirect.com/science/article/pii/S0952197622001476.

[258] Lu Liu, Xiao Song, and Zhetao Zhou. “Aircraft engine remaining useful life estimation via a double attention-based data-driven architecture”. In: *Reliability Engineering & System Safety* 221 (2022), p. 108330. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2022.108330. URL: https://www.sciencedirect.com/science/article/pii/S0951832022000102.

[259] Wennian Yu, Il Yong Kim, and Chris Mechefske. “Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme”. In: *Mechanical Systems and Signal Processing* 129 (2019), pp. 764–780. ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2019.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0888327019303061.

[260] Emmanuel Ramasso and Abhinav Saxena. “Performance Benchmarking and Analysis of Prognostic Methods for CMAPSS Datasets.” In: *International Journal of Prognostics and Health Management* 5.2 (2014), pp. 1–15.

[261] *EDP Dataset*. Accessed on 20.09.2024. URL: https://www.edp.com/en/innovation/open-data/data.

[262] Andrew Kusiak and Wenyan Li. “The prediction and diagnosis of wind turbine faults”. In: *Renewable Energy* 36.1 (2011), pp. 16–23. ISSN: 0960-1481. DOI: https://doi.org/10.1016/j.renene.2010.05.014. URL: https://www.sciencedirect.com/science/article/pii/S0960148110002338.

[263] Takuya Akiba et al. *Optuna: A Next-generation Hyperparameter Optimization Framework*. 2019. arXiv: 1907.10902 [cs.LG]. URL: https://arxiv.org/abs/1907.10902.

[264] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

[265] Jason Brownlee. *Deep learning for time series forecasting: predict the future with MLPs, CNNs and LSTMs in Python*. Machine Learning Mastery, 2018.

[266] Jing-Ru C. Cheng Minhee Kim and Kaibo Liu. "An adaptive sensor selection framework for multisensor prognostics". In: *Journal of Quality Technology* 53.5 (2021), pp. 566–585. DOI: 10.1080/00224065.2021.1960934. eprint: https://doi.org/10.1080/00224065.2021.1960934. URL: https://doi.org/10.1080/00224065.2021.1960934.

[267] Maryna Garan, Khaoula Tidriri, and Iaroslav Kovalenko. "A Data-Centric Machine Learning Methodology: Application on Predictive Maintenance of Wind Turbines". In: *Energies* 15.3 (2022). ISSN: 1996-1073. DOI: 10.3390/en15030826. URL: https://www.mdpi.com/1996-1073/15/3/826.

[268] Cyriana M.A. Roelofs et al. "Autoencoder-based anomaly root cause analysis for wind turbines". In: *Energy and AI* 4 (2021), p. 100065. ISSN: 2666-5468. DOI: https://doi.org/10.1016/j.egyai.2021.100065. URL: https://www.sciencedirect.com/science/article/pii/S2666546821000197.

[269] Sungheon Park and Nojun Kwak. "Analysis on the Dropout Effect in Convolutional Neural Networks". In: Mar. 2017, pp. 189–204. ISBN: 978-3-319-54183-9. DOI: 10.1007/978-3-319-54184-6_12.

[270] Xiongjie Jia et al. "Condition monitoring and performance forecasting of wind turbines based on denoising autoencoder and novel convolutional neural networks". In: *Energy Reports* 7 (2021), pp. 6354–6365. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2021.09.080. URL: https://www.sciencedirect.com/science/article/pii/S2352484721008854.

[271] Ming-De Liu, Lin Ding, and Yu-Long Bai. "Application of hybrid model based on empirical mode decomposition, novel recurrent neural networks and the ARIMA to wind speed prediction". In: *Energy Conversion and Management* 233 (2021), p. 113917.

[272] *Wind Turbine SCADA open data*. Accessed on 20.09.2024. GitHub. URL: https://github.com/sltzgs/Wind_Turbine_SCADA_open_data/tree/main.

[273] Nina Effenberger and Nicole Ludwig. "A collection and categorization of open-source wind and wind power datasets". In: *Wind Energy* 25.10 (2022), pp. 1659–1683. DOI: https://doi.org/10.1002/we.2766. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.2766. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/we.2766.

[274] *Beberibe Dataset*. Accessed on 20.09.2024. URL: https://zenodo.org/records/1475197#.Y3KRoPfMLIU.

[275] *DSWE Dataset*. Accessed on 20.09.2024. URL: https://aml.engr.tamu.edu/book-dswe/dswe-datasets/.

[276] *PCWG*. Accessed on 20.09.2024. URL: https://pcwg.org/.

[277] Charlie Plumley. *Kelmarsh Wind Farm Data*. Accessed on 20.09.2024. 2022. URL: https://zenodo.org/record/5841834#.YgpBQ_so-V7.

[278] Charlie Plumley. *Penmanshiel Wind Farm Data*. Accessed on 20.09.2024. 2022. URL: https://doi.org/10.5281/zenodo.8253010.

[279] *Ørsted Anholt Offshore Wind Farm*. Accessed on 20.09.2024. URL: https://orsted.com/en/our-business/offshore-wind/offshore-operational-data.

[280] Welson Bassi, Alcantaro Lemes Rodrigues, and Ildo Luis Sauer. "Dataset on SCADA Data of an Urban Small Wind Turbine Operation in São Paulo, Brazil". In: *Data* 8.3 (2023). ISSN: 2306-5729. DOI: 10.3390/data8030052. URL: https://www.mdpi.com/2306-5729/8/3/52.

[281] *Vestas V52 Wind Turbine, 10-minute SCADA Data*. Accessed on 20.09.2024. URL: https://data.mendeley.com/datasets/tm988rs48k/2.

[282] *Kaggle Wind Turbine Scada Dataset*. Accessed on 20.09.2024. URL: https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset.

[283] *DelftBlue Documentation*. 2024. URL: https://doc.dhpc.tudelft.nl/delftblue/.