ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

INGEGNERIA ELETTRONICA, TELECOMUNICAZIONI E TECNOLOGIE DELL'INFORMAZIONE

Ciclo 38

Settore Concorsuale: 09/E3 - ELETTRONICA

Settore Scientifico Disciplinare: ING-INF/01 - ELETTRONICA

DATA-DRIVEN APPROACHES TO PROGNOSTICS AND HEALTH MANAGEMENT FOR INTELLIGENT MAINTENANCE

Presentata da: Silvia Onofri

Coordinatore Dottorato

Davide Dardari

Supervisore

Riccardo Rovatti

Co-supervisore

Mauro Mangia

Esame finale anno 2026

Abstract

Modern Cyber-Physical Systems (CPSs) tightly integrate physical processes with digital control and communication layers, operating in increasingly complex, interconnected, and safety-critical environments. Ensuring their reliability and availability requires advanced monitoring and maintenance strategies capable of predicting faults and managing degradation before failures occur. Within this context, Prognostics and Health Management (PHM) has emerged as a comprehensive framework that combines sensing, data processing, diagnosis, and life prediction into intelligent decision support for condition-based and predictive maintenance planning.

This dissertation explores data-driven methodologies to enhance the PHM pipeline across both prognostic and diagnostic dimensions. The first part focuses on Remaining Useful Life (RUL) estimation, introducing autoencoder-based feature extraction techniques for health indicator generation, along with multi-class, similarity-based, and degradation-based models for RUL estimation. These approaches are further extended to edge-level implementations to evaluate feasibility in resource-constrained environments. The second part addresses anomaly detection as a core diagnostic task, proposing a general framework for systematic benchmarking of detectors through synthetic yet realistic anomaly generation on time-series data.

Combining theoretical modeling, numerical validation, and implementation studies, this work advances the development of robust PHM methodologies. The proposed frameworks offer adaptive and computationally efficient tools to support intelligent, data-driven maintenance in modern monitoring applications.

Acknowledgements

A special thanks goes to Yvan, my greatest support and the patient co-traveler of this PhD adventure. Thank you for helping me navigate countless deadlines with calm, and for keeping both me and every submission on track. Following our PhDs in different groups has made the ride both entertaining and occasionally challenging, but never boring.

My sincere gratitude goes to my advisors, Riccardo and Mauro, for their sharp insights, and for somehow managing to keep me motivated throughout this journey.

To my long-time "lab colleagues", Filippo, Andriy, Alex, Clemens, Lorenzo and Livia, thank you for making the lab a much more enjoyable place to work in. Coming in every morning was easier knowing that the day would always be brightened by memes, questionable jokes, and philosophical discussions of dubious academic value.

I would also like to thank the people at Huawei, whose support and technical expertise shaped my professional growth and who kindly ensured I would never again underestimate the meaning of "industrial deadlines".

Finally, to my family, thank you for supporting me, trusting me, and pretending (with admirable consistency) to know what I actually do for a living.

Contents

List of Figures

List of Tables

Acronyms

ACC Accelerometer. 69, 77, 81, 83, 87, 88

AD Anomaly detection. 59, 67, 71

AE Autoencoder. 9, 16–23, 25–28, 31, 33, 34, 44, 45, 47, 50, 54, 64

AR Autoregressive. 63

AUC Area Under the Curve. 65, 77

CBM Condition-Based Maintenance. 2

CNN Convolutional Neural Network. 9, 11, 40

CPS Cyber-Physical System. 2, 4, 10, 39

DL Deep Learning. 10, 39, 40, 64, 77

ECG Electrocardiogram. 60, 69, 77, 78, 81–83, 87, 88, 90, 92, 95

GD Gaussian Distribution. 63, 88, 89, 91

GNN Gaussian Narrowband Noise. 73, 88, 95

GWN Gaussian White Noise. 73, 83, 88, 89, 92, 95

HI Health Index. 7, 9, 16, 23, 25–32, 36, 40–44, 47, 50, 51, 53–55

IF Isolation Forest. 64

LOF Local Outlier Factor. 64, 88, 90

LSTM Long Short-Term Memory. 11, 26, 40

MCU Microcontroller Unit. 3, 40, 41

MD Mahalanobis Distance. 63, 83, 88, 90

ML Machine Learning. 9, 10, 39, 64, 88

MSE Mean Squared Error. 18, 27, 33, 45, 47

NN Neural Network. 40, 42, 43, 45, 46, 50, 64

OC One-Class Support Vector Machine. 64

PCA Principal Component Analysis. 9, 16, 62, 64, 88, 90, 91

PdM Predictive Maintenance. 2, 16, 51

PHM Prognostics and Health Management. 2, 3, 7, 9, 15, 39, 59, 67

RMSE Root Mean Square Error. 12, 34, 35, 45, 46, 48

RNN Recurrent Neural Network. 9, 11

ROC Receiver Operating Characteristic. 65, 77

RUL Remaining Useful Life. 2, 3, 7–11, 15–18, 25–27, 29, 35–37, 39–45, 48, 53

SPE Squared Prediction Error. 62, 83, 88

SVM Support Vector Machine. 11

TV Total Variation. 77, 91

ZC Zero Crossing. 77, 83, 90–92

List of Symbols

$\mathbb{R}$	Set of real numbers
tr	Trace operator
$\cdot^\top$	Transposition operator
$\mathrm{diag}(\cdot)$	Diagonal matrix with argument on the diagonal
$\lvert \cdot \rvert$	Absolute value
$\lVert \cdot \rVert$	Norm
$\boldsymbol{I}_n$	Identity matrix with dimension n $\times$ n
$\boldsymbol{\Sigma}$	Covariance matrix
f	Probability density function
$\mathbb{P}[\cdot]$	Probability of an event
$\mathbf{E}[\cdot]$	Expectation operator

Note: Notation for data is defined locally within each chapter in accordance with the respective experimental framework.

Introduction

In modern industry, the reliable operation of production systems depends on all components maintaining a healthy operational state. Each element must operate efficiently to ensure the overall performance and reliability of the process. However, the increasing automation and interconnection of machines, production lines, and infrastructures have significantly raised system complexity. In such interconnected environments, even small undetected defects can propagate across subsystems, triggering cascading effects, production losses, and potential safety hazards [2]. Regardless of how good the design or manufacturing quality may be, physical deterioration and performance degradation are inevitable over time [3]. Consequently, ensuring system reliability, safety, and availability has emerged as a key objective for industrial operators [3], leading to an increasing emphasis on effective maintenance strategies.

Traditionally, industrial maintenance has relied on two main paradigms: corrective (or reactive) maintenance, where repair or replacement occurs only after a fault has taken place, and preventive (or scheduled) maintenance, where interventions are performed periodically based on predefined maintenance plans tailored to the specific characteristics and failure behaviour of the equipment [4, 5, 6]. While these strategies have been widely adopted, both have evident limitations. Corrective maintenance can lead to costly downtime and safety risks, while preventive maintenance, though safer, may result in unnecessary part replacements and inefficient use of resources.

In recent years, however, the industrial world has undergone a significant transformation driven by the digitalization of production systems, an evolution often referred to as Industry 4.0 [7]. The introduction of advanced sensing, communication, and data-processing technologies has enabled continuous monitoring of machines and processes. Modern machines are equipped with numerous embedded sensors, measuring temperature, vibration, current, pressure, and other physical quantities, that collect large amounts of data in real time, providing information on the operational state of each component [4, 8].

The integration of these sensing and computational capabilities has led to the emergence of industrial Cyber-Physical Systems (CPSs) [9, 10, 7], where physical processes are tightly coupled with digital control and communication layers. Within such systems, the continuous exchange of information between sensors, actuators, and computational units enables real-time monitoring, fault detection, and adaptive decision-making. This paradigm shift has paved the way for new, data-driven approaches to maintenance that move beyond fixed schedules and reactive strategies.

The availability of continuous monitoring data has laid the foundation for Condition-Based Maintenance (CBM) and Predictive Maintenance (PdM). In CBM, maintenance decisions are guided by the measured condition of the system rather than by predefined time intervals [3]. PdM builds upon the same principles but incorporates prognostic capabilities to forecast the future health of the system. Rather than merely detecting degradation and anomalies, PdM uses models, often data-driven or physics-based, to estimate the Remaining Useful Life (RUL) of components and to plan interventions before the condition reaches a critical threshold [4, 11, 8].

These developments converge in the broader discipline of Prognostics and Health Management (PHM), which provides a unified framework that integrates sensing, data analysis, diagnosis, and prognosis to support intelligent maintenance decisions. PHM allows both the detection of current faults and the prediction of future failures, offering the foundation for data-driven, condition-based, and predictive maintenance in complex industrial environments [4, 8, 12].

- **Diagnosis:** Diagnosis deals with the detection, isolation, and identification of faults when an abnormal condition occurs, aiming to detect abnormal behaviours, locate the faulty component, and determine the nature and root cause of the problem [3]. Within the PHM framework, diagnosis represents a core element of CBM, typically following the *anomaly detection* stage [4].

- **Prognosis:** Prognosis focuses on predicting the future health of a system and estimating the RUL of its components [8, 12, 13]. It can leverage diagnostic information, such as fault type, location, and severity, to model how the detected degradation will evolve over time. Within the PHM framework, prognosis constitutes the foundation of PdM, enabling maintenance actions to be planned proactively, minimizing unplanned downtime and optimizing resource utilization [7, 4].

As illustrated in Fig. 0.1, within the PHM framework, diagnosis and prognosis represent distinct yet tightly coupled processes that jointly support intelligent maintenance of complex industrial systems.

Reliable models are fundamental to reduce unplanned downtime, improve system

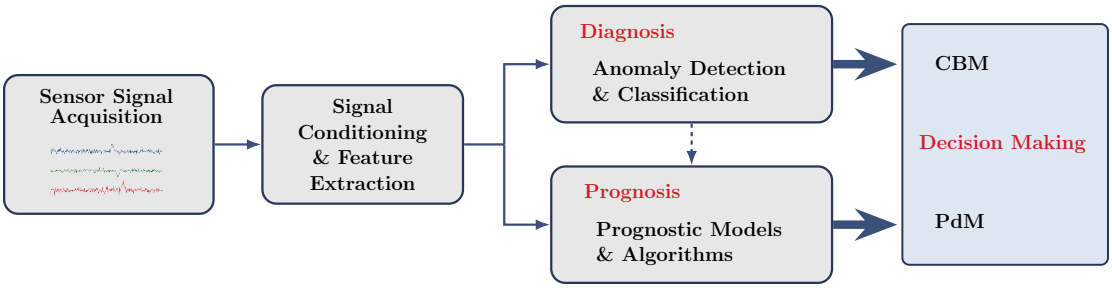

Figure 0.1: Schematic overview of the Prognostics and Health Management (PHM) framework, illustrating the integration of signal acquisition and processing, diagnosis, and prognosis to support condition-based and predictive maintenance decisions.

availability, and optimize maintenance planning. However, modern industrial assets are equipped with a large number of heterogeneous sensors generating high-frequency data streams, which must be processed and interpreted in real time. This massive amount of information increases the complexity of data acquisition, fusion, and storage, and poses significant challenges for computational efficiency, especially when analysis needs to be performed directly on embedded or edge devices with limited resources.

Based on the concepts introduced above, this dissertation focuses on two fundamental aspects of PHM: (i) the prediction of system degradation (*prognosis*), and (ii) the detection of abnormal behaviours (*anomaly detection*).

These two research directions structure the thesis into two main parts:

Part I of the thesis is devoted to *Prognostics and Remaining Useful Life Estimation*, addressing various aspects of RUL prediction algorithms, from health indicator design to edge-level deployment. **Chapter 1** introduces the fundamental concepts of prognostics, including the definition of time-to-failure, degradation modelling, and the design of health indicators for RUL estimation. Building on these concepts, **Chapter 2** focuses on feature extraction and health index formulation as indicators of the health state of a satellite system, comparing data-driven approaches with traditional methods. **Chapter 3** presents a data-driven, similarity-based approach for the RUL estimation of turbofan engines, introducing an intermediate classification step that enhances state-of-the-art techniques. Finally, **Chapter 4** extends the proposed RUL estimation framework by including an evaluation of its implementation on a Microcontroller Unit (MCU).

Part II focuses on *Anomaly Detection*, which represents the first step of the diagnostic process and a key enabler for early fault identification. **Chapter 5** introduces the main concepts of Anomaly Detection, describing the mathematical models and

outlining the algorithms that will be adopted throughout the dissertation. **Chapter 6** presents a comprehensive framework for benchmarking anomaly detection algorithms that operate on time-series data. The core of this framework is an extensive collection of anomaly models designed to reproduce the diverse effects that real-world faults can impose on signals representing normal system behavior. These models enable the synthetic injection of realistic disturbances into healthy data, facilitating controlled and repeatable testing conditions. The proposed approach is validated through two monitoring scenarios, in which multiple anomaly detection techniques are evaluated and compared in terms of robustness and effectiveness.

Together, these two research directions aim to advance intelligent monitoring and maintenance strategies for complex CPSs.

Part I

Prognostics and Remaining Useful Life Estimation

Chapter 1

Overview of Remaining Useful Life Estimation

The Remaining Useful Life (RUL) of a system (Fig. 1.1) is defined as the time interval between the current time instant t_0 and the expected failure time t_1 [1]:

$$\text{RUL} = t_1 - t_0 \tag{1.1}$$

It represents a fundamental component of a PHM framework, as it quantifies how long a system or component can continue to operate before reaching a failure or critical condition [13, 11].

The RUL is inherently a *random variable*, whose value depends on several factors such as the component's age, the operating environment, and the condition-monitoring data available at time t_0 [1]. Therefore, effective RUL estimation requires the deployment of sensor networks capable of continuously monitoring relevant environmental conditions and system parameters in real time. The choice of sensing modalities depends on the system architecture and dominant failure mechanisms. For instance, vibration measurements are particularly effective for detecting early wear in rotating machinery, while temperature or voltage sensors provide insight into thermal or electrical stress conditions [13]. The acquired raw signals must then be processed and transformed into meaningful representations of system health [14]. This transformation typically involves feature extraction and dimensionality reduction techniques, resulting in the definition of a Health Indicator, also referred to as a Health Index (HI), a normalized scalar variable generally ranging from 1 (healthy state) to 0 (failure state), that reveals degradation and quantifies the current condition of the system [14]. Based on the evolution of the HI, the overall degradation process can be divided into four generic stages: healthy, caution, repair, and failure, whose specific thresholds depend on the characteristics and operating context of the system, as illustrated in Fig. 1.1. The HI thus

serves as an intermediate abstraction between sensor-level data and higher-level prognostic models, enabling the estimation of the RUL through the observation of degradation trends over time. In this context, two main paradigms can be distinguished according to how degradation information is exploited: *degradation-based* and *similarity-based* models.

Degradation-based approaches explicitly analyze the temporal evolution of the health indicator to model its progression toward a predefined failure threshold. These methods are applicable when safety or performance limits can be clearly defined, and degradation trend analysis can be used to infer the time-to-failure [15, 16]. In contrast, similarity-based models estimate the RUL by comparing the current degradation trajectory with historical instances that have evolved to failure, assuming that similar degradation patterns lead to comparable remaining lifetimes [17, 18].

Overall, RUL estimation encompasses a wide range of methodological families that build upon these two paradigms, each differing in how degradation features are represented, compared, or extrapolated over time. In Chapter 4, we present the implementation of novel degradation-based and similarity-based approaches, along with a comparative analysis in terms of predictive performance and computational efficiency.

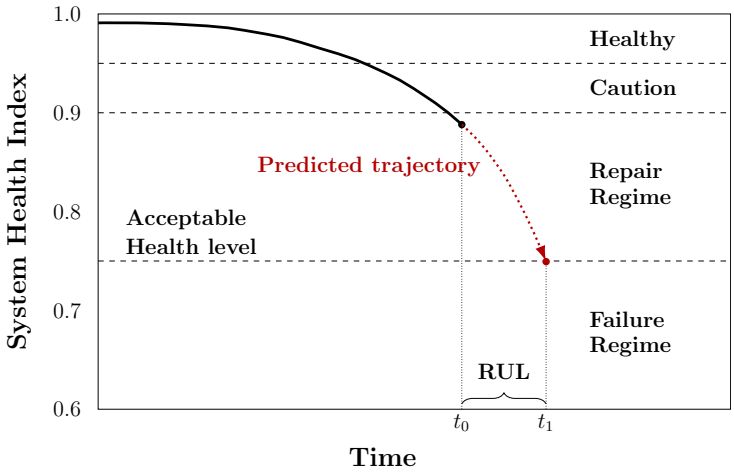

Figure 1.1: Illustration of the Remaining Useful Life (RUL), defined as the time interval between the current time t_0 and the predicted failure time t_1. Adapted from the conceptual representation proposed in [1].

1.1 Health Indicator (HI) construction

A fundamental step in any RUL estimation pipeline is the definition of the health index, a quantitative metric that reflects the current degradation level of a component. The methods for constructing HIs can be broadly categorized into four main families, depending on the way degradation information is extracted and represented [19].

- **Feature-based HIs:** These methods employ classical signal processing techniques such as time-domain analysis (e.g., root mean square, kurtosis) and frequency-domain analysis (e.g., spectral entropy, amplitude ratios) to extract degradation-related features from raw monitoring signals. The selected statistical or spectral features directly serve as the HI, capturing variations in the health condition and fault characteristics of machinery [13].
- **Multi-feature fusion-based HIs:** To achieve a more comprehensive assessment of equipment health, multiple individual features can be fused to form a composite HI. Feature fusion can be accomplished through dimensionality reduction techniques such as Principal Component Analysis (PCA), correlation analysis, or ML models that assign weights to features based on their relevance to degradation. This approach improves robustness against noise and single-sensor limitations [20, 13].
- **Distance-based HIs:** In these methods, the health state is quantified by measuring the similarity or dissimilarity between the current condition and a reference healthy state. Common distance metrics include Euclidean distance, Mahalanobis distance, and Kullback-Leibler divergence. The resulting distance value serves as the HI, where larger distances typically indicate greater deviation from the normal operating condition [21].
- **Deep learning-based HIs:** Recent advancements in deep learning have enabled the automatic derivation of HIs from raw sensory data. Architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Autoencoders (AEs) are capable of learning hierarchical representations that capture nonlinear degradation patterns. These models can directly process complex, high-dimensional monitoring data, producing latent embeddings that serve as data-driven HIs for prognostics [22, 18].

Overall, these approaches represent complementary strategies: traditional signal-based methods provide interpretable and physically grounded indicators, whereas data-driven and deep learning methods yield more abstract but potentially more informative health representations suitable for modern PHM frameworks. Chapter 2 presents a comparative analysis of AE-based deep learning and feature-based approaches for HI extraction.

1.2 RUL prediction methods

Existing RULs prediction methodologies can be broadly divided into three paradigms: *model-based*, *data-driven*, and *hybrid* methods [14, 1]. Each paradigm has different theoretical foundations, advantages, and limitations, and their selection depends on the availability of data, knowledge of the physics of the system, and computational constraints.

1.2.1 Model-based methods

Model-based (or physics-based) approaches rely on mathematical and physical models to simulate system degradation over time [8]. They exploit known physical failure mechanisms, such as wear, crack propagation, or fatigue, and are typically formulated through stochastic or deterministic degradation models. Commonly used model-based techniques include physical degradation laws (e.g., Archard's law for wear, Paris' law for crack growth) and state-space models such as Kalman Filters (KF), Extended Kalman Filters (EKF), or Particle Filters (PFs) to simulate degradation behaviors [23].

Applications of model-based methods span multiple industrial domains. In *aerospace systems*, Paris' law predicts fatigue life in aircraft structures; in *rotating machinery*, Archard's law is used to model wear, complemented by dynamic and vibration models for fault detection; in *battery systems*, electrochemical models such as the Doyle-Fuller-Newman (DFN) model and Solid Electrolyte Interphase (SEI) growth models are widely adopted. These models offer strong interpretability but may struggle with highly nonlinear dynamics and parameter uncertainty. Consequently, they are often limited to systems where degradation mechanisms are well understood and measurable.

1.2.2 Data-driven methods

The data-driven paradigm has become increasingly dominant with the proliferation of sensor data, CPSs, and advanced computational resources. These methods infer degradation behavior directly from data without requiring explicit physical knowledge of the system. Data-driven RULs prediction models encompass statistical, Machine Learning (ML), and Deep Learning (DL) approaches.

- **Statistical approaches:** Statistical models describe degradation evolution using stochastic processes or probabilistic distributions. Examples include the Wiener process, suitable for continuous and noisy degradation trajectories, and the Gamma process, effective for monotonic degradation [24]. Other models such as Autoregressive Moving Average (ARMA), Markov chains, and

Weibull distributions are commonly applied in systems exhibiting stationary, discrete, or probabilistic degradation behaviors, respectively [8].

- **ML and DL approaches:** Machine learning methods such as Support Vector Machine (SVM) and Gaussian Process Regression (GPR), have shown strong predictive capabilities for complex, nonlinear degradation trends [25, 26].

  Deep learning architectures like CNNs and RNNs, especially Long Short-Term Memory (LSTM) variants, further enhance the capacity to model temporal dependencies and nonlinear patterns in high-dimensional sensor data [27, 28]. Recent studies also incorporate attention mechanisms and transfer learning to improve model generalization and adaptability [29, 30].

While data-driven approaches demonstrate excellent adaptability, they depend heavily on large, high-quality datasets and often lack physical interpretability. Nevertheless, they remain the only feasible solution in scenarios where no sufficient physical knowledge exists to model the analyzed systems. In this dissertation, purely data-driven methods are adopted due to the lack of physical models describing the analyzed use cases.

1.2.3 Hybrid methods

Hybrid models combine the interpretability of physics-based models with the flexibility of data-driven learning to overcome their respective limitations.

A common hybridization strategy involves embedding physical constraints or domain knowledge into neural network training, leading to Physics-Informed Neural Networks (PINNs) [31]. Other works combine neural networks with PFs [32], or Gaussian Process Regression with PFs for uncertainty-aware RUL estimation [33].

Despite higher computational complexity, hybrid frameworks often outperform single-method approaches in terms of accuracy and robustness, particularly in dynamic or data-limited environments.

1.2.4 Metrics

The performance of RUL estimation algorithms is generally assessed by evaluating the errors between the predicted and the actual RUL values [34]. In the following, we present the evaluation metrics that will be adopted in this work to objectively measure the accuracy and reliability of the RUL estimation algorithms.

We define the multivariate time series acquired from a generic system unit equipped with m sensors as a matrix:

$$\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}] \tag{1.2}$$

where T is the length of the time series and $\boldsymbol{x}_t \in \mathbb{R}^m$ is a column vector that contains the m sensor readings at each time step t.

Let $\hat{\text{RUL}}(\boldsymbol{X}_l)$ denote the predicted remaining useful life of the l-th unit, and $\text{RUL}(\boldsymbol{X}_l)$ the corresponding true value. The performance of a prognostic model on a test dataset with L units is evaluated using metrics that depend on the prediction error e_l on each unit $l = 0, \ldots, L-1$:

$$e_l = \hat{\text{RUL}}\left(\boldsymbol{X}_l\right) - \text{RUL}\left(\boldsymbol{X}_l\right) \tag{1.3}$$

- **Root Mean Square Error (RMSE):**

$$\text{RMSE} = \sqrt{\frac{1}{L}\sum_{l=0}^{L-1} e_l^2} \tag{1.4}$$

- **Score** [35]:

$$S = \sum_{l=0}^{L-1}\left(e^{\gamma|e_l|} - 1\right), \quad \gamma = \begin{cases} 1/a_1 & \text{if } e_l \leq 0 \\ 1/a_2 & \text{otherwise} \end{cases} \tag{1.5}$$

  Usually, $a_1 > a_2$ so the score imposes a greater penalty on late predictions ($e_l > 0$) compared to early predictions ($e_l \leq 0$). The lower the value of S, the better is the performance.

- **Accuracy** [36, 29]:

$$A = \frac{1}{L}\sum_{l=0}^{L-1} f(e_l), \quad f(e_l) = \begin{cases} 1 & \text{if } e_l \in [-a_1, a_2] \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

  Accuracy categorizes a predicted $\hat{\text{RUL}}$ as correct when $e_l \in [-a_1, a_2]$. Errors outside this range, specifically $e_l < -a_1$ and $e_l > a_2$, correspond to false positives (FP) and false negatives (FN), respectively.

Alpha - Lambda Performance

To evaluate the models' performance over time, $\alpha - \lambda$ plots [37, 38, 39] are adopted. These plots visualize the predicted RUL against the actual remaining cycles to failure, providing insight into the models' accuracy across different stages of the component's degradation.

The $\alpha - \lambda$ metric [40] is a binary indicator used to evaluate the accuracy of the predicted RUL at a specific time instance, t_λ. It verifies whether the predicted RUL at t_λ, denoted as $\hat{\text{RUL}}_\lambda$, lies within an acceptable range determined by the

α-bounds, which are defined as a percentage of the actual RUL at t_λ, RUL_λ. The metric is formally defined as:

$$\alpha - \lambda = \begin{cases} 1 & \text{if } \hat{\text{RUL}}_\lambda \in [(1-\alpha)\text{RUL}_\lambda, (1+\alpha)\text{RUL}_\lambda] \\ 0 & \text{otherwise} \end{cases} \tag{1.7}$$

where λ refers to the t_λ instant at which the prediction is performed and α is the percentage value defining the acceptable confidence bounds. The α value is typically selected by an analyst based on the accuracy required to satisfy user-specific success criteria [40].

Chapter 2

Autoencoder-based Health Indicator Construction for Satellite Health Monitoring

With the fast development of space technology, Prognostics and Health Management (PHM) techniques are playing an increasingly significant role in ensuring satellite safety and enabling intelligent maintenance strategies. Satellite PHM (S-PHM) combines in-orbit telemetry data with data science methods and classical technology to implement smart monitoring methods and guide the management of satellites. As introduced in Chapter 1, health monitoring involves detecting component degradation, diagnosing its causes, and predicting the Remaining Useful Life (RUL), in order to plan effective maintenance actions and prevent unexpected system shutdowns [41].

Nowadays, satellite structures are becoming increasingly complex, incorporating multiple interconnected devices and subsystems that must remain healthy to ensure proper operation. Due to these interconnections, even a minor failure can endanger the safety of the entire mission if no corrective action is taken [42]. Furthermore, satellites operate for extended periods in the harsh space environment; therefore, despite extensive reliability tests, checks, and preventive measures during the design phase, faults may still occur during operation, leading to sudden or gradual degradation of material properties and the satellite's overall health status [43]. When a malfunction is detected, satellites cannot be physically repaired in orbit; however, they can be reconfigured from the ground by experts who activate embedded redundancies.

RUL estimation is one of the key aspects of S-PHM; to ensure satellite safety throughout the entire mission, it is essential to predict the RUL of critical systems

at an early stage so that, once a failure occurs, effective corrective actions can be promptly taken from the ground station. In addition to improving satellite safety standards, RUL estimation enables the assessment of components' optimal lifespan, thereby reducing raw material consumption, energy usage, waste, and pollution, which aligns with the principles of today's low-carbon economy.

The RUL of a system depends on the current time, the operating environment, and condition monitoring [1]. Therefore, a fundamental requirement for RUL estimation is that sensors must be installed in representative satellite areas to monitor environmental conditions and specific system parameters of interest in real time. As discussed in Chapter 1, RUL estimation is typically performed by developing a model based on the time evolution of Health Indicators (HIs) or features [44]. A signal-based feature is a quantity derived from processing signal data that reflects equipment degradation. When designing algorithms for Predictive Maintenance (PdM), health indicator trends can be used to identify the degrading performance of a system, indicative of developing fault conditions. As highlighted in Section 1.1, significant features can be extracted using time-domain, frequency-domain, and time-frequency analysis [45].

With the increasing number of sensors in satellite systems, telemetry data has become high-dimensional, multimodal, and heterogeneous, creating a growing need to reduce the dimensionality of the feature set [46]. Feature fusion techniques are commonly employed for dimensionality reduction [47], with Principal Component Analysis (PCA) [48] being one of the most widely adopted approaches. This chapter describes a feature selection method based on the Autoencoder (AE) to achieve dimensionality reduction. In the health monitoring domain, AEs have already been utilized to extract visually interpretable profiles of industrial systems [49] and to quantify epistemic and aleatory uncertainties [50]. Here, we propose to use the AE to identify which features should be monitored in cases where no degradation model is available or as the first stage of the processing chain when additional information is provided. More specifically, this study evaluates the capability of AE-based approaches to extract more accurate health indicators for RUL estimation in the context of injected drifts in sensor data.

The chapter is structured as follows. Section 2.1 discusses the theoretical background of the research and introduces the proposed approaches. Before conclusion, Section 2.2 presents the achieved numerical evidences.

2.1 Mathematical models

As discussed in Chapter 1, RUL estimation is a field in which several families of methods have been proposed so far. The two main paradigms are similarity-based approaches [17] and degradation-based approaches [15]. Both methods require the

identification of one or more features to be tracked during the system's lifetime, whose behavior changes predictably as the system degrades; see Fig. 2.1 for some graphical examples. The accuracy of the predicted RUL depends not only on the model but also on the quality of the selected degradation features [45]. The proposed AE-based framework aims to identify a single optimal HI in both degradation-agnostic and degradation-aware scenarios, independently of the chosen model.

In the case of satellite telemetry data obtained from a target subsystem, the identification of the features to be tracked starts from a set of L available time series. We represent successive chunks of this data as $\boldsymbol{x}[t] \in \mathbb{R}^{n \times L}$, where each input $\boldsymbol{x}[t]$ contains n successive sensor readings from each time series, and t denotes the index of the time window. Without loss of generality, we consider standardized time series, i.e., with zero mean and unit variance.

To simulate degradation scenarios, artificial contributions are introduced following profiles that emulate wear effects, such that for each time series:

$$\tilde{\boldsymbol{x}}_i[t] = \boldsymbol{A}_i[t]\boldsymbol{x}_i[t] + \boldsymbol{d}_i[t] \qquad i = 1, \ldots, L \tag{2.1}$$

where $\boldsymbol{x}_i[t]$ is the vector containing readings from the i-th sensor, $\boldsymbol{A}_i[t] \in \mathbb{R}^{n \times n}$ and $\boldsymbol{d}_i[t] \in \mathbb{R}^n$ model the disturbances emulating the degradation, and $\tilde{\boldsymbol{x}}[t] = (\tilde{\boldsymbol{x}}_1[t], \ldots, \tilde{\boldsymbol{x}}_L[t])$ represents the degraded signals. In particular, we focus on three possible scenarios, where deviations are modeled as:

(i) an additional offset that causes an exponential increase of the mean value (MVI). This degradation is modeled with $\boldsymbol{A}_i[t] = \boldsymbol{I}_n$ where $\boldsymbol{I}_n$ is the n-dimensional identity matrix, and $\boldsymbol{d}_i[t] = \mu[t]\mathbf{1}_n$ where $\mathbf{1}_n$ is a n-dimensional vector whose entries are all 1, and $\mu[t]$ is a scalar representing the additive mean. In particular, $\mu[t] = c\, e^{(t-t_0)/\tau}$ where τ scales the exponential trend and t_0 is a reference time at which the degradation has amplitude c;

(ii) additive zero-mean white noise with linearly increasing standard deviation (SDI), i.e., $\boldsymbol{A}_i[t]$ still are $\boldsymbol{I}_n$, while elements of each $\boldsymbol{d}_i[t]$ are drawn as instances of a Gaussian random variable with zero mean and standard deviation $\sigma[t] = \beta t$, where β controls the increase rate;

(iii) a rotation of the signal principal subspace of an angle that increases linearly (SRI). The rotation is modeled with $\boldsymbol{d}_i[t] = \mathbf{0}_n$ and $\boldsymbol{A}_i[t] = \boldsymbol{U}_i \boldsymbol{R}_{\theta[t]} \boldsymbol{U}_i^\top$ rotating of an angle $\theta[t]$ the subspace spanned by the eigenvectors associated to the two largest eigenvalues. Hence, $\boldsymbol{U}_i$ is the matrix whose columns are the eigenvectors of the covariance matrix of $\boldsymbol{x}_i[t]$ and $\boldsymbol{R}_{\theta[t]}$ is a matrix rotating two specific elements of a vector.

RUL estimation relies on tracking the system's degradation level, which in this case is represented by the features μ, σ, or θ used to generate the disturbances. It is

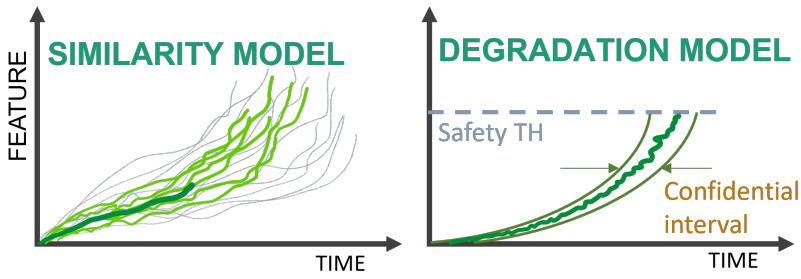

Figure 2.1: High-level representation of similarity and degradation-based models for RUL estimation, based on their capability to track a feature correlated with the degradation level.

worth noting that a robust RUL estimation corresponds to tracking these features in a way that is independent of the specific cause of degradation. The solution we propose is to use the reconstruction error of an AE as a proxy for the degradation level.

AEs are a type of neural network composed of two sub-networks, namely an encoder and a decoder, connected through a bottleneck layer that represents the latent space. The encoder compresses $\boldsymbol{x}[t]$ into a lower-dimensional representation $\boldsymbol{z}[t] \in \mathbb{R}^k$, while the decoder takes $\boldsymbol{z}[t]$ as input and produces $\hat{\boldsymbol{x}}[t]$ as output. An AE is trained to minimize the difference between $\boldsymbol{x}[t]$ and $\hat{\boldsymbol{x}}[t]$, for instance by minimizing the Mean Squared Error (MSE), defined as:

$$\text{MSE}[t] = \frac{1}{n} \|\boldsymbol{x}[t] - \hat{\boldsymbol{x}}[t]\|_2^2 \tag{2.2}$$

During training, the normal behavior represented by $\boldsymbol{x}[t]$ is learned and embedded in the parameters of both the encoder and decoder sub-networks. When an input is corrupted according to (2.1), the encoder is expected to be less effective in compression, leading to poorer reconstruction by the decoder. For this reason, any form of degradation should affect the reconstruction quality, which we measure in terms of MSE. Hence, MSE can serve as a degradation score to be monitored for RUL estimation. In the setting considered here, the MSE is therefore expected to correlate with the degradation level μ, σ, or θ.

The MSE of an AE trained on the normal behavior of the signal is therefore a feature completely agnostic to the cause of degradation. However, in some scenarios, system engineers can leverage prior knowledge about degradation mechanisms to make feature tracking, i.e., RUL estimation, more effective. For instance, in [37], the features to be monitored and their degradation trends are well known. To address cases like this, we propose a second stage in addition to the AE, which

specifically predicts the features of interest. This stage implements a regression function that takes both the signal $\tilde{\boldsymbol{x}}[t]$ and its corresponding reconstructed signal $\hat{\boldsymbol{x}}[t]$ as inputs, and produces an estimation of the degradation level as output. In our framework, the regressors are implemented as neural networks that generate the estimates $\hat{\mu}$, $\hat{\sigma}$, or $\hat{\theta}$ of the features μ, σ, or θ, depending on the scenario under consideration. The block diagram of the entire proposed framework is shown in Fig. 2.2.

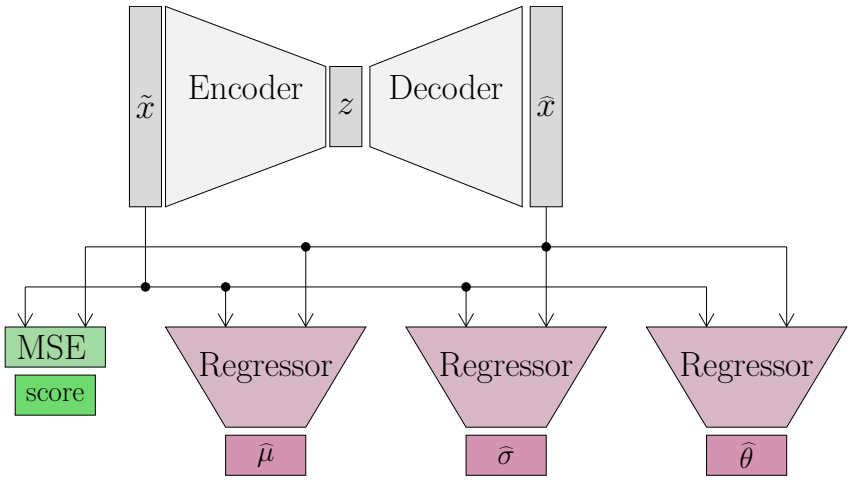

Figure 2.2: Block scheme of the proposed framework. The score of the AE works as an agnostic estimator of the degradation level, along with three different regressors tracking mean, standard deviation, and the angle of rotation of the principal subspace.

2.2 Numerical evidences

The proposed approach for condition monitoring is tested on a satellite telemetry dataset. The dataset contains the readings of $L = 4$ sensors measuring the same quantity related to four different components operating within the same subsystem. The readings were collected over an 18-months period at a rate of one sample every 2 seconds, i.e., approximately 24 million samples for each time series. Each time series is standardized by removing the mean and scaling to unit variance and then it is split into windows of length $n = 1920$, corresponding to 64 minutes. The dataset is split into three sets: the first 12 months are employed as training set to make the AE model learn the normal behavior $\boldsymbol{x}[t]$, the following 6 months form both the training set for the regressors (4 months) and the test set for performance assessment (2 months). The latter two sets are corrupted by the degradation effects

according to (2.1)[1].

As a result of a preliminary architectural search, we selected an AE with a symmetrical structure, i.e., the decoder is a mirrored version of the encoder. The encoder processes the $n \times L$-dimensional input and compresses it into a k-dimensional latent vector, with $k = 384$, by means of three one-dimensional convolutional (Conv1D) layers followed by a dense layer. The decoder begins with a dense layer and continues with three transposed convolutional (ConvT1D) layers. The hyperparameters characterizing the encoder and decoder are reported in Table 2.1.

Table 2.1: Structure and hyperparameters characterizing the adopted AE

Layer Type	**Filters**	**Kernel Size**	**Stride**	**Output Size**	**Activation Function**
Encoder					
Input				1920×4	
Conv1D	16	150	6	320×16	ReLU
Conv1D	32	100	4	80×32	ReLU
Conv1D	64	50	4	20×64	ReLU
Dense				384	Linear
Decoder					
Input				384	
Dense				20×64	ReLU
ConvT1D	32	50	4	80×32	ReLU
ConvT1D	16	100	4	320×16	ReLU
ConvT1D	4	150	6	1920×4	Linear

Regarding the regressor, it has a similar architecture to the AE's encoder. The only differences are an additional $n \times L$ input $\hat{\boldsymbol{x}}$, making the Input layer output size of $n \times 2L$, and the final dense layer with just 1 output neuron.

We compare the performance of the regressor and the AE with specific algorithms for chosen classes of degradation. Specifically, we use as reference the mean estimator for MVI, the standard deviation estimator for SDI, and Oja [51] for SRI.

Oja is an iterative algorithm that estimates the eigenvectors defining the principal subspace of a time series. In our case, for each $i = 1, \ldots, L$, the t-th iteration

[1]For MVI, the training set of the regressors uses $c = 0.1$, $t0 = 44$days, $\tau = 1\,000\,000$, for the test set $c = 0.001$, $t0 = 14$days, $\tau = 500\,000$. For SDI $\beta = 1/N$ where N is the number of signal windows. For SRI, the angle θ linearly increases from 0 to $\pi/36$ over the entire length of the dataset.

produces $\hat{\boldsymbol{U}}_p^i[t]$ which is the estimate of the two principal eigenvectors $\boldsymbol{U}_p^i \in \mathbb{R}^{n\times 2}$:

$$\hat{\boldsymbol{U}}_p^i[t] = \Omega\left(\hat{\boldsymbol{U}}_p^i[t-1] + \gamma \tilde{\boldsymbol{x}}_i[t]\tilde{\boldsymbol{x}}_i[t]^\top \hat{\boldsymbol{U}}_p^i[t-1]\right) \tag{2.3}$$

where γ is the learning rate, and $\Omega(\cdot)$ performs the orthonormalization of the columns of the argument. Oja is run over the entire test set, for $t = 0$ the algorithm considers U_p^i as the initial matrix. The estimation error is measured with the reconstruction error [52], defined as

$$e_i[t] = \left\| \boldsymbol{U}_p^i - \hat{\boldsymbol{U}}_p^i[t]\hat{\boldsymbol{U}}_p^i[t]^\top \boldsymbol{U}_p^i \right\|_F \tag{2.4}$$

where $\|\cdot\|_F$ indicates the Frobenius norm of the argument. The quantity we use to estimate the degradation level is the average of $e_i[t]$ over i that we indicate as $e[t]$.

2.2.1 Results

We report here the achieved performance in terms of the ability to track the three considered types of degradation effects. Note that for the AE and Oja methods, which do not directly estimate the tracked feature, the results are reported as rescaled scores.

Results for MVI are shown in Fig. 2.3 and in the left columns of Fig. 2.6.

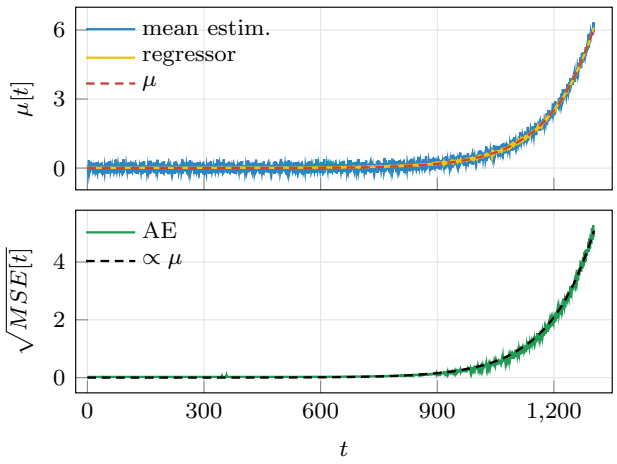

Figure 2.3: Top plot contains estimated μ values by means of the proposed regressor and a reference mean estimation. Bottom plot compares the square roots of the AE scores with the rescaled true profile.

The former highlights how the two proposed solutions, namely the AE and the regressor, significantly reduce the estimation variance compared to the reference estimator (mean estimator), especially for low values of μ. The latter presents the

histograms obtained when μ is fixed at two specific target values, $\mu = 0.1$ and $\mu = 2$, for the two proposed solutions. Although all methods can track μ with a precision that slightly depends on the target value, the regressor exhibits the lowest estimation variance.

Similarly, Fig. 2.4 and the central column of Fig. 2.6 show the results for the SVI scenario.

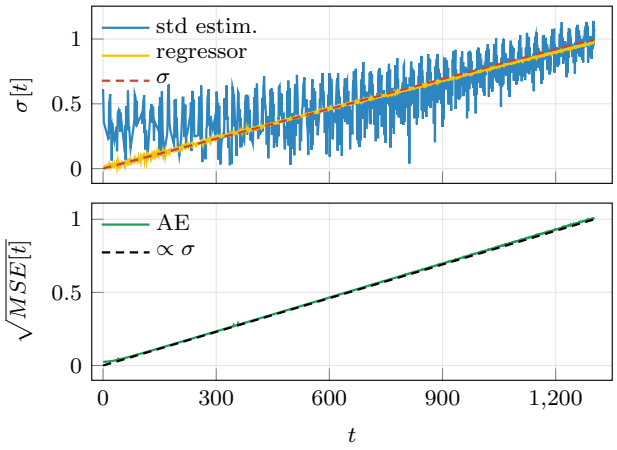

Figure 2.4: Top plot contains estimated σ values by means of the proposed regressor and a reference standard deviation estimation. Bottom plot compares the AE scores with the rescaled true profile.

In this case, the proposed solutions clearly outperform the reference standard deviation estimator, which suffers from high variance and bias, particularly at low σ values. Compared to the regressor, the AE demonstrates lower variance.

Regarding the SRI case, the results are presented in Fig. 2.5 and in the right column of Fig. 2.6.

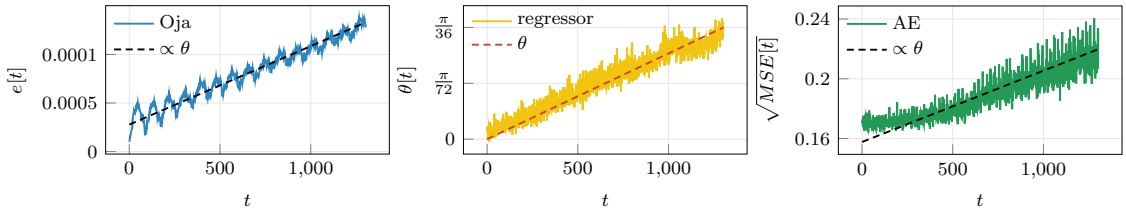

Figure 2.5: Left plot contains the error profile for Oja compared with the scaled reference. The estimations of θ and its scaled version for the two proposed solutions are in the other two plots.

Although all three estimators encounter some difficulty in tracking the trend of θ, the regressor achieves the best performance in terms of both accuracy and variance.

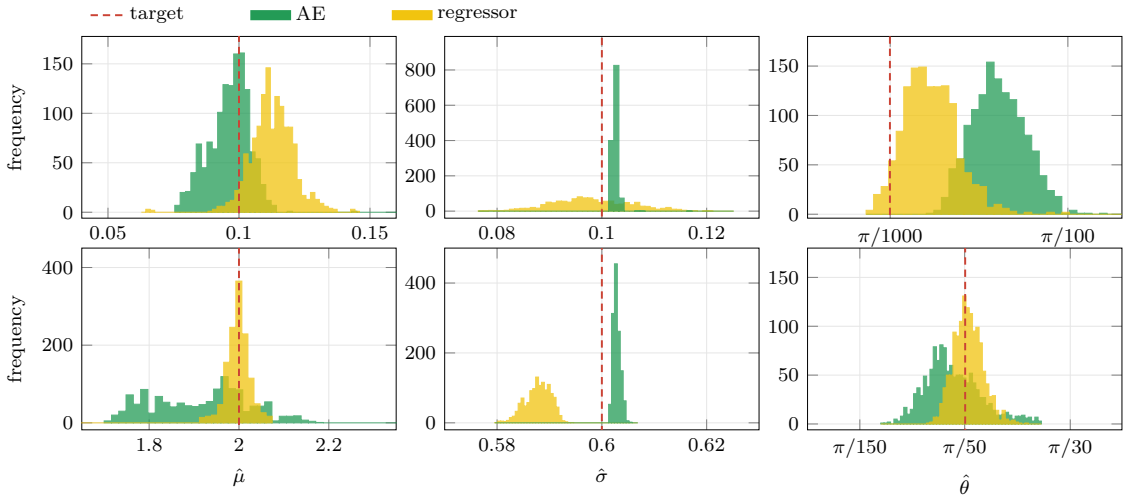

Figure 2.6: Histograms of regressor's predictions and AE's scaled scores for some target values of μ (left plots), σ (center plots), and θ (right plots).

As a final remark, while the regressor and the reference estimator are specifically designed for the corresponding degradation models, the AE produces a score capable of tracking all three degradation trends without any assumption about the cause of degradation.

2.3 Conclusion

This Chapter focuses on the identification of meaningful and informative (HIs) or features that can effectively represent the degradation level of a system. Selecting an appropriate feature to be tracked is a critical task that usually requires in-depth knowledge of the monitored system and its degradation mechanisms.

We propose a preliminary study on an AE-based approach that employs the reconstruction error as a score capable of tracking multiple degradation effects. The AE can also be integrated into the design of a regressor properly tuned on a specific model of degradation.

The proposed approaches have been tested on real telemetry data from an in-orbit satellite, where three different degradation trends were artificially injected into the signals to simulate realistic degradation scenarios. The obtained results confirm that the AE represents a promising solution for both degradation-agnostic and degradation-aware scenarios.

Chapter 3

Multi-class Similarity-based Approach for Remaining Useful Life Estimation

The estimation of the Remaining Useful Life (RUL) of a machine is one of the major prognostic activities in industrial applications [53] [1]. Predicting the failure time enables effective planning of maintenance activities, thereby reducing unnecessary preventive interventions and lowering overall maintenance costs [54] [55].

As discussed in Chapter 1, the field of RUL estimation has seen the development of various methodologies. Early approaches were model-based and required a thorough understanding of the physical principles governing system deterioration and failure [56]. With the advent of advanced computing and large-scale data collection, there has been a significant shift toward data-driven methods [57]. These approaches can infer the RUL from historical data without the need for a comprehensive understanding of the underlying physics of the system [58].

When historical run-to-failure data is available, several data-driven approaches for RUL estimation leverage Autoencoder (AE) in conjunction with similarity-based metrics [18, 29, 59, 36]. The AE is used to convert operational multi-sensor time-series data into a health index (HI), i.e., an indicator that quantifies the system's health state, typically ranging from 1 to 0 [1]. The approach operates in two stages. During the offline phase, the AE is trained on historical run-to-failure data to generate the corresponding HI curves, which are collected in a reference library. During the online phase, the multivariate time series acquired from the monitored system is transformed into the corresponding HI, which is then compared with the reference library to estimate the system's RUL.

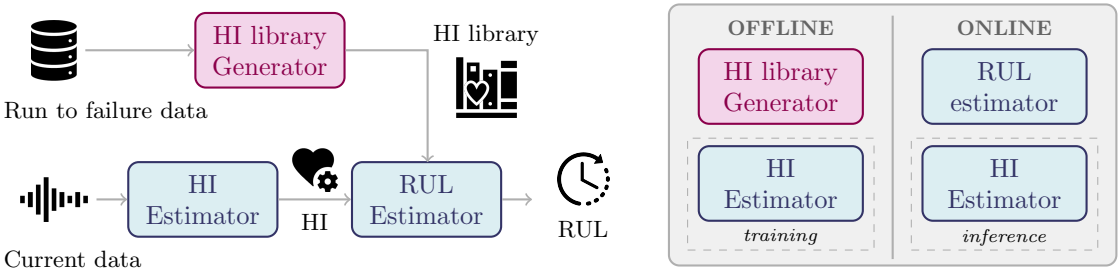

Figure 3.1: Connection of the main blocks of the reference framework for RUL estimation: the HI library generator (trained offline), the HI estimator (trained offline and deployed online), and the similarity-based RUL estimator (used in inference).

This similarity-driven approach has been proposed for a multitude of AE architectures. Malhotra et al. [36] introduced the use of an LSTM Encoder-Decoder, Yu et al. [18] employed a bidirectional recurrent neural network AE design, and Duan et al. [29] improved the BiGRU AE by incorporating an attention mechanism and skip connection.

In this chapter, we propose a RUL estimator based on AE and similarity methods that, unlike existing approaches, explicitly accounts for the failure mode to enhance prediction accuracy. A system may comprise multiple components, subsystems, or processes, each potentially leading to different types of failures. We assume C failure modes, each representing a specific type of malfunction, fault, or breakdown. These distinct classes are considered both in the creation of the HI library and during the RUL estimation phase. The HI library is partitioned into C sub-libraries, each corresponding to a different failure mode. In the online phase, when new data is acquired, a classifier identifies the failure mode from the multivariate input data and selects the appropriate library for comparison. This multi-class approach allows us to handle diverse failure modes and adjust its response based on the characteristics of the observed failures. Furthermore, since the classification precedes the similarity measure, the comparison is restricted to only those HI curves sharing the same failure class. Since curves of the same class are likely to exhibit similar failure patterns, the proposed approach offers a more accurate estimation.

This chapter is organized as follows. Section 3.1 presents the mathematical model, while Section 3.2 describes the simulation framework and reports the numerical evidence. Finally, the main conclusions are drawn in the last section.

3.1 Mathematical model

A similarity-based estimator of the RUL of an operating machine computes a health index (HI) and compares it with a library of run-to-failure HI curves referring

to similar machines. The entire estimation process comprises three blocks: *(i)* a block that processes historical time-to-failure data and produces the HI library, referred to as HI library Generator; *(ii)* a block that maps multivariate series to the corresponding HI curves, named HI estimator; and *(iii)* a block that takes an HI curve and predicts the machine's failure time online, which we call RUL estimator.

The HI library generator works exclusively offline, creating the HI library from historical data. The HI estimator is trained offline and deployed online to predict the HI curve of the current data. The RUL estimator operates online to predict the RUL of the current machine by comparing its estimated HI curve with the HI library. Fig. 3.1 illustrates the framework and the connections between the three blocks.

3.1.1 HI library generator

The historical run-to-failure dataset is modeled as a collection $\mathbb{X}$ of J multivariate time series. Each time series collects the readings of m sensors monitoring the condition of a different machine. It is modeled as a matrix $\boldsymbol{X} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_{T-1}]$ where T is the length of the time series and $\boldsymbol{x}_t \in \mathbb{R}^m$ is a column-vector that includes the m sensor readings at each time point t. In run-to-failure data, T also corresponds to the failure time and it is in general different for every time series, therefore, when we specifically refer to the j-th time series $\boldsymbol{X}_j$, we specify failure time as T_j and sensor reading for the time instant t as $\boldsymbol{x}_{j;t}$.

The collection $\mathbb{X}$ is used offline to build the HI library generator that transforms every run-to-failure time series into a Health Index (HI) time series ranging from 1 (optimal health) to 0 (imminent failure). These values are used as labels in training a linear regressor that continuously estimates the HI value from the available input time series.

As suggested in [18, 29], the HI library generator can be based on an Autoencoder (AE), which is a type of neural network consisting of an encoder (ENC) mapping the input into an embedding, and a decoder (DEC) that takes the embedding and reconstruct the input time series. In detail, AE processes time series that are split into overlapping windows $\boldsymbol{W}_t = [\boldsymbol{x}_t, \ldots, \boldsymbol{x}_{t+n-1}]$, where t ranges from 0 to $T-n-1$. The encoder embeds each window into a latent vector $\boldsymbol{z}_t \in \mathbb{R}^k$. The decoder then takes this vector $\boldsymbol{z}_t$ and reconstructs the original window, producing $\hat{\boldsymbol{W}}_t$. The objective during training is to minimize the Mean Squared Error (MSE) between the original and reconstructed windows:

$$\text{MSE} = \sum_{j=0}^{J-1} \frac{1}{nT_j} \sum_{t=0}^{T_j-n-1} \left\| \boldsymbol{W}_{j;t} - \hat{\boldsymbol{W}}_{j;t} \right\|_2^2 \tag{3.1}$$

Applying the AE to each time series $\boldsymbol{X} \in \mathbb{X}$, we obtain the latent time series

$\boldsymbol{Z} = [\boldsymbol{z}_0, \ldots, \boldsymbol{z}_{T-n-1}]$ that acts as a proxy for assessing the health status of the monitored machine [18, 29, 59, 36]. To transform $\boldsymbol{Z}$ into an HI curve, we assume the machine begins its operation in a healthy state (HI = 1). We identify this condition with the first three embeddings $\boldsymbol{Z}_{\text{norm}} = \{\boldsymbol{z}_0, \boldsymbol{z}_1, \boldsymbol{z}_2\}$ [18, 29]. As the machine continues to function, its health gradually deteriorates, causing the embeddings $\boldsymbol{z}_t$ to progressively deviate from $\boldsymbol{Z}_{\text{norm}}$. We measure this deviation as follows:

$$\delta_t = \frac{1}{|\boldsymbol{Z}_{\text{norm}}|} \sum_{\boldsymbol{z} \in \boldsymbol{Z}_{\text{norm}}} \|\boldsymbol{z}_t - \boldsymbol{z}\|_2 \tag{3.2}$$

where $t = 0, \ldots, T-n-1$, and $|\boldsymbol{Z}_{\text{norm}}| = 3$ represents the cardinality of the $\boldsymbol{Z}_{\text{norm}}$ set.

Since the δ_t profile may vary from machine to machine, we define the HI by normalizing the δ_t values within the range $[0, 1]$ as follows:

$$h_t = \frac{\delta_{\max} - \delta_t}{\delta_{\max} - \delta_{\min}} \tag{3.3}$$

where $\delta_{\max}$ and $\delta_{\min}$ represent the maximum and minimum values of the deviation δ_t observed over its operational life. The obtained vector $\boldsymbol{h} = [h_0, h_1, \ldots, h_{T-n-1}]$ is commonly defined as HI curve [18, 29] and represents the degradation path of a machine. The collection of HI curves from all the J machines forms a library denoted as $\mathbb{H}$. Fig. 3.2 illustrates the AE and the subsequent blocks that convert a time series into a HI curve.

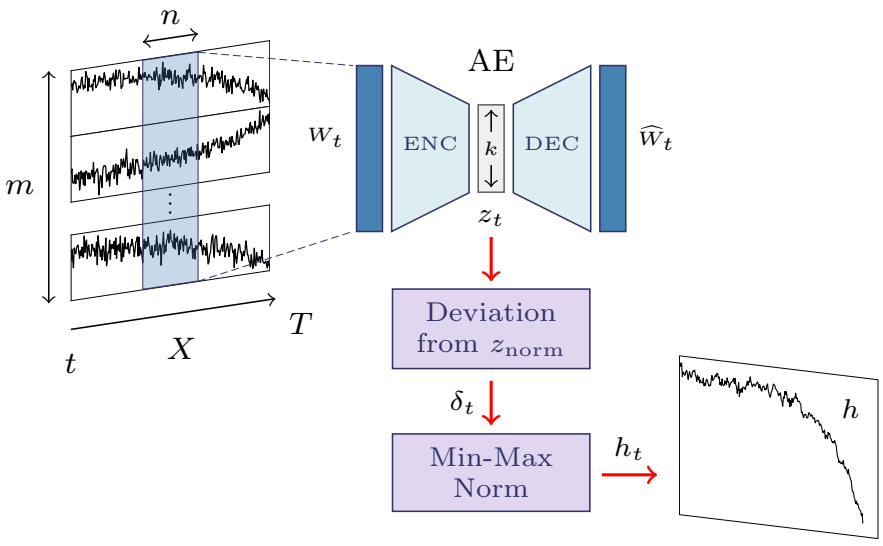

Figure 3.2: Block diagram of the AE-based HI generator.

3.1.2 HI estimator

The HI library generator creates an HI curve $\boldsymbol{h}$ for each time series $\boldsymbol{X}$ of the historical run-to-failure data. Nevertheless, these curves cannot be inferred directly from the data coming from a new machine since they require complete knowledge of the time series until failure. We use the mapping from $\boldsymbol{X}$ to $\boldsymbol{h}$ to define a supervised task addressable with a regressor. This regressor is therefore designed to estimate the HI from the data and trained with the historical data.

Authors in [18, 29, 17] have proposed a linear regressor that maps each data sample $\boldsymbol{x}_t$ into an HI as follows:

$$\hat{h}_t = \boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ \boldsymbol{x}_t \end{bmatrix} \tag{3.4}$$

where $\hat{h}_t$ indicates the estimate of h_t and $\boldsymbol{\theta}$ is the $(m+1)$-dimensional parameter vector obtained by minimizing the least square error. By applying the regressor to the times series in $\mathbb{X}$, we obtain J profiles $\hat{\boldsymbol{h}} = \{\hat{h}_0, \ldots, \hat{h}_{T-n-1}\}$ that work as a library $\hat{\mathbb{H}}$ for the computation of the similarity measure.

3.1.3 RUL estimator

The RUL estimator takes the estimations of HI contained in $\hat{\boldsymbol{h}}$ and updates an estimate of the RUL of the monitored machine. Since $\hat{\boldsymbol{h}}$ is generated through regression based on each data sample, its length grows progressively with time and operational cycles. As more data is collected over time, the curve incorporates additional information, allowing for a more comprehensive representation of the machine's degradation process which suggests that the error in predicting RUL is expected to diminish over time.

As proposed in [18, 29, 59, 36], the RUL estimator can be implemented as a similarity model that compares the current $\hat{\boldsymbol{h}}$ with the HI library $\hat{\mathbb{H}}$ and determines RUL as a weighted average of the remaining useful life indicated by the most similar HI curves in the library.

To measure the similarity between the current HI curve $\hat{\boldsymbol{h}}$ and a given run-to-failure $\hat{\boldsymbol{h}}_j$ in the library $\hat{\mathbb{H}}$, we use the following expression:

$$\mathrm{Sim}_{j,\tau}(\boldsymbol{X}) = \exp\left(-\frac{d(\hat{\boldsymbol{h}}, \hat{\boldsymbol{h}}_j, \tau)}{\lambda}\right) \tag{3.5}$$

where τ is a time lag taking into account possible displacement between $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{h}}_j$, $d(\hat{\boldsymbol{h}}, \hat{\boldsymbol{h}}_j, \tau)$ is the average squared distance between the two curves, and $\lambda > 0$ is a relaxation factor that regulates the degree of similarity under various distance

measures. The average squared distance is defined as:

$$d\left(\hat{\boldsymbol{h}}, \hat{\boldsymbol{h}}_j, \tau\right) = \frac{1}{T} \sum_{t=0}^{T-1} \left(\hat{h}_t - \hat{h}_{j:t+\tau}\right)^2 \tag{3.6}$$

where T is the length of $\hat{\boldsymbol{h}}$, i.e., the number of cycles monitored until now. It is important to note that $\hat{\boldsymbol{h}}_j$ can be used in the computation of the similarity only if $T > T_j + \tau$.

Among all HI curves in the library, only the most similar curves contribute to the RUL estimate. As suggested in [18, 29], this selection is represented by the set of pairs indexes j and time lag τ as follows:

$$\mathbb{S} = \{(j, \tau) \,|\, \mathrm{Sim}_{j,\tau}(\boldsymbol{X}) \geq \beta \max_{j,\tau} \mathrm{Sim}_{j,\tau}(\boldsymbol{X})\} \tag{3.7}$$

where the scalar $\beta \in [0,1]$ is a coefficient that defines the similarity threshold, ensuring that $\mathbb{S}$ includes only those profiles whose similarity with $\hat{\boldsymbol{h}}$ exceeds β times the maximum similarity.

Each element in $\mathbb{S}$ provides a tentative RUL as:

$$\hat{\mathrm{RUL}}_{j,\tau}(\boldsymbol{X}) = T_j - (\tau + T) \tag{3.8}$$

and the final estimate is the average over $\mathbb{S}$ of these $\hat{\mathrm{RUL}}_{j,\tau}(\boldsymbol{X})$ weighted by the corresponding similarity $\mathrm{Sim}_{j,\tau}(\boldsymbol{X})$ as follows:

$$\hat{\mathrm{RUL}}(\boldsymbol{X}) = \frac{\sum_{(j,\tau)\in\mathbb{S}} \mathrm{Sim}_{j,\tau}(\boldsymbol{X}) \times \hat{\mathrm{RUL}}_{j,\tau}(\boldsymbol{X})}{\sum_{(j,\tau)\in\mathbb{S}} \mathrm{Sim}_{j,\tau}(\boldsymbol{X})} \tag{3.9}$$

In this work, we extended the reference scheme described above by considering the existence of C distinct failure modes. Since the same cause of failure leads to a similar degradation pattern, the idea is to provide a different processing path for each failure class so that both the regressor and the similarity model can be tuned on a specific failure mode. We group the time series belonging to the same failure class C in the set $\mathbb{X}_c = \{\boldsymbol{X}_j \mid y_j = c\}_{j=0}^{J-1}$, where the label $y_j = 0, \ldots, C-1$ indicates the failure class of the j-th machine.

We therefore provide a different regressor for every class $c = 0, \ldots, C-1$. Each regressor is characterized by the parameter vector $\boldsymbol{\theta}_c$ and trained with the subset of examples $(\boldsymbol{x}_t, h_t)$ related to the c-th failure class. As a result, each failure mode c refers to a different library $\hat{\mathbb{H}}_c$ comprising the HI curves obtained from the time series and regressor associated with the c-th class of failure. In the online phase, we introduce a classifier that predicts the class of failure c from the time series $\boldsymbol{X}$

and selects c-th regressor, which transforms $\boldsymbol{X}$ into the curve $\hat{\boldsymbol{h}}$ that the similarity model compares with the only library $\hat{\mathbb{H}}_c$.

The cost of the multi-class approach consists of: *(i)* a higher number of parameters which are $\boldsymbol{\Theta} = \{\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_{C-1}\}$ where $\boldsymbol{\theta}_c$ is the $(m+1)$-dimensional parameter vector corresponding to the c-th failure class; *(ii)* the need for a classifier that determines the failure affecting the current data and selects the corresponding regressor.

Figure 3.3 provides a detailed representation of all the blocks and processing steps of the novel multi-class framework. In the offline phase, the training of the AE and the generation of the HI curves $\mathbb{H}$ remain unchanged, while the regressor is replaced by a bank of C regressors that must be trained before being employed to produce the libraries $\hat{\mathbb{H}}_c$ with $c = 0, \ldots, C-1$. Differently from the reference case, the offline phase also provides the training of the classifier that, in the online phase, infers the cause of deterioration of the monitored machine and selects the corresponding regressor and similarity model.

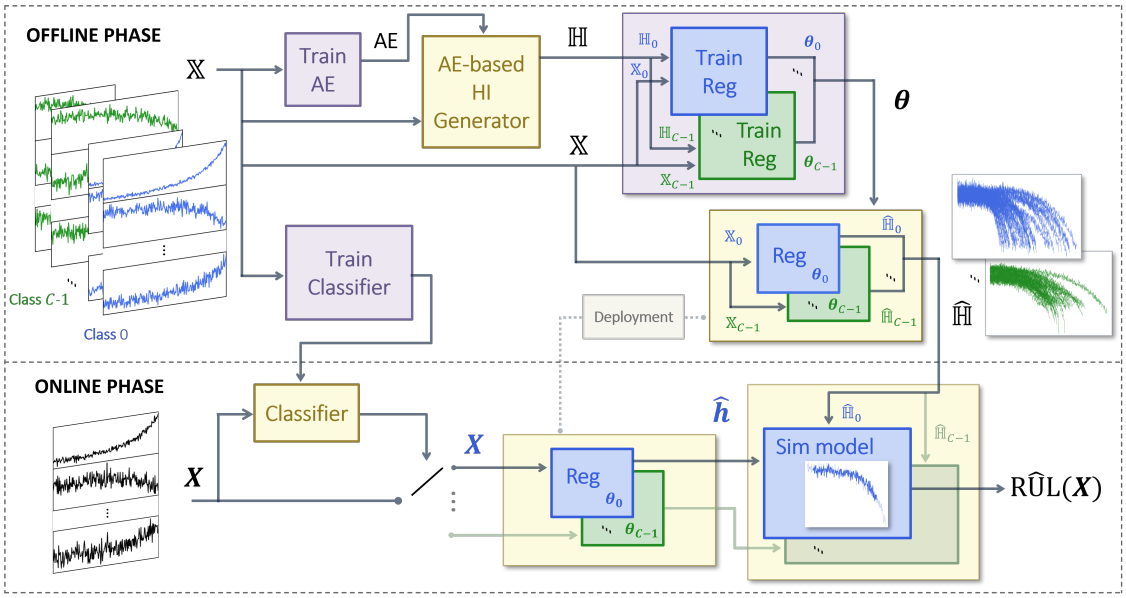

Figure 3.3: Flowchart of the multi-class framework.

A further improvement of the reference approach consists of a change in how $\hat{\text{RUL}}(\boldsymbol{X})$ is computed. We modify the set $\mathbb{S}$ that identifies the portions of HI curves in the library that are most similar to the monitored HI curve. In (3.7), for each curve indexed by j more than one τ may be selected. However, we observed that selecting only the best τ for each j leads to a more accurate performance. Hence, we redefine the set $\mathbb{S}$ as:

$$\mathbb{S} = \{(j, \tau_j) \,|\, \text{Sim}_{j,\tau_j}(\boldsymbol{X}) \geq \beta\, \overline{\text{Sim}}\} \tag{3.10}$$

where

$$\tau_j = \arg\max_{\tau} \mathrm{Sim}_{j,\tau}(\boldsymbol{X}) \text{ with } j = 0, \ldots, J-1$$

is the time lag that maximizes the similarity between $\hat{\boldsymbol{h}}$ and $\hat{\boldsymbol{h}}_j$, and

$$\overline{\mathrm{Sim}} = \max_{j} \mathrm{Sim}_{j,\tau_j}(\boldsymbol{X})$$

represents the maximum similarity between $\hat{\boldsymbol{h}}$ and all other profiles in $\hat{\mathbb{H}}$.

The tentative and final RUL estimates can thus be formally expressed as

$$\hat{\mathrm{RUL}}_{j,\tau_j}(\boldsymbol{X}) = T_j - (\tau_j + T) \tag{3.11}$$

and

$$\hat{\mathrm{RUL}}(\boldsymbol{X}) = \frac{\sum_{(j,\tau_j)\in\mathbb{S}} \mathrm{Sim}_{j,\tau_j}(\boldsymbol{X}) \times \hat{\mathrm{RUL}}_{j,\tau_j}(\boldsymbol{X})}{\sum_{(j,\tau_j)\in\mathbb{S}} \mathrm{Sim}_{j,\tau_j}(\boldsymbol{X})} \tag{3.12}$$

3.2 Numerical evidence

In this section, we present the use case employed for the experiments. We describe the dataset, detail the components of both the reference and the multi-class frameworks, and present the metrics employed for functional assessment.

3.2.1 Turbofan engine dataset

The proposed multi-class approach for RUL estimation was tested on the NASA C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) jet engine simulated data and compared with the reference estimator in which a single regressor predicts the HI curve independently of the failure mode.

The dataset, described in [35], is a collection of four sub-datasets, each featuring distinct operating conditions and fault modes, as shown in Table 3.1. These sub-datasets comprise several multivariate time series, further divided into training and test sets. Each dataset also provides a vector with the actual RUL values of the test engines, intended for evaluation purposes.

Each time series originates from a different engine with a unique operational history and manufacturing variations. Starting with normal operation, the engine eventually develops a fault. In the training dataset, this fault worsens until system failure, while in the test set, the series ends before system failure.

Data are provided as text files with 26 columns of numerical values, separated by spaces. Each row represents data from a single operational cycle, while each column

corresponds to a different variable. The first column indicates the engine unit number, and the second column corresponds to the operational cycle index. Columns three to five represent the operational settings, reflecting varying conditions during operation. The remaining 21 columns contain sensor readings collected from various engine components, such as temperature, pressure, and speed measurements, which reflect the engine's health status over time. This study focuses on the subset of 14 sensor readings that exhibit consistent trends with engine degradation[1], as suggested by Zhang et al. [60] and Li et al. [61].

To test our multi-class approach, we selected a dataset that featured multiple failure classes. The choice fell on FD003 which presents two different failure modes: HPC and fan degradation [35], to which we assigned the labels $c = 1$ and $c = 0$, respectively. Each selected time series is standardized to normalize the sensor readings to a zero mean and unit variance before being fed into the neural networks.

Table 3.1: Overview of the four C-MAPSS datasets

Dataset	FD001	FD002	FD003	FD004
Training Units (J)	100	260	100	249
Testing Units (L)	100	259	100	248
Operating Conditions	1	6	1	6
Fault Modes (C)	1	1	2	2

3.2.2 Models and parameters

Firstly, the dataset is used to train the AE. It takes windows $\boldsymbol{W}_t$ with $m = 14$ and $n = 6$ as input and returns reconstructed windows $\hat{\boldsymbol{W}}_t$ with the constraint of a latent representation with a single vector $\boldsymbol{z}_t$ of size $k = 100$. The encoder comprises a 1D-convolutional layer with 140 filters, a kernel size of 4, a stride of 2, and padding of 1, with ReLU activation, followed by a linear layer. The decoder mirrors the encoder's structure, featuring an initial dense layer with ReLU activation, succeeded by a transposed convolutional layer. The AE is trained by minimizing the Mean Squared Error (MSE) between the $\boldsymbol{W}_t$ and $\hat{\boldsymbol{W}}_t$ (see Table 3.2 for a model parameter summary).

In the online phase of the multi-class approach, the system needs a classifier that processes the input time series $\boldsymbol{X}$ and infers which of the two possible failures affects the monitored machine and selects the corresponding regressor. This classifier is implemented as a neural network that processes windows $\boldsymbol{W}_t$ of n samples with

[1]The selected sensors are the ones identified with the following indexes: #2, #3, #4, #7, #8, #9, #11, #12, #13, #14, #15, #17, #20, #21.

dimension m and returns $y \in [0,1]$ which is the probability for $\boldsymbol{W}_t$ to be related to the failure class $c = 1$. The time series classification goes through the processing of all overlapping windows $\boldsymbol{W}_t$ composing $\boldsymbol{X}$ to obtain the probabilities y_t. $\boldsymbol{X}$ is classified as $c = 1$, if the majority of y_t is greater than 0.5, otherwise $c = 0$. The architecture implementing the classifier is identical to the encoder in AE except for using 7 filters in the convolutional layer and the output that is a scalar with a sigmoid activation function that forces values in the range [0,1] (see Table 3.2). The network is trained by minimizing the Binary Cross Entropy between the prediction y and the actual class c for all windows of all J time series in the training set $\mathbb{X}$.

Table 3.2: Structure and hyperparameters characterizing the adopted neural networks, namely the AE and the classifier (Class).

Model	Layer Type	Filters	Kernel Size	Stride	Output Size	Activation Function
AE	**Encoder**					
	Input	-	-	-	6×14	-
	Conv1D	140	4	2	3×140	ReLU
	Dense	-	-	-	100	Linear
	Decoder					
	Input	-	-	-	100	-
	Dense	-	-	-	3×140	ReLU
	ConvT1D	140	4	2	6×14	Linear
Class	Input	-	-	-	6×14	-
	Conv1D	7	4	2	3×7	ReLU
	Dense	-	-	-	1	Sigmoid

The estimation of the RUL was carried out by imposing $\lambda = 0.1$ in the computation of the similarity (3.5) and $\beta = 0.99$ in the selection of the curves composing the set $\mathbb{S}$ (3.10).

We selected three metrics to evaluate the RUL prediction schemes on a test dataset with L units: RMSE (1.4), score S (1.5), and accuracy A (1.6), as detailed in Section 1.2.4. For the C-MAPSS dataset, the scaling parameters are conventionally set to $a_1 = 13$ and $a_2 = 10$ [35, 36, 29].

3.2.3 Results

The results are reported in Fig. 3.4 and Table 3.3. Fig. 3.4 shows a comparison between the predicted RUL of the test engines obtained through the proposed multi-class estimator (orange diamonds), the ones obtained from the reference approach

(purple plus signs), and the target values (light gray circles). The test engines are ordered by RUL for a better visual interpretation of the results. It is noticeable that, as the target RUL increases, the RUL estimation error typically grows. This outcome is expected because estimating the RUL of a machine is more challenging during its early stages, especially when the machine is in a healthy condition. The worsening of the estimators' performance with the increase of the target RUL is more evident in Table 3.3, which reports the RMSE, score S, and accuracy A of both approaches across different portions of the test set. In the first 20% of the test set, i.e., machines with a short target RUL, both reference and multi-class estimators achieve the best performance with 55% and 75% accuracy, respectively. By considering more machines with longer RUL, performance gradually decreases. In all cases, the proposed multi-class approach consistently outperforms the reference one, with the largest improvement observed for shorter RUL values, where better prediction yields the greatest benefit.

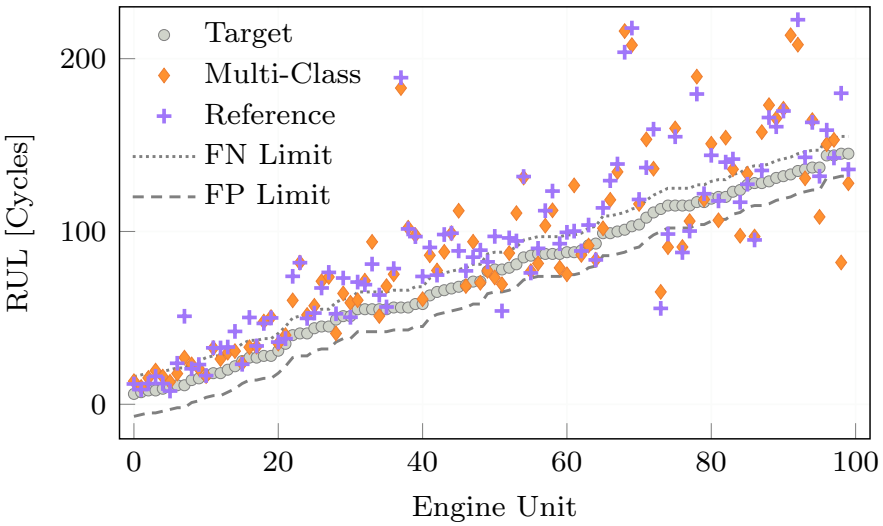

Figure 3.4: RUL predictions using the reference estimator and the proposed multi-class estimator on dataset FD003.

Table 3.3: Performance comparison

Test set Portion	Reference Estimator RMSE	S	A	Multi-Class Estimator RMSE	S	A
20%	14.9	101	55.0%	10.4	38	75.0%
40%	28.5	598k	47.5%	26.7	328k	52.5%
60%	26.7	598k	41.7%	25.0	328k	51.7%
80%	32.1	724k	35.0%	31.4	464k	47.5%
100%	33.3	788k	36.0%	32.6	469k	42.0%

To further compare the two approaches, we observe the difference between the absolute error of the reference (ref) and multi-class (MC) approaches defined as:

$$\Delta e_l = \left|e_l^{\text{ref}}\right| - \left|e_l^{\text{MC}}\right| \tag{3.13}$$

Δe_l measures the difference in the performance of the two estimators where $\Delta e_l > 0$ indicates that the multi-class outperforms the reference estimator. Fig. 3.5 shows Δe_l for $l = 0, \ldots, 19$, i.e., the engines with the shortest RUL composing the first 20% of the test set. As evident from the figure, negative Δe_l values exhibit small absolute values, implying that, for these units, the two methods perform similarly. On the other hand, positive Δe_l exhibit higher absolute values. Through post hoc analysis, this phenomenon can be attributed to the fact that, in the reference approach, the final RUL estimation includes instances from the library that do not share the same failure mode as the test unit.

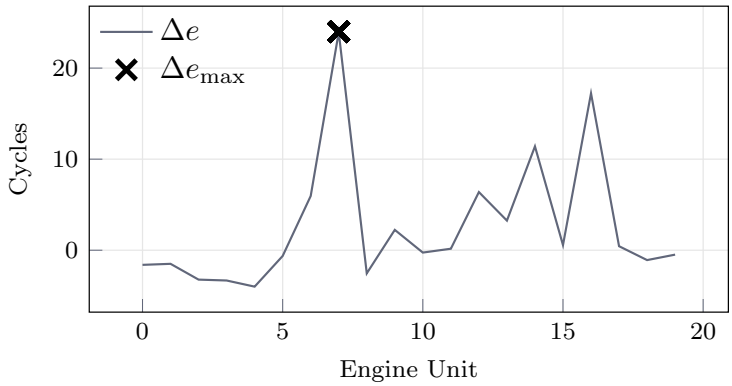

Figure 3.5: The difference in the performance Δe of the multi-class and reference estimators measured on the first 20 engine units.

To highlight this phenomenon, we focus on the largest absolute value $\Delta e_{\max}$ (black cross), observed at engine unit $l = 7$. Fig. 3.6 depicts the estimated HI curve of engine unit $l = 7$ compared with its most similar HI curve in the library, for both the multi-class (top plot) and the reference estimator (bottom plot). In the multi-class approach, engine $l = 7$ is first classified into class $c = 1$, then, its HI curve $\hat{\boldsymbol{h}}$ (in green) is estimated through the parameter vector $\boldsymbol{\theta}_1$, and subsequently compared with the library $\hat{\mathbb{H}}_1$. On the contrary, in the reference approach, the HI curve $\hat{\boldsymbol{h}}$ (in black) is estimated through the unique parameter vector $\boldsymbol{\theta}$ and compared to the entire library. In the reference approach, the most similar instance of the library (in grey) results in a preliminary $\hat{\text{RUL}}_{j,\tau}$ further from the target RUL compared to the multi-class case. From a retrospective analysis of the dataset, we observed that the most similar instance in the reference case, which leads to an overestimation of the final $\hat{\text{RUL}}$, does not belong to class 1. This finding highlights the superiority of the multi-class approach, which more accurately estimates the HI profiles through

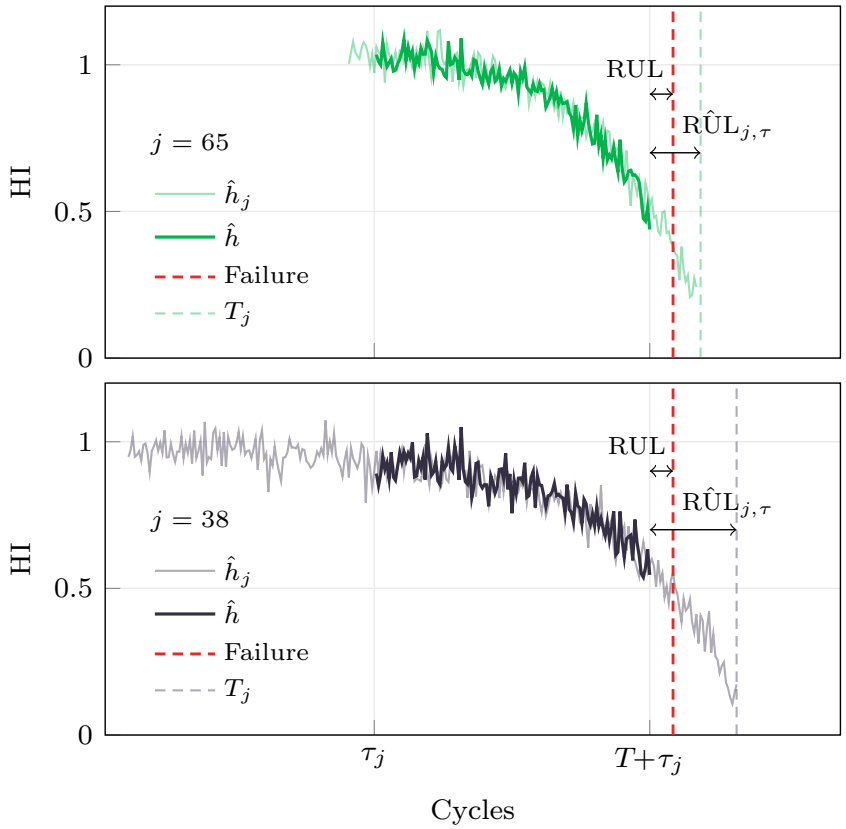

Figure 3.6: Comparison between the online HI curve of engine unit $l = 7$ and its most similar library HI curve for the multi-class estimator (top plot) and the reference estimator (bottom plot).

its set of regressors. Furthermore, it excludes comparisons with instances from different classes, thereby preventing any skewing of the final estimation.

3.3 Conclusion

An accurate estimation of the remaining useful life of a machine enables efficient and timely maintenance planning. In this work, we addressed the case of monitored systems that may fail due to multiple possible causes and proposed a similarity-driven RUL estimator that leverages the different failure modes to enhance prediction performance. The main building blocks are an autoencoder trained and used offline to generate the health index libraries used in the similarity model, a classifier, and a set of regressors that are trained offline and employed online first to predict the failure cause and then infer the health index used by the similarity to estimate the RUL. The numerical evidence shows that the multi-class approach outperforms

the reference method, which predicts the health index independently of the failure class. The improvement is more evident – up to a 20% increase in accuracy – when the machine is close to failure, hence when RUL estimation is critical.

Chapter 4

Edge Deployment of Data-Driven RUL Estimation Methods

As discussed in the previous chapters, Prognostics and Health Management (PHM) has gained significant attention in the industry as a key discipline for improving the reliability, maintainability, and affordability of complex engineering systems [62]. Prognostics focuses on predicting when a system or component will no longer meet operational requirements by analyzing deviations and degradation from nominal conditions, with the ultimate goal of forecasting the Remaining Useful Life (RUL) [63]. As introduced in Chapter 1, several approaches have been proposed based on data availability and system knowledge, falling under the broad categories of physics-based and data-driven methods [1]. In recent years, the latter, particularly those utilizing Deep Learning (DL) techniques, have attracted significant interest due to the growth of Cyber-Physical Systems (CPS) and the widespread availability of Big Data [64].

Deep learning solutions are primarily deployed on high-performance computing platforms like Graphics Processing Units (GPUs), due to their substantial computing and memory demands [65, 66]. However, there is a growing interest in implementing these algorithms on edge devices, as leveraging computational resources close to end-users achieves low latency, higher throughput, and better responsiveness in applications [67]. Additionally, edge computing enhances privacy, security, and reliability for end-users [68].

For these reasons, recent years have seen attempts to optimize network operations using the TinyML paradigm, which enables machine learning on embedded edge devices with minimal processing power and memory. TinyML faces significant challenges due to limited hardware resources, including processing power, memory capacity, and clock speed. Typically, it operates on 32-bit microcontrollers

(MCUs) with less than 500 kB of on-chip SRAM, few megabytes of on-board flash memory, and clock frequencies below 200 MHz [69]. In TinyML, the deployment and training of neural models typically follow a specialized workflow tailored to the constraints and requirements of MCU-based environments. Specifically, neural models are often trained on servers and cloud, and the trained models are then deployed on edge devices to optimize the speed of the inference phase [70] [71]. Libraries such as TensorFlow Lite and TorchScript have been developed to facilitate the deployment of machine learning models on microcontrollers. These libraries allow models trained on a PC using TensorFlow or PyTorch to be converted into a format suitable for microcontroller deployment [72] [73]. Despite the mature theoretical framework of TinyML, the application of Neural Network (Neural Network (NN))-based techniques for RUL prediction on edge devices remains relatively limited in the literature. Athanasakis et al. [69] deploy several machine learning models, namely LSTM, CNN, XGBoost, and Random Forest on an STMF767ZI microcontroller to predict the RUL of turbofan engines. They compared the impact of model optimization techniques, such as pruning and quantization, in terms of RUL evaluation and footprint metrics. Adducul et al. [74] present a DL approach called Edge-Based RUL Estimation (EBRULE) for battery RUL prediction on edge devices. They trained four neural network architectures on NASA and CALCE Lithium-ion battery datasets using a GPU, with inference performed on a Raspberry Pi. Ren et al. [75] propose a binary neural network, called BTCAN, to predict the RUL in industrial edge applications, along with a hardware implementation of BTCAN on a field-programmable gate array (FPGA) device. However, to the best of our knowledge, no study has implemented a similarity-based RUL prediction approach on resource-constrained devices. Additionally, most existing studies rely on platforms such as Raspberry Pi boards, FPGAs, or smartphones, which are relatively high-cost and high-power compared to MCUs [76].

This work aims to fill this gap by developing and deploying lightweight RUL prediction models on constrained ultra-low-power MCUs, extending the multi-class framework proposed in our previous study [77] and described in Chapter 3.

In [77], and as detailed in Section 3.1.2, the adopted HI estimator is a simple linear regressor trained offline, chosen for its minimal number of parameters to facilitate deployment on resource-constrained edge devices. In this extended work, we propose a novel lightweight neural network to evaluate whether a more complex model can improve predictions while maintaining implementability on edge devices. Additionally, we introduce a novel alternative RUL estimation block based on degradation rather than similarity, which does not require an HI library for the final RUL estimation. Degradation trend analysis is performed when safety thresholds are defined [15] [16]. Note that, in our framework, an operational threshold can be inferred directly from run-to-failure data, eliminating the need for prior knowledge of sensor-data thresholds.

The main contributions of this work can be summarized as follows:

1. A novel neural network architecture is proposed for the online prediction of the HI curve, designed for deployment on resource-constrained systems with low computational and memory requirements.
2. A degradation-based RUL estimator is developed, directly tuned on HI values and designed as an alternative to the similarity-based approach.
3. All the investigated solutions are deployed on a platform based on an MCU equipped with an ARM Cortex-M7 processor, in order to evaluate their execution time and memory requirements for both the HI estimation and RUL estimation stages.

This chapter is structured as follows. Section 4.1 presents the extended mathematical model, while Section 4.2 details the experimental setup adopted for the analysis. Section 4.3 presents and discusses the numerical results. Finally, the main findings are summarized in the conclusion.

4.1 Mathematical model

This work extends the multi-class branch of the RUL prediction framework proposed in [77] and detailed in Chapter 3, which accounts for the existence of C distinct failure modes within the same system. As shown in Section 3.2.3 the multi-class formulation enhances RUL estimation accuracy while significantly reducing the complexity of the similarity-based model, as it limits the number of required comparisons by approximately a factor of C.

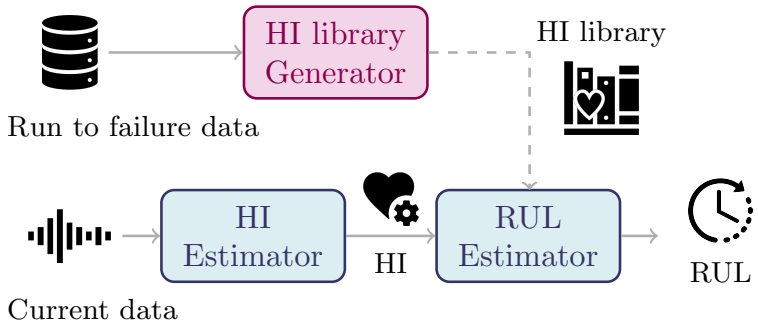

Figure 4.1: Connection of the main blocks of the framework for RUL estimation: the HI library generator (trained offline), the HI estimator (trained offline and deployed online), and the RUL estimator (used in inference). The similarity-based RUL estimator uses the HI library (dashed line), while the degradation-based RUL estimator operates independently of it.

The proposed framework maintains the same overall structure detailed in Section 3.1, consisting of three main functional blocks: the HI library generator, the HI estimator, and the RUL estimator. The general architecture is illustrated in Fig. 4.1, while the following paragraphs describe the extensions introduced in this work.

HI library generator. This block remains unchanged with respect to the version described in Section 3.1.1. It is responsible for generating a reference library of Health Indicator (HI) trajectories.

HI estimator. The HI Estimator block follows the same structure described in Section 3.1.2. In this work, we propose an alternative implementation based on a NN regressor that maps each data sample $\boldsymbol{x}_t$ into its corresponding HI value. While the previously adopted linear regressor is extremely lightweight and well suited for real-time applications, it often sacrifices estimation accuracy, which can lead to higher RUL prediction errors. To explore the trade-off between computational effort and estimation fidelity, the proposed NN-based mapping enhances the accuracy of HI estimation while remaining compatible with deployment on resource-constrained edge devices. Following the multi-class approach described in Section 3.1, a separate NN-based regressor is trained for each failure class, instead of relying on a single model.

RUL estimator. The similarity-based RUL estimation method follows the same approach discussed in Section 3.1.3. Accordingly, the final RUL estimate is computed as the similarity-weighted average of the tentative estimates $\hat{\mathrm{RUL}}_{j,\tau_j}(\boldsymbol{X})$ over the set $\mathbb{S}$, leading to the final formulation:

$$\hat{\mathrm{RUL}}(\boldsymbol{X}) = \frac{\sum_{(j,\tau_j)\in\mathbb{S}} \mathrm{Sim}_{j,\tau_j}(\boldsymbol{X}) \times \hat{\mathrm{RUL}}_{j,\tau_j}(\boldsymbol{X})}{\sum_{(j,\tau_j)\in\mathbb{S}} \mathrm{Sim}_{j,\tau_j}(\boldsymbol{X})} \tag{4.1}$$

Fig. 4.2 illustrates an example of RUL estimator based on similarity metrics performed after the novel NN-based HI Estimator. The red curve, denoted as $\hat{\boldsymbol{h}}$, represents the HI curve predicted by the NN-based regressor up to T. The blue lines indicate the selected set of the most similar HI curves $\hat{\boldsymbol{h}}_j$ from the library. These selected curves provide the tentative RUL estimates contributing to the final RUL estimate. The predicted failure time (dashed blue line) is shown alongside the real one (dashed red line).

In this work, we also consider a degradation model [78, 15, 79] as an alternative approach for RUL estimation. A degradation model estimates RUL by projecting the HI trend into the future using methods like curve fitting without relying on comparisons with an HI library. As a result, it requires less storage space and may lead to a lower computational complexity. In particular, we here consider

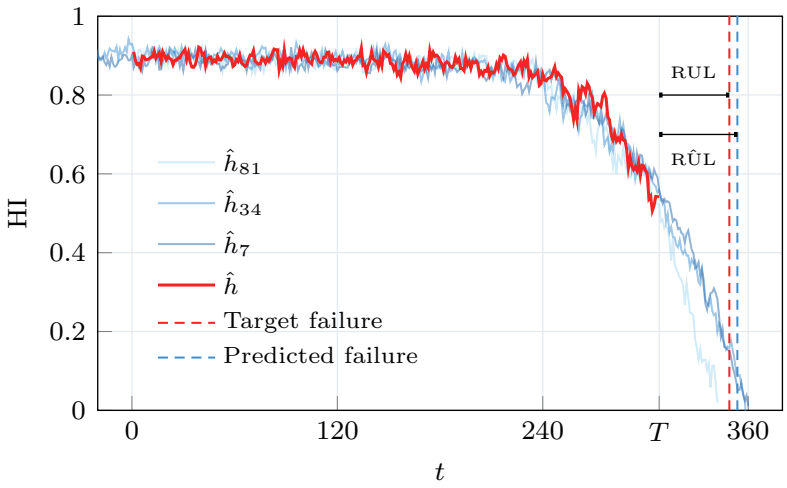

Figure 4.2: RUL estimation based on the similarity model described by (3.12), where the set $\mathbb{S}$ consists of the instances 81, 34, and 7, along with their respective time lag values.

polynomial fitting so that the current HI curve $\hat{\boldsymbol{h}}$ is fitted with a p-degree polynomial characterized by $p+1$ coefficients $a_0, a_1, \ldots, a_p$ such that:

$$\hat{h}_t \simeq a_0 + a_1 t + a_2 t^2 + \cdots + a_p t^p = \sum_{i=0}^{p} a_i t^i \tag{4.2}$$

The vector of polynomial regression coefficients must be computed online from the current HI curve $\hat{\boldsymbol{h}}$ using ordinary least squares estimation. Then, we estimate RUL as the smallest positive real root of the polynomial with the Newton-Raphson method.

Fig. 4.3 shows an example of polynomial fitting with $p = 4$ and the process of RUL estimation. The blue curve $\hat{\boldsymbol{h}}$ represents the HI curve predicted by the NN-based regressor and the light blue line shows its polynomial approximation projected in the future to estimate the failure cycles. The degradation-based approach is most effective when degradation is pronounced, as the polynomial can better approximate the degradation trajectory.

4.2 Experimental setting

In this section, we detail the experimental use case employed to compare the proposed approach. The implementation of the core building blocks for both the reference and proposed frameworks and the development environment for online deployment are described.

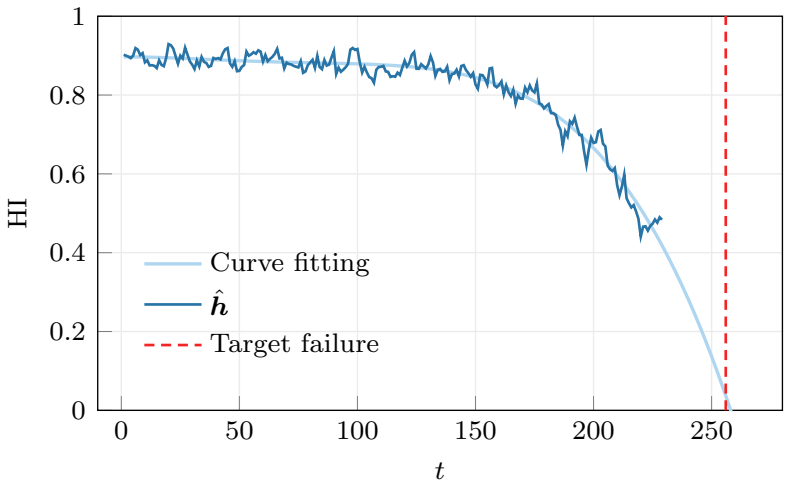

Figure 4.3: Degradation model applied to the current estimation $\hat{\boldsymbol{h}}$, where the estimated profile corresponds to a 4th order fitting function.

4.2.1 Models and parameters

In this work, we consider as a reference the framework proposed in [77], with a multi-class linear regressor as an HI estimator and the similarity model as an RUL estimator that we will call LIN+SIM. The two alternative solutions proposed here are:

- NN+SIM which replaces the linear regressor with one based on a neural network;
- NN+DEG which also replaces the similarity model with the degradation model as a RUL estimator.

We neglect the discussion of LIN+DEG because it performs significantly worse than the abovementioned solutions. The reason resides in the linear regression that fails to provide a sufficiently accurate estimation of the online HI, thus making the polynomial fitting task harder.

Each solution includes an AE trained offline to build the HI Library Generator block and a classifier that operates online to process the input time series $\boldsymbol{X}$, identify the active failure mode, and select the corresponding regressor. Both architectures were described in Section 3.2.2.

The time series classification permits the selection of the regressor that maps the time series $\boldsymbol{x}$ into an HI curve $\hat{\boldsymbol{h}}$. In the multi-class reference scenario, the regressors are linear so they are characterized by the coefficient vectors $\boldsymbol{\theta}_0$ and $\boldsymbol{\theta}_1$ that are obtained offline by minimizing the least square errors as described in Section 3.1.2. A linear regressor processes one input $\boldsymbol{x}_t$ at a time and returns an HI $\hat{h}_t$. Processing

all inputs in $\boldsymbol{x}$, we obtain the HI curve $\hat{\boldsymbol{h}}$.

Differently, the NN-based regressors proposed in this work take a window $\boldsymbol{W}_t$ with dimensions $n \times m$ as input (exactly as the classifier) and return an HI $\hat{h}_t$ (exactly as the linear regressors). A regressor of this kind therefore generates the HI curve $\hat{\boldsymbol{h}}$ by processing all windows $\boldsymbol{W}_t$. This regressor is trained to minimize the MSE between h_t and $\hat{h}_t$.

Table 4.1 provides a detailed overview of the structure and hyperparameters of the adopted neural networks, including the AE, the classifier, and the NN-based regressors.

Table 4.1: Structure and hyperparameters characterizing the adopted neural networks, namely the AE, the classifier (Class), and the NN-based regressors (Reg).

Model	Layer Type	Filters	Kernel Size	Stride	Output Size	Activation Function
AE	**Encoder**					
	Input	-	-	-	6×14	-
	Conv1D	140	4	2	3×140	ReLU
	Dense	-	-	-	100	Linear
	Decoder					
	Input	-	-	-	100	-
	Dense	-	-	-	3×140	ReLU
	ConvT1D	140	4	2	6×14	Linear
Class	Input	-	-	-	6×14	-
	Conv1D	7	4	2	3×7	ReLU
	Dense	-	-	-	1	Sigmoid
Reg	Input	-	-	-	6×14	-
	Conv1D	7	4	2	3×7	ReLU
	Dense	-	-	-	1	Linear

For the RUL estimator block, we define the parameters of the similarity model for the LIN+SIM and NN+SIM approaches, as well as the polynomial order for the NN+DEG approach. We set the coefficient $\beta = 0.99$ in (3.10) while the coefficient λ in (3.5) is set to 0.1 for the reference framework LIN+SIM and 0.05 for NN+SIM. This difference arises because the NN-based regressor produces less noisy HI curves, which allow for a smaller relaxation factor.

Regarding the degradation model, we select a polynomial order $p = 4$. This value is determined by finding the order that minimizes the RMSE between the predicted and actual RULs of all run-to-failure engine data. In this analysis, data are

truncated within the last 120 operational cycles to focus on the cycles approaching failures, as recommended in [80] [81]. Table 4.2 reports the RMSE for all tested polynomial orders and shows that $p = 4$ achieves the lowest RMSE.

Table 4.2: RMSE values for different polynomial orders.

Poly order p	RMSE
1	215589.0
2	297.2
3	49.6
4	28.3
5	44.9
6	59.4

4.2.2 Development environment

As an edge device, we chose the ATSAMV71/SAM V71 Xplained Ultra evaluation kit, which is a platform for developing applications with the ATSAMV71Q21 microcontroller. It features an ARM Cortex-M7 core operating at up to 300 MHz, along with 2 MB of Flash memory and 384 kB of SRAM. The ATSAMV71 board was used to evaluate the online phase of our framework and to assess both the feasibility of our framework and the computational performance of the proposed algorithms in terms of execution time and memory footprint.

The models are implemented using custom libraries organized as follows:

- *linalg*: a linear algebra module defining structures such as vectors, matrices, and 3-dimensional tensors, supporting various data types (float32, int16, int8). It also includes linear algebra operations, such as matrix multiplication.
- *dnn*: a module implementing the deep neural network layers to build the classifier and the NN-based regressor.
- *networks*: a module defining the classifier and the regressor architectures.
- *library*: a module containing the HI curves associated with each failure class and gathered in the HI library.
- *models_param*: a module containing all the parameters of the models, namely the linear and neural regressors associated with each failure class and the neural classifier.

Since computational performance is evaluated in terms of execution time and memory footprint, we consider two compiling optimization options: *(i)* aggressive optimization (-O3) that minimizes execution time; *(ii)* optimization for memory size

(-Os) that focuses on reducing the memory footprint. Both options were evaluated on the ATSAMV71 to determine the balance between speed and memory usage for the specific application.

4.3 Experimental evidence

The experiments were carried out in two phases. In the first phase, we implemented the reference and proposed frameworks in a Python environment based on PyTorch for design, training, and functional testing. Then, we deployed the blocks operating online (HI and RUL estimators) on the selected edge device for computational performance evaluation.

All experiments were conducted on sub-dataset FD003 of the C-MAPSS dataset, described in Section 3.2.1. FD003 was selected to validate the multi-class approach, as it includes multiple failure modes: HPC and fan degradation, labeled as $c = 1$ and $c = 0$, respectively.

4.3.1 Regression performance

We first evaluate the quality of the linear and neural regressors, trained on offline HI targets obtained using the AE, by comparing their reconstructed HI values to these targets. The discrepancy between the predicted and target HI values is measured using the Mean Squared Error (MSE). According to this metric, the neural model is 100% likely to outperform the linear model, as it demonstrated superior reconstruction performance across all units and both failure classes. Table 4.3 reports the average MSE calculated across all instances, separately for class 0, class 1, and the overall total. For failure class 0, the neural model achieves a 70.1% reduction in average MSE compared to the linear model. Similarly, for failure class 1, the neural model reduces the average MSE by 64.4%. Across the entire dataset, the neural regressor lowers the average MSE by 65.0% compared to the linear regressor, providing more accurate health index curve estimations for the test instances, at the cost of requiring more parameters.

Table 4.3: Regression performance evaluated using mean MSE for class 0, class 1, and for the overall dataset

	LIN REG	NEU REG
Class 0	8.7e-3 ±5.1e-3	2.6e-3 ±2.7e-3
Class 1	11.8e-3 ±6.0e-3	4.2e-3 ±2.4e-3
Total	10.0e-3 ±5.8e-3	3.5e-3 ±2.7e-3

4.3.2 Prediction performance

We compare the RUL estimation performance for the two solutions employing the similarity model as RUL estimator that are LIN+SIM and NN+SIM.

Fig. 4.4 displays RUL prediction values for test engines from dataset FD003. To enhance visual clarity, the engines are sorted by their target RUL. The figure shows RUL predictions generated by the novel neural regression-based method (green crosses) and the reference linear regression-based method (orange diamonds), alongside the actual target RUL values (gray dots).

As a quantitative analysis, Table 4.4 reports the RMSE (1.4), score S (1.5), and accuracy A (1.6) of both methods across different segments of the test set (with $a_1 = 13$ and $a_2 = 10$ for the C-MAPSS dataset, as conventionally adopted). For the first 10% of the test set, which includes engines with a short target RUL, both methods show optimal performance, achieving 80% and 100% accuracy, respectively. As the analysis extends to engines with longer RULs, performance metrics decline gradually. Nonetheless, the novel approach consistently surpasses the reference method, particularly in terms of score.

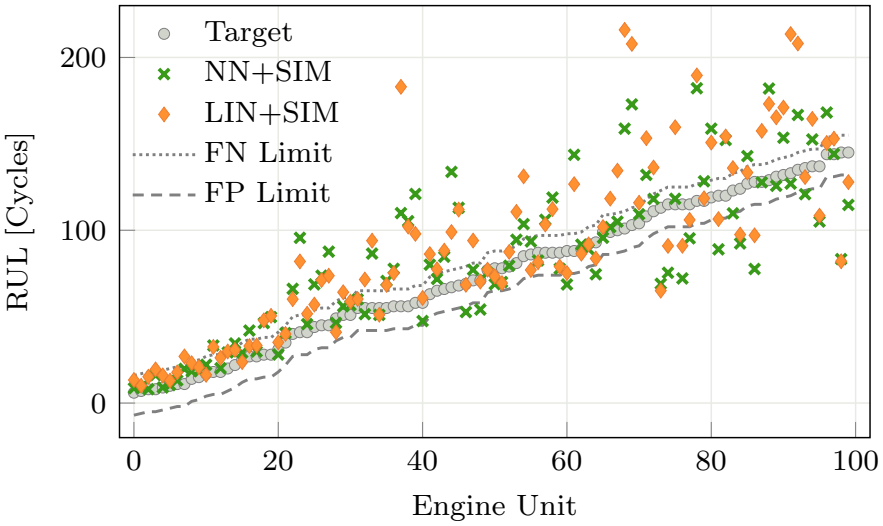

Figure 4.4: RUL predictions obtained using the multi-class estimators on dataset FD003: the one based on linear regression and the one based on the novel neural regression approach.

Regarding the degradation model, i.e., the NN+DEG solution, we must consider that this RUL estimator is most effective when degradation is pronounced. For machines that are still in good condition, i.e., when the HI curve degradation is minimal, the degradation model proves ineffective in predicting future trends and, consequently, the RUL of the machine. Therefore, we have chosen to evaluate the

Table 4.4: Performance comparison

Test set Portion	LIN + SIM RMSE	S	A	NN + SIM RMSE	S	A
10%	8.5	13.2	80.0%	4.4	4.5	100.0%
20%	10.4	38.0	75.0%	9.4	30.4	75.0%
40%	26.7	328k	52.5%	22.3	1312	60.0%
60%	25.0	328k	51.7%	22.3	2256	56.7%
80%	31.4	464k	47.5%	25.7	4695	52.5%
100%	32.6	469k	42.0%	26.7	5227	46.0%

model's performance only for those machines for which the HI curve reaches values below a determined threshold. Table 4.5 presents the performance for a decreasing HI threshold. As the threshold value decreases, the model's performance improves, reaching 100% accuracy for threshold value equal to 0.5.

Table 4.5: Degradation model performance

Threshold	RMSE	S	A
0.8	19.8	4286.6	68.2%
0.7	9.4	51.5	79.4%
0.6	5.1	11.1	95.5%
0.5	3.6	6.4	100.0%

4.3.3 Alpha - Lambda performance

To assess the performance of the models over time, we employ alpha-lambda ($\alpha-\lambda$) plots [37, 38, 39] described in Section 1.2.4. These plots visualize the predicted RUL against the actual remaining cycles to failure, providing insight into the models' accuracy at different stages of the component's degradation. We compare the alpha-lambda plots of the NN+SIM and NN+DEG approaches, which differ in the RUL estimator block.

To evaluate the accuracy of the predicted RUL at a specific time instance, t_λ, we use the $\alpha - \lambda$ metric (1.7) setting $\alpha = 0.2$ (20%), as suggested by [40].

Since the C-MAPSS dataset provides only truncated test instances with unknown progression to failure and only the RUL values are available, we derive $\alpha - \lambda$ plots from run-to-failure instances exclusively drawn from the training set. We employed a k-fold cross-validation strategy to enhance assessment robustness. This strategy splits the dataset into k subsets (or *folds*). Then, the model is iteratively trained on $k-1$ folds and evaluated on the one left out, ensuring that the model's performance

is not overestimated due to data leakage or overfitting. From an initial set of 100 test instances (44 from class 0 and 56 from class 1), we generated 11 folds. Each fold contained a randomly chosen subset of 9 test instances (4 from class 0 and 5 from class 1), with the remaining 91 instances used for training. We trained the AE and the class-specific regressors for each fold.

Fig. 4.5 presents the $\alpha - \lambda$ plots for three test units, Unit 20, Unit 71, and Unit 90, obtained using the two NN-based model variations: NN+SIM and NN+DEG for $\alpha = 20\%$. These plots are limited to the final 120 cycles preceding failure because it was considered reasonable to neglect predictive capabilities for cycles beyond this range, as recommended in [80] [81]. Each curve is displayed with a color gradient reflecting the HI value, with green indicating good health and red indicating failure. This visualization provides a clearer interpretation of the model's ability to track RUL estimates within the specified threshold range.

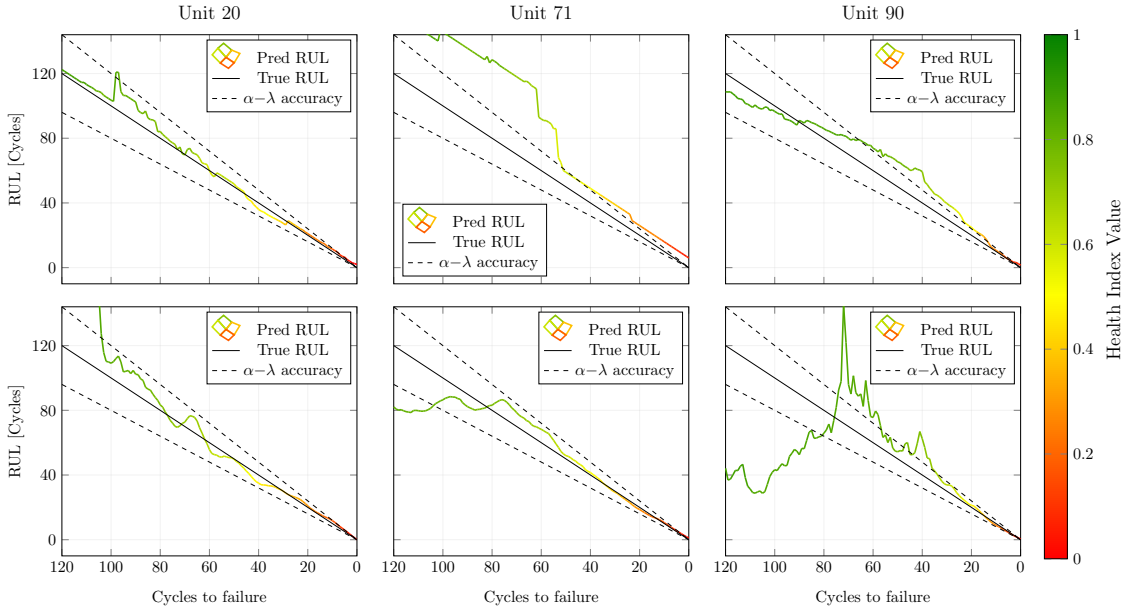

Figure 4.5: $\alpha - \lambda$ plots for three test units (Unit 20, Unit 71, and Unit 90), selected from k-fold cross-validation, generated using the two NN-based model variations: NN+SIM (top line) and NN+DEG (bottom line) for $\alpha = 0.2$. The plots are restricted to the final 120 cycles before end of life.

Table 4.6 reports the $\alpha - \lambda$ accuracy, computed across all test units, for $\alpha = 0.2$ at different time intervals before failure. The cycles to failure were discretized into bins of 20-cycle width to analyze how the $\alpha - \lambda$ accuracy varies as the system approaches the end of life.

The similarity-based model demonstrates higher $\alpha - \lambda$ accuracy in the early stages of degradation but exhibits a decline in performance as failure approaches, where

Table 4.6: Performance measured in terms of $\alpha-\lambda$ accuracy ($\alpha = 0.2$). The results are reported from the bins corresponding to cycles closest to failure to those at earlier stages of degradation.

Cycles to failure	NN + SIM	NN + DEG
bin range	$\alpha-\lambda$ (%)	$\alpha-\lambda$ (%)
$0-20$	22.3	37.6
$20-40$	28.7	47.0
$40-60$	35.6	36.7
$60-80$	49.5	30.3
$80-100$	52.8	22.7
$100-120$	49.3	12.5

higher precision is required. It is important to consider that the current performance may be influenced by the limited size of the reference library, further constrained by the classification process and k-fold cross-validation, which restrict the diversity of available failure patterns. As an example, the similarity-based model in Unit 71 struggles to find sufficiently similar patterns within the reference library and ends up failing RUL estimation.

Conversely, the degradation-based model proves more reliable when degradation becomes more pronounced but struggles in the early stages. This limitation stems from the polynomial fitting process, which does not always yield positive real solutions, preventing the model from consistently generating valid RUL estimates.

We conducted an uncertainty quantification analysis to investigate the relationship between RUL predictions and the corresponding HI values. Associating uncertainty with HI enables direct application in real-time settings, where both the HI value and the estimated RUL are continuously available. This approach facilitates an immediate assessment of confidence in RUL predictions, enhancing their reliability and interpretability in online predictive maintenance (PdM) frameworks.

The HI was discretized into bins of width 0.1, enabling a detailed analysis of the prediction error distribution across different HI ranges. Table 4.7 reports the mean, median (med), and the 10th (P10) and 90th (P90) percentiles of all the prediction errors computed when the HI reaches certain values for the NN+SIM and NN+DEG models, with a focus on the critical HI range of 0–0.6. The results demonstrate that NN+DEG outperforms NN+SIM within this range, achieving lower mean prediction errors and standard deviation across all bins. However, it is important to note that the NN+DEG model becomes less effective for higher HI values, where degradation trends are less evident.

From a maintenance perspective, narrower percentile ranges correspond to higher

confidence in RUL estimates and support more informed, risk-aware decision-making. In particular, given the adopted error definition (1.3), upper-bound error percentiles (e.g., P90) represent worst-case RUL overestimation scenarios and can be exploited to define conservative scheduling strategies, thereby reducing the risk of unexpected failures in safety-critical applications.

Table 4.7: Prediction error statistics: mean, median (med), P10, P90 in cycles for the NN+SIM and NN+DEG models, computed at specific HI bin values within the critical range (0–0.6).

HI	NN + SIM				NN + DEG			
bin	mean	med	P10	P90	mean	med	P10	P90
$0.0 - 0.1$	1.8	2.0	-2.0	5.9	0.7	0.7	-1.5	2.8
$0.1 - 0.2$	2.0	2.4	-2.3	6.0	1.3	1.3	-1.7	4.8
$0.2 - 0.3$	2.1	2.9	-6.0	8.3	2.1	1.9	-2.3	6.1
$0.3 - 0.4$	3.1	4.0	-6.0	10.8	2.7	2.6	-1.8	7.5
$0.4 - 0.5$	4.0	5.3	-6.0	12.5	4.1	3.7	-4.4	11.1
$0.5 - 0.6$	4.9	7.2	-12.0	18.3	4.1	5.1	-8.2	15.7

4.3.4 Computational performance

We assess computational performance in terms of execution time and memory footprint. The memory footprint of the three algorithms is reported in Table 4.8 while execution time is in Table 4.9.

Table 4.8 presents the program memory usage for optimization levels -Os and -O3, stored in Flash, along with data memory usage, which is allocated in RAM. This differentiation highlights the storage requirements for executable code and runtime data, respectively. We additionally report the contribution of parameters within the program memory to highlight the relevance of models and library sizes in memory usage.

Table 4.8: Memory footprint

	LIN + SIM	NN + SIM	NN + DEG
Program -Os	111.9 kB	113.4 kB	24.5 kB
Program -O3	113.5 kB	114.9 kB	26.5 kB
of which Param	101.5 kB	102.8 kB	5.1 kB
Data	13.8 kB	13.8 kB	18.1 kB

The optimization level -Os reduces program memory usage across all models, with a maximum reduction of 7.5% for the NN+DEG model. The NN+DEG model

requires the least on-chip memory among all models, as the degradation approach does not need libraries to estimate RUL. For similarity-based models, the novel NN+SIM approach uses 1.19% more total memory than the reference LIN+SIM model, primarily due to the neural regressors having a higher parameter count compared to the linear ones.

Table 4.9 presents the average and maximum execution times for the three methods, both in the -Os optimization setting and in the -O3 setting. The table separates the contributions of the classification, regression, and RUL estimation blocks.

Table 4.9: Execution time

	Classifier		Regressor		RUL Estimator		Total	
	mean	max	mean	max	mean	max	mean	max
-Os								
LIN + SIM [ms]	7.27	21.24	0.06	0.18	20.87	35.71	28.21	42.60
NN + SIM [ms]	7.27	21.24	6.04	17.64	20.01	34.60	33.40	47.95
NN + DEG [ms]	7.27	21.24	6.04	17.64	9.92	28.97	23.23	67.85
-O3								
LIN + SIM [ms]	6.80	19.87	0.06	0.17	20.50	35.21	27.37	41.63
NN + SIM [ms]	6.80	19.87	5.59	16.32	19.72	34.09	32.12	46.55
NN + DEG [ms]	6.80	19.87	5.59	16.32	9.64	28.15	22.03	64.34

The classifier, the regressors, and the degradation model for RUL estimation exhibit execution times that increase linearly with the length of the test series. In contrast, the RUL estimator based on the similarity model depends not only on the test series length but also on the parameter τ and the number of library instances selected for the RUL computation in (3.12). The longer the input time series, the fewer the HI curves in the library for which we can compute the squared distance and the fewer the possible time lags τ. As a result, the relationship between the series length and the execution time of the RUL estimation block based on similarity is shown in Fig. 4.6 for both classes of failure.

As shown in Table 4.9, the -O3 optimization consistently results in lower time measurements compared to -Os, with a minimum improvement of 3% and a maximum of 5.16% on the total mean measurements, and a range of 2.27% to 5.17% for the total max measurements. Considering the results of the -O3 optimization, the lowest average execution times are achieved with the NN+DEG model, showing a 20% improvement compared to the LIN+SIM model. Among the two similarity-based models, the novel NN+SIM approach increases execution time by 17.4% due to the introduction of the neural regressor block. Regarding the maximum execution

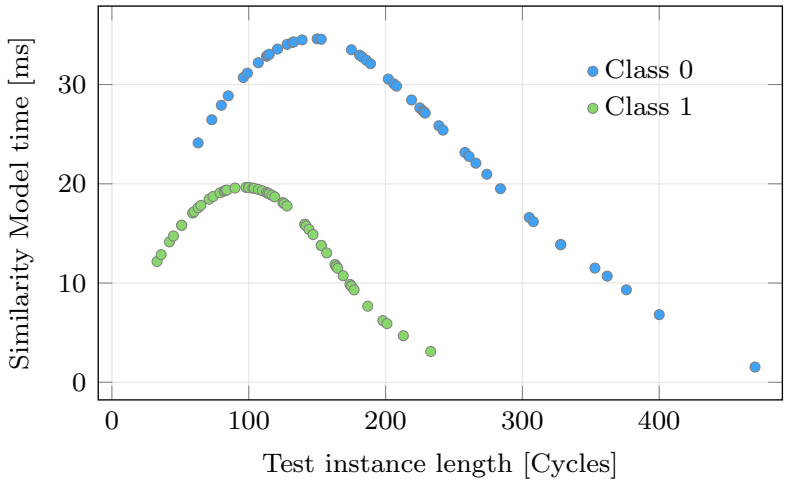

Figure 4.6: Execution time of the RUL estimation block based on the similarity model as a function of the test instance length.

values, the highest execution time is recorded in the NN+DEG model, corresponding to the longest test instance, since all its blocks scale linearly with the length. In contrast, the maximum total execution times for the similarity-based models do not occur with the longest instance, due to the non-linear behavior of the RUL Estimator block shown in Fig. 4.6. As illustrated in the figure, longer test instances correspond to shorter execution times for the RUL estimator block based on similarity metrics. This is due to the fact that the longer the input time series, the fewer the library instances available for comparison with the test HI.

4.4 Conclusion

An effective prediction of a machine's RUL leads to higher reliability and lower maintenance costs. These advantages are further enhanced when the algorithms run locally on low-cost, resource-limited devices such as microcontroller-based platforms.

In this work, we started with a state-of-the-art multi-class approach that leverages the different signatures of the failures and is based on an offline AE and an online linear regressor followed by a similarity model. We extended this approach by proposing an alternative regressor based on neural networks and a different RUL estimator based on a degradation model.

The experiments focused on the C-MAPSS turbofan engine dataset and the involved methods were implemented on a microcontroller equipped with an ARM Cortex-M7 processor. The experimental evidence showed that the neural regressor led to a

significant and consistent improvement in the RUL estimation accuracy, up to 20% in the case of machines close to failure. The degradation model allows for a drastic reduction in memory usage as it does not require the storage of parameters such as the HI library for the similarity model. In addition, the RUL estimator based on the degradation model allows for a reduction in the execution time of about 50% in the average case and 17% in the worst case.

In conclusion, the NN+SIM model outperforms the LIN+SIM model, making it the preferable choice due to its better accuracy and the minimal trade-off in terms of increased resource usage (1.19% more on-chip memory and 17.4% higher mean execution time in the -O3 optimization).

The NN+DEG model, while not always providing RUL estimates for high HI values, represents a good option when the primary goal is to minimize resource usage. Specifically, in the -Os optimization, it reduces memory footprint by 66.5% and lowers average execution time by 30.4% compared to the NN+SIM model. Additionally, it exhibits superior predictive performance in the critical HI range of 0–0.6 and during the final 40 cycles preceding failure, where high accuracy is essential for reliable prognostics.

Part II

Anomaly Detection for System Diagnostics

Chapter 5

Overview of Anomaly Detection

The ability to detect deviations from expected operating behavior is a cornerstone of modern PHM systems. Within this context, Anomaly detection (AD) serves as an early-warning mechanism, enabling the timely identification of abnormal conditions that may precede component degradation or system-level faults [82]. Timely detection of anomalies can prevent catastrophic outcomes, making AD a crucial tool in ensuring the reliability and safety of complex systems.

In this chapter, we introduce the fundamental concepts of anomaly detection in time series, including the classification of anomaly types, the main detection methodologies, and the performance metrics used for evaluation.

5.1 Anomaly detection on time series

In PHM applications, a network of sensors enables the continuous monitoring and real-time processing of system data. Data collected and recorded over time are referred to as time series, which capture the temporal evolution of physical quantities describing the system's behavior. Analyzing these data is fundamental for detecting deviations from expected operating conditions and anticipating potential faults. In a monitoring scenario, the state of the system can be represented by samples of a generic quantity $\boldsymbol{x}[t]$ collected at discrete time instants t. Each sample reflects the system condition at that specific time. In the context of AD, $\boldsymbol{x}[t]$ is considered a realization drawn from one of two possible sources: the normal operating condition, denoted by $\boldsymbol{x}^{\text{ok}}$, or an anomalous condition, denoted by $\boldsymbol{x}^{\text{ko}}$.

These two sources are modeled as stationary, discrete-time stochastic processes that generate independent and identically distributed (i.i.d.) n-dimensional vectors $\boldsymbol{x}^{\text{ok}}, \boldsymbol{x}^{\text{ko}} \in \mathbb{R}^n$, each described by its probability density function (PDF), $f_{\boldsymbol{x}}^{\text{ok}} : \mathbb{R}^n \to \mathbb{R}^+$ and $f_{\boldsymbol{x}}^{\text{ko}} : \mathbb{R}^n \to \mathbb{R}^+$. Thus, at any time t, the observed sample can be expressed

as $\boldsymbol{x}[t] = \boldsymbol{x}^{\text{ok}}[t]$ or $\boldsymbol{x}[t] = \boldsymbol{x}^{\text{ko}}[t]$. Since the samples are i.i.d., the explicit time index can be omitted.

The objective of an anomaly detector is to discriminate between normal instances $\boldsymbol{x}^{\text{ok}} \sim f_{\boldsymbol{x}}^{\text{ok}}$ and anomalous ones $\boldsymbol{x}^{\text{ko}} \sim f_{\boldsymbol{x}}^{\text{ko}}$. Given an input $\boldsymbol{x}$, the detector outputs either anomaly scores or binary labels [83]. Scoring-based methods assign a numerical score to each instance, reflecting its degree of anomalousness, whereas binary labeling methods classify each instance as either anomalous or normal. The scoring function $z(\boldsymbol{x})$ is typically designed such that higher scores correspond to greater anomalousness.

In the literature [84], types of anomalies in univariate and multivariate time series are categorized into the following three groups:

1. **Point anomalies**: A point anomaly is an individual data instance that deviates significantly from other values in the entire time series (global outlier) or from its neighboring points (local outlier). In the field of time series data, point outlier detection is the most common outlier detection task [85].

2. **Contextual anomalies**: A contextual anomaly, also known as a conditional anomaly, is a data instance that behaves anomalously within a specific context or subset of the data, but not necessarily in the overall dataset. Contextual anomalies can be difficult to detect, as they may not stand out when the entire dataset is considered, but they can have significant impacts on the results of analysis and decision-making. Contextual anomalies are common in time-series data: for example, for what concerns the atmospheric temperature time-series, a value of 30°C during summer is normal in many locations, while the same temperature in winter would be abnormal [86].

3. **Collective anomalies**: A collection of similar data instances that behaves anomalously with respect to the entire dataset is termed a collective anomaly. For example, a prolonged period of low values in an Electrocardiogram (ECG) may indicate a premature contraction, which is a type of abnormal heartbeat [87]. However, a single low value on its own is not necessarily unusual or cause for concern.

5.1.1 Sensor and system faults

Sensor networks provide a detailed representation of physical phenomena and system behaviors. However, to extract meaningful insights from the collected data, it is essential to ensure the quality and reliability of the measurements. Sensor faults are a common issue in networked monitoring systems and can compromise the validity of the acquired data, thus hindering the ability to draw accurate conclusions [88].

In general, two main categories of anomalies can be identified. The first category

Table 5.1: Examples of data corruptions and system alterations

Sensor faults	
Type	**Description**
Noise faults	Occur due to hardware failure, environmental conditions, or battery supply issues, resulting in unexpected high variance or noise in sensor values [89].
Short fault/spike	Manifest as an instantaneous increase in the rate of change of sensor values and is caused by hardware or connection failures [90, 91].
Clipping or saturation	Result in a clipping of the extreme values in sensor readings, either due to physical limitations of certain sensor types or calibration issues [92, 93].
Stuck-at faults	Refer to situations when the output shows low variance with samples concentrated around a constant value [94].
Constant	Due to improper calibration or drift of calibration parameters over time, sensor readings may manifest a consistent deviation by a constant value [95].
Narrow-band interference	Considers spurious narrow-band signals contaminating the band of the signal of interest [96].
Dead-zone	Is a non-linearity that manifests when a sensor or actuator fails to provide a non-null output value [97].
System alterations	
Type	**Description**
Aging	Occurs due to the natural and unavoidable variation of the physical properties of the system over time [98, 99].
Wearing	Alterations caused by degradation of the system's components due to repeated environmental or mechanical stimuli, e.g., erosion, friction [100].
Abrupt changes	Sudden variations in the system's behaviour. Examples of such alterations include damages in a civil structure (tendon/strand breakages [101] and earthquakes [102]); irregular heart rate, irregular rhythm, and ectopic rhythm during heart activity [103].

comprises sensor-related faults, as previously discussed, which stem from issues in the sensing devices and directly affect data quality. The second category includes system-related anomalies, or system alterations, which reflect genuine changes in the behavior of the monitored system or in the underlying physical processes. These alterations represent deviations from the nominal operating conditions and can have different impacts on system performance depending on the context. Examples of sensor faults and system anomalies observed in various monitoring scenarios are shown in Table 5.1.

5.2 Anomaly detection techniques

The problem of anomaly detection has been studied extensively and a variety of techniques to solve it have been proposed [84, 104]. Each of them tackles the problem in a different way, according to specific assumptions made on the normal and/or anomalous data. These assumptions imply that the performance of each approach highly depends on the nature of both normal and anomalous data. Moreover, depending on the adopted assumption the available techniques can be split into these main categories:

5.2.1 PCA-based methods

Principal Component Analysis (PCA) [105] is a statistical technique used to represent data in a reduced subspace of dimension $k < n$, where most of the signal variance is preserved. When this subspace is constructed using only data corresponding to normal operating conditions, deviations from it can be interpreted as potential anomalies.

The construction of the PCA model begins with the estimation of the covariance matrix from the normal samples[1]:

$$\boldsymbol{\Sigma}^{\mathrm{ok}} = \mathbf{E}\left[\boldsymbol{x}^{\mathrm{ok}}\boldsymbol{x}^{\mathrm{ok}\top}\right]. \tag{5.1}$$

A spectral decomposition of $\boldsymbol{\Sigma}^{\mathrm{ok}}$ gives

$$\boldsymbol{\Sigma}^{\mathrm{ok}} = \boldsymbol{U}^{\mathrm{ok}}\boldsymbol{\Lambda}^{\mathrm{ok}}\boldsymbol{U}^{\mathrm{ok}\top} \tag{5.2}$$

where $\boldsymbol{U}^{\mathrm{ok}}$ is an $n \times n$ orthonormal matrix containing the eigenvectors of $\boldsymbol{\Sigma}^{\mathrm{ok}}$, and $\boldsymbol{\Lambda}^{\mathrm{ok}}$ is a $n \times n$ diagonal matrix whose entries are the corresponding eigenvalues, ordered such that $\lambda_0^{\mathrm{ok}} \geq \lambda_1^{\mathrm{ok}} \geq \cdots \geq \lambda_{n-1}^{\mathrm{ok}} \geq 0$.

The eigenvectors associated with the k largest eigenvalues define the *principal subspace*, represented by the matrix $\boldsymbol{U}_k^{\mathrm{ok}}$, which contains the first k columns of $\boldsymbol{U}^{\mathrm{ok}}$. The remaining $n - k$ eigenvectors span the *residual subspace*, which captures variations not explained by the main components.

Based on this formulation, two complementary PCA-based anomaly detectors can be defined [106]:

- Squared Prediction Error (SPE_k): This detector evaluates the portion of the signal that lies outside the principal subspace. The anomaly score is computed

[1] Without loss of generality, we assume the mean vector of $\boldsymbol{x}^{\mathrm{ok}}$ to be null, i.e., $\mathbf{E}\left[\boldsymbol{x}^{\mathrm{ok}}\right] = 0$. If the mean is not zero, the data can be centered by subtracting the mean.

as:

$$z(\boldsymbol{x}) = \left\| \boldsymbol{x} - \boldsymbol{U}_k^{\mathrm{ok}} \boldsymbol{U}_k^{\mathrm{ok}\top} \boldsymbol{x} \right\|^2 . \tag{5.3}$$

- Hotelling's T-squared statistic (T_k^2): This detector instead focuses on deviations that occur *within* the principal subspace. The score is defined as:

$$z(\boldsymbol{x}) = \left\| (\boldsymbol{\Lambda}_k^{\mathrm{ok}})^{-1/2} \boldsymbol{U}_k^{\mathrm{ok}\top} \boldsymbol{x} \right\|^2 , \tag{5.4}$$

  where $\boldsymbol{\Lambda}_k^{\mathrm{ok}}$ denotes the $k \times k$ upper-left submatrix of $\boldsymbol{\Lambda}^{\mathrm{ok}}$.

5.2.2 GD-based methods

Methods based on the Gaussian Distribution (GD) assume that normal data samples follow a multivariate Gaussian distribution. Two commonly used detectors relying on this assumption are described below.

- Mahalanobis Distance (MD) [84]: This method measures the distance of a sample $\boldsymbol{x}$ from the center of the estimated Gaussian distribution, accounting for feature correlations and their corresponding variances. The anomaly score is defined as the Mahalanobis distance:

$$z(\boldsymbol{x}) = \left\| (\boldsymbol{\Lambda}^{\mathrm{ok}})^{-1/2} \boldsymbol{U}^{\mathrm{ok}\top} \boldsymbol{x} \right\| . \tag{5.5}$$

- **Autoregressive model ($\mathbf{AR}_p$)** [83]: This method captures the temporal dependencies within a signal by fitting a linear regression model that predicts future samples from past observations. During testing, anomalies are identified by evaluating the prediction error, i.e., the deviation between the predicted and actual values. For a model of order $p \in [1, n)$, the anomaly score is computed as:

$$z(\boldsymbol{x}) = \frac{1}{n-p} \sum_{i=0}^{n-p-1} (x_{p+i} - \tilde{x}_i)^2, \tag{5.6}$$

  where $\tilde{\boldsymbol{x}}$ contains the predicted values for the last $n-p$ samples of $\boldsymbol{x}$, estimated from the first p samples.

Both MD and AR-based detectors are simple and computationally efficient, but their performance may degrade in complex, real-world scenarios where the data distribution deviates significantly from the Gaussian assumption.

5.2.3 Learning-based methods

Machine learning (ML)

Machine Learning (ML)-based methods identify anomalies by learning patterns directly from the data, without assuming a specific underlying distribution. Three representative approaches are briefly recalled below.

- Local Outlier Factor (LOF_h) [107] This method relies on the concept of local density deviation. It defines the anomaly score of a sample $\boldsymbol{x}$ as the ratio between the average density of its h nearest neighbors and its own local density. Points that are substantially less dense than their neighbors are assigned higher scores.

- Isolation Forest (IF_q) [108] This approach is based on the assumption that anomalous samples are easier to isolate from the rest of the data. It builds an ensemble of q randomly generated trees that recursively partition the dataset. The anomaly score is inversely proportional to the average path length required to isolate a sample across the ensemble of trees.

- One-Class Support Vector Machine ($\text{OC}_{kernel,\nu}$) [109] This method learns a non-linear mapping that projects data into a higher-dimensional space where a hyperplane separates normal and anomalous samples. The score is computed as the distance of each point from the separating hyperplane, or equivalently from the origin in the transformed space. The main hyperparameters are the *kernel* function defining the mapping and the parameter ν, which specifies the expected fraction of support vectors in the training data.

Deep learning (DL)

Deep learning (DL)-based methods extend traditional anomaly detection approaches by leveraging NNs to model complex and non-linear data relationships. These techniques are particularly effective when the underlying data structure cannot be well represented through linear projections or simple statistical models.

A common approach is the Autoencoder (AE) [110], which can be regarded as a non-linear extension of the PCA model. The AE learns to reconstruct normal data by compressing the input into a lower-dimensional latent representation through an encoder network and then reconstructing it with a decoder. Anomalies are identified as samples that cannot be accurately reconstructed, since they deviate from the patterns learned during training.

5.2.4 Metrics

In binary anomaly detection, the decision process relies on comparing the detector's score $z(\boldsymbol{x})$ with a threshold that defines the boundary between normal and abnormal behavior. To evaluate the detector's performance independently of a specific threshold, a widely adopted criterion is the Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC), defined as:

$$\mathrm{AUC} = \mathbb{P}\Big[z(\boldsymbol{x}^{\mathrm{ko}}) > z(\boldsymbol{x}^{\mathrm{ok}})\Big] \tag{5.7}$$

The AUC $\in [0,1]$ quantifies the probability that a randomly selected anomalous sample $\boldsymbol{x}^{\mathrm{ko}}$ will receive a higher anomaly score than a randomly selected nominal one $\boldsymbol{x}^{\mathrm{ok}}$. An AUC value of 1 denotes perfect discrimination, whereas AUC = 0.5 corresponds to random guessing.

In Chapter 6, we present a framework for benchmarking anomaly detection algorithms operating on time-series data. The framework integrates a comprehensive set of synthetic anomalies designed to reproduce the diverse effects that real-world faults can induce on nominal signals, thereby enabling controlled and realistic evaluation of detector performance.

Chapter 6

A Framework for the Assessment of Time-Series Anomaly Detectors

Modern industry and engineering heavily rely on the collection and analysis of time-series data generated by sensors for monitoring, control, and optimization. In this context, anomaly detection (AD) plays a crucial role in detecting system malfunctions and failures in data chains, serving as a fundamental building block of PHM-driven maintenance and reliability strategies.

As highlighted in Chapter 5, the literature offers a wide range of AD algorithms [84], and the main challenge often lies in selecting the most suitable one for a given application. An effective evaluation of an anomaly detector, however, requires prior knowledge of the possible anomaly types and access to a sufficiently large number of their occurrences. In practice, this condition is rarely satisfied due to the scarcity of anomalous data, particularly during the system design phase. This issue can be framed within the broader problem of *model selection*, which consists of identifying the statistical model that best fits the available data [111, 112]. In the specific context of time-series AD, this topic has been systematically explored only in recent studies [112, 113, 114, 115], which highlight the potential of synthetic anomaly generation as a promising solution.

Nevertheless, adopting such a strategy requires: *i*) identifying the anomaly types most relevant to the application domain; *ii*) modeling and characterizing these anomalies; and *iii*) developing a synthetic data generation procedure. To date, the literature still lacks a unified framework capable of encompassing and automating all these steps.

This research direction has already been explored for image data, although in a different context than AD. The authors in [116] developed a framework to generate common image corruptions with varying intensities, aiming to assess the robustness

of classifiers.

In the case of time series, most studies have focused on specific sources of abnormality [89, 88, 117, 90], and only a few works have outlined a systematic procedure for anomaly generation [118, 119, 120, 121, 122, 112].

Several investigations have examined anomalies affecting monitoring devices. For instance, the studies in [89, 88, 117] analyze common sensor faults, such as spike, stuck-at, or low-battery errors, and classify them into categories like *short*, *noise*, and *constant*. Similarly, the authors in [90] model a limited set of faults to evaluate the detection capabilities of a specific class of detectors [109]. Conversely, the work in [122] focuses on the malfunctioning of the monitored system itself and introduces two anomaly models, namely *spectrum* and *basis*, describing the statistical deviation from normal behavior.

Alternatively, the studies in [118, 120, 119] decouple the anomalies from their physical source and classify them based on their effect on the signal. In particular, the authors in [118] model the normal signal as a combination of *seasonality* and *trend* components, generating anomalies as perturbations of these two factors. In contrast, [119, 120] adopt a more general formulation of the normal signal but restrict their analysis to a few anomaly types, such as *global*, *local*, and *dependency* anomalies.

Finally, the works in [121, 112] propose a broader collection of anomalies that capture several effects observed in real-world signals. Although this represents a significant step forward, the proposed sets remain non-exhaustive, and the corresponding frameworks lack a shared parameter linked to anomaly severity, an aspect that is crucial for enabling a comprehensive quantitative evaluation of detectors across different anomaly types.

In this chapter, we propose a framework based on synthetic anomalies for a systematic assessment of a detector. We address the issue of anomaly characterization by identifying the effects of the anomalies on the signal, enabling the development of models that are independent of the source. As a result, we set up a flexible testing suite, depicted in Fig. 6.1, for assessing anomaly detectors within any specific context. With historical data and domain-specific knowledge, designers can tailor this test suite to their applications.

This chapter is organized as follows. Section 6.1 introduces the models for both normal and anomalous time-series signals, which form the basis for the subsequent analysis. Section 6.2 presents the complete framework for anomaly detector selection. Anomalies are categorized according to their effect on the signal power, which may increase, remain invariant, or decrease. The first group includes superimposed disturbances, the second represents deviations that preserve power but alter the information content, and the third comprises nonlinear distortions that reduce power.

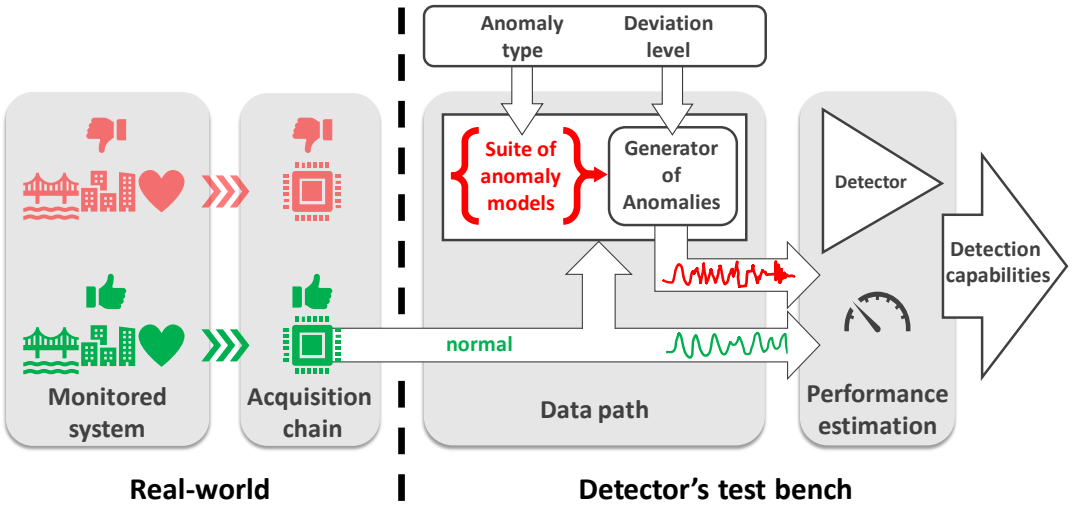

Figure 6.1: Framework that allows testing a detector aimed at identifying unknown anomalies coming from different sources.

To ensure consistency across anomaly types, each perturbation is parameterized by a single variable controlling the deviation from the normal signal. Section 6.3 details the setup of the numerical experiments and presents the corresponding results. The framework is validated on two real-world case studies: human health monitoring, using electrocardiogram (ECG) data, and structural health monitoring, using acceleration data (ACC). In both scenarios, signals are synthetically corrupted to create datasets for testing and evaluating different detectors. Finally, the main conclusions are drawn.

6.1 Models of normal and anomalous signals

In this section, we present the model and underlying assumptions used to characterize normal observations, and we introduce the general formulation that describes anomalous behaviors.

6.1.1 From time series to time instances

Data collected and recorded continuously over an interval of time are known as time series. Formally, it can be defined as a set S of pairs each comprising a vector of samples of m simultaneously monitored quantities $\boldsymbol{s}_i = (s^{(0)}, \ldots, s^{(m-1)})$ acquired at a time instant, or timestamp t_i: $S = \{(\boldsymbol{s}_i, t_i) | i \in \mathbb{N}, t_k < t_l \text{ if } k < l\}$. When only one quantity is tracked, i.e., $m = 1$, the time series S is called univariate, while if $m > 1$, S is a multivariate time series. Since the time interval between two

consecutive measurements usually remains constant, the information on time can be dropped. In this work, we focus on univariate series only, which ultimately can be seen as a vector $\boldsymbol{s}$ of ordered measurements $\boldsymbol{s} = (s_0, \dots, s_k, \dots)$. From now on we treat $\boldsymbol{s}$ as a signal sampled with a time period T.

In a realistic scenario where the signal $\boldsymbol{s}$ must be processed in a real-time fashion, it is impractical to treat $\boldsymbol{s}$ as a single entity. We model $\boldsymbol{s}$ as a concatenation of not overlapping signal instances each made of n consecutive samples: $\boldsymbol{s} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_k \dots)$, with $\boldsymbol{x}_j = (s_{jn}, \dots, s_{j(n+1)-1})$. We also suppose that the length of the instance n is large enough so that $\boldsymbol{x}^j$ captures the main statistical features of the entire signal $\boldsymbol{s}$, and that the dependency between $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ with $k \neq j$ is negligible. This allows us to focus on one $\boldsymbol{x} = (x_0, \dots x_{n-1})$ instance at a time.

6.1.2 Model of normal signals

Given these assumptions, the acquired signal samples are collected in a vector $\boldsymbol{x} \in \mathbb{R}^n$, which can represent either a source of normal readings $\boldsymbol{x}^{\text{ok}} \in \mathbb{R}^n$ or a source of anomalous readings $\boldsymbol{x}^{\text{ko}} \in \mathbb{R}^n$, as described in Section 5.1. The two sources are assumed to have different statistics which should help telling anomalous instances from normal ones. With no loss of generality, we assume $\boldsymbol{x}^{\text{ok}}$ to have zero mean $\mathbf{E}\left[\boldsymbol{x}^{\text{ok}}\right] = 0$ and covariance matrix $\boldsymbol{\Sigma}^{\text{ok}} = \mathbf{E}\left[\boldsymbol{x}^{\text{ok}}\boldsymbol{x}^{\text{ok}\top}\right]$. We also assume $\boldsymbol{x}^{\text{ok}}$ normalized to unit power so that $\frac{1}{n}\mathbf{E}\left[\|\boldsymbol{x}^{\text{ok}}\|^2\right] = 1$, where $\|\cdot\|$ indicates the l_2 norm.

In real-world signals, power is typically unevenly distributed in the signal space. This can be observed in the spectral decomposition $\boldsymbol{\Sigma}^{\text{ok}} = \boldsymbol{U}^{\text{ok}}\boldsymbol{\Lambda}^{\text{ok}}\boldsymbol{U}^{\text{ok}\top}$ with $\boldsymbol{U}^{\text{ok}}$ being the $n \times n$ orthonormal matrix containing the eigenvectors of $\boldsymbol{\Sigma}^{\text{ok}}$ and $\boldsymbol{\Lambda}^{\text{ok}}$ being the $n \times n$ diagonal matrix listing the corresponding eigenvalues $\lambda_0^{\text{ok}} \geq \lambda_1^{\text{ok}} \geq \dots \geq \lambda_{n-1}^{\text{ok}} \geq 0$. Such eigenvalues are not equal and, given their decreasing order, one may find an $\bar{n}$ such that the fraction of power contained in the first $\bar{n}$ components is

$$\gamma = \frac{1}{n}\sum_{j=0}^{\bar{n}-1}\lambda_j^{\text{ok}} \gg \frac{\bar{n}}{n} \tag{6.1}$$

and thus identifies the so-called $\bar{n}$ principal components of the signal. Signal components beyond the principal ones are usually dominated by noise and carry little information on the features of the normal signal.

6.1.3 Model of anomalous signals

As described in Section 5.1.1 anomalies are often described according either to their cause or to their effect on the system in each specific application. Examples of both sensor faults and system anomalies observed in different monitoring scenarios have

already been presented in Table 5.1. Though such domain knowledge is always valuable, its generalization into an assessment framework requires abstracting from causes and from general effects to focus on the changes that the anomalies cause on the acquired signals.

To this end, we propose a dictionary of anomalies derived from a unified mathematical model, specialized to capture the signal features that are typically affected by real-world anomalies. We focus on anomalies that are *variations* of the normal signal in the sense that anomalous waveforms can be written starting from normal waveforms as

$$\boldsymbol{x}^{\mathrm{ko}} = c\left(\boldsymbol{x}^{\mathrm{ok}}\right) + \boldsymbol{d} \tag{6.2}$$

where $c : \mathbb{R}^n \mapsto \mathbb{R}^n$ is a potentially nonlinear function modeling how the anomaly changes the normal signal for which we assume $\mathbf{E}\left[c\left(\boldsymbol{x}^{\mathrm{ok}}\right)\right] = 0$, and $\boldsymbol{d} \in \mathbb{R}^n$ contains the samples of a disturbance waveform independent of $\boldsymbol{x}^{\mathrm{ok}}$ for which we assume power $\frac{1}{n}\mathbf{E}\left[\|\boldsymbol{d}\|^2\right] = a^2$ for a given parameter a. Independence between $\boldsymbol{d}$ and $\boldsymbol{x}^{\mathrm{ok}}$ implies $\mathbf{E}\left[{\boldsymbol{x}^{\mathrm{ok}}}^{\top}\boldsymbol{d}\right] = \mathbf{E}\left[\boldsymbol{d}^{\top}c\left(\boldsymbol{x}^{\mathrm{ok}}\right)\right] = 0$ so that the two components in (6.2) give separate power contributions. Depending on the anomaly, c may be a simple scaling, a full linear mapping identified by an $n \times n$ matrix, or a non-linearity.

To effectively span the anomaly space, we apply variations to the normal signal separately, i.e., if $a^2 > 0$ then c amounts only to signal scaling, while if $a^2 = 0$ we either redistribute signal power by means of a linear transformation, or corrupt signal samples by means of a nonlinear c.

To quantify how much the anomalous waveforms are expected to be different from normal ones, we make use of *deviation* [122]

$$\delta = \frac{1}{n}\mathbf{E}\left[\left\|\boldsymbol{x}^{\mathrm{ok}} - \boldsymbol{x}^{\mathrm{ko}}\right\|^2\right] = 1 + \frac{1}{n}\mathrm{tr}\left(\boldsymbol{\Sigma}^{\mathrm{ko}}\right) - \frac{2}{n}\mathbf{E}\left[{\boldsymbol{x}^{\mathrm{ok}}}^{\top}c\left(\boldsymbol{x}^{\mathrm{ok}}\right)\right] \tag{6.3}$$

where $\boldsymbol{\Sigma}^{\mathrm{ko}} = \mathbf{E}\left[\boldsymbol{x}^{\mathrm{ko}}{\boldsymbol{x}^{\mathrm{ko}}}^{\top}\right]$ is the correlation matrix of the anomalous signal and we have exploited that $\frac{1}{n}\mathrm{tr}\left(\boldsymbol{\Sigma}^{\mathrm{ok}}\right) = 1$, as well as the uncorrelatedness of $\boldsymbol{x}^{\mathrm{ok}}$ and $\boldsymbol{d}$.

Deviation are used to parameterize the difficulty of the AD task, so that detectors can be tested against different anomalies that, in principle, offer the same level of challenge.

In case $a^2 = 0$ and $c\left(\boldsymbol{x}^{\mathrm{ok}}\right) = \boldsymbol{C}\boldsymbol{x}^{\mathrm{ok}}$ for some matrix $\boldsymbol{C}$ we may set

$$\boldsymbol{C} = \boldsymbol{U}^{\mathrm{ko}}\sqrt{\boldsymbol{\Lambda}^{\mathrm{ko}}}\sqrt{\boldsymbol{\Lambda}^{\mathrm{ok}}}^{-1}{\boldsymbol{U}^{\mathrm{ok}}}^{\top} \tag{6.4}$$

that leverages the spectral decomposition also of the anomalous signals $\boldsymbol{\Sigma}^{\mathrm{ko}} = \boldsymbol{U}^{\mathrm{ko}}\boldsymbol{\Lambda}^{\mathrm{ko}}{\boldsymbol{U}^{\mathrm{ko}}}^{\top}$ and on the straightforward definition of the square root of a diagonal

matrix. The linearity of the mapping between normal and anomalous signals implies that the last expectation in (6.3) becomes

$$\mathbf{E}\left[\boldsymbol{x}^{\mathrm{ok}\top} c\left(\boldsymbol{x}^{\mathrm{ok}}\right)\right] = \operatorname{tr}\left(\boldsymbol{C}\boldsymbol{\Sigma}^{\mathrm{ok}}\right) = \operatorname{tr}\left(\boldsymbol{U}^{\mathrm{ko}}\sqrt{\boldsymbol{\Lambda}^{\mathrm{ko}}}\sqrt{\boldsymbol{\Lambda}^{\mathrm{ok}}}\boldsymbol{U}^{\mathrm{ok}\top}\right) \tag{6.5}$$

6.2 Framework for detector selection

In this section, we describe the proposed framework for detector selection, illustrated in Fig. 6.1. After the target detectors are trained on normal data, the designer can use a set of normal test samples to generate their corresponding anomalous counterparts. These anomalous signals are produced according to the selected anomaly type, chosen from a predefined suite, and by setting the desired anomaly intensity through a deviation parameter. Subsequently, the detectors are evaluated on both normal and anomalous datasets to assess their ability to discriminate between the two conditions.

To describe the procedure in more detail, we: *i*) introduce the anomaly suite, categorized according to the effect of each anomaly on the signal power; *ii*) describe the implementation of the anomalies; *iii*) provide the mathematical expressions defining their deviation levels; and *iv*) define the metric used for detector evaluation.

6.2.1 Anomalies that increase signal power

If we assume that c is the identity, $a^2 > 0$ and the anomalous signal has power $\frac{1}{n}\mathbf{E}\left[\|\boldsymbol{x}^{\mathrm{ko}}\|^2\right] = 1 + a^2$. We consider several possible disturbances.

Constant

The disturbance is constant, with $d_j = \pm a$ for $j = 0, \ldots, n-1$.

Step

We define

$$d_j = \pm a \begin{cases} r\sqrt{n/\bar{\jmath}} & j = 0, \ldots, \bar{\jmath}-1 \\ (1-r)\sqrt{n/(n-\bar{\jmath})} & j = \bar{\jmath}, \ldots, n-1 \end{cases} \tag{6.6}$$

where $r \in \{-1,1\}$ determines whether we have a rising or a falling step and $\bar{\jmath} \in \{0, \ldots, n-1\}$ sets the step position within the time window.

Impulse

We define

$$d_j = \pm a \begin{cases} \sqrt{n} & \text{if } j = \bar{j} \\ 0 & \text{otherwise} \end{cases}, \quad 0 \leq j < n \tag{6.7}$$

where $\bar{j} \in \{0, \ldots, n-1\}$ sets the pulse position.

Gaussian white noise (GWN)

We define $\boldsymbol{d}$ as a simple White Gaussian noise.

Gaussian narrowband noise (GNN)

The disturbance is a Gaussian signal with band $[f_0 - B/2, f_0 + B/2]$ with $0 \leq f_0 \leq 1/2$ as center frequency and $0 \leq B \leq \min\{f_0, 1/2 - f_0\}$ as bandwidth.

6.2.2 Anomalies that do not change signal power

In this case, anomalies alter the normal signal by redistributing its power in the signal space. Redistribution can be between the normal signal and an independent disturbance or between the components of the normal signal itself.

Mixing with a disturbance

When a disturbance with power $a^2 \leq 1$ is present, we model mixing by setting $c\left(\boldsymbol{x}^{\text{ok}}\right) = \sqrt{1-a^2}\boldsymbol{x}^{\text{ok}}$. As a disturbance, we consider only constant $\boldsymbol{d}$ and white noise $\boldsymbol{d}$.

Intra-signal mixing

When power redistribution affects only the components of the normal signal, we assume this happens by linear mapping, i.e., by some matrix $\boldsymbol{C}$ yielding power invariance, i.e., causing $\frac{1}{n}\text{tr}\left(\boldsymbol{\Sigma}^{\text{ko}}\right) = 1$.

Time warping Time warping [123] alters the normal signal by local decelerations. It can be modeled assuming that the entries in $\boldsymbol{x}^{\text{ok}}$ are samples at rate f_s taken from a continuous-time waveform $\boldsymbol{x}^{\text{ok}}(t)$ subject to a time-to-time mapping $w(t)$ yielding an anomalous waveform $\boldsymbol{x}^{\text{ko}}(t) = \boldsymbol{x}^{\text{ok}}(w(t))$ that is sampled into the vector $\boldsymbol{x}^{\text{ko}}$. With this, defining the sampling instants $t_j = \frac{j}{f_\text{s}}$ for $j = 0, \ldots, n-1$, we have

$$\boldsymbol{x}_j^{\text{ko}} = \boldsymbol{x}^{\text{ko}}(t_j) = \boldsymbol{x}^{\text{ok}}(w(t_j)) = \sum_{k=0}^{n-1} \boldsymbol{x}_k^{\text{ok}} \eta(w(t_j) f_\text{s} - k) \mathrm{d}w_k \tag{6.8}$$

where $\eta(\cdot)$ is the interpolating function allowing waveform reconstruction from its samples and $\mathrm{d}w_j$ accounts for local time warping to guarantee power invariance. Clearly in this case $\boldsymbol{C}_{j,k} = \eta(w(t_j)f_\mathrm{s} - k)\mathrm{d}w_k$.

Further to power redistribution along time, we have to consider power redistribution in spectrum, i.e., from the point of view of the spectral decomposition of the correlation matrix $\boldsymbol{\Sigma}^{\mathrm{ok}} = \boldsymbol{U}^{\mathrm{ok}}\boldsymbol{\Lambda}^{\mathrm{ok}}\boldsymbol{U}^{\mathrm{ok}\top}$, in which the first $\bar{n}$ components are the principal ones and characterize normality. We may think of two alterations.

Spectral alteration This anomaly affects how power is distributed among the principal components. It can be modelled by altering the eigenvalues on the diagonal of $\boldsymbol{\Lambda}^{\mathrm{ok}}$ obtaining $\boldsymbol{\Lambda}^{\mathrm{ko}}$ and thus $\boldsymbol{C}$ from (6.4).

Principal subspace alteration This anomaly changes the principal components. It can be modelled by altering the columns of $\boldsymbol{U}^{\mathrm{ok}}$ to obtain a new orthonormal $\boldsymbol{U}^{\mathrm{ko}}$ and thus $\boldsymbol{C}$ from (6.4).

6.2.3 Anomalies that decrease signal power

Saturation

Saturation is modeled by clipping the largest samples in a window to a maximum value x_{SAT} as in

$$x_j^{\mathrm{ko}} = \begin{cases} x_{\mathrm{SAT}}\mathrm{sign}\left(x_j^{\mathrm{ok}}\right) & \text{if } \left|x_j^{\mathrm{ok}}\right| > x_{\mathrm{SAT}} \\ x_j^{\mathrm{ok}} & \text{otherwise} \end{cases} \tag{6.9}$$

Dead-zone

In contrast to saturation, the dead-zone anomaly sets to 0 the samples smaller than x_{DZ} as in

$$x_j^{\mathrm{ko}} = \begin{cases} 0 & \text{if } \left|x_j^{\mathrm{ok}}\right| < x_{\mathrm{DZ}} \\ x_j^{\mathrm{ok}} & \text{otherwise} \end{cases} \tag{6.10}$$

6.2.4 Anomaly implementation

Considering the taxonomy of anomalies presented above, we provide a definition of how the random components of the mathematical models are mapped into actual instances.

- The symbol $\pm$ is implemented either with a $+$ or $-$ with $\frac{1}{2}$ probability.
- The rising parameter r is drawn uniformly from $\{0,1\}$, and the position $\bar{j}$ is set to $\lceil n/2 \rceil$.

- When $\boldsymbol{d}$ is a Gaussian noise in the band $[f_0 - {}^{B}/_{2}, f_0 + {}^{B}/_{2}]$ we draw it as a Gaussian random vector with zero means and covariance $\boldsymbol{\Sigma}$ with [124]

$$\boldsymbol{\Sigma}_{j,k} = a^2 \cos\left(2\pi(j-k)f_0\right) \operatorname{sinc}\left((j-k)B\right) \tag{6.11}$$

  for $j, k = 0, \ldots, n-1$. When white noise is needed we simply set $\boldsymbol{\Sigma} = a^2 \boldsymbol{I}$.

- Interpolation η uses 3-rd order cardinal splines [125].

- The warping function is $w(t) = (1-\alpha)\, t$ with $\alpha \in [0, 1]$ controlling local deceleration.

- We concentrate spectral alterations on the principal components and assume that the noise subspace is untouched, i.e., that $\lambda_j^{\text{ko}} \neq \lambda_j^{\text{ok}}$ for $j = 0, \ldots, \bar{n} - 1$ but $\lambda_j^{\text{ko}} = \lambda_j^{\text{ok}}$ for $j = \bar{n}, \ldots, n-1$. The number of principal components $\bar{n}$ is computed according to the criterion defined in [126].

  The first $\bar{n}$ eigenvalues of $\boldsymbol{\Lambda}^{\text{ok}}$ are altered by defining $\boldsymbol{\ell}^{\text{ko}} = \frac{1}{\sqrt{n\gamma}} \left(\sqrt{\lambda_0^{\text{ko}}}, \ldots, \sqrt{\lambda_{\bar{n}-1}^{\text{ko}}}\right)^{\top}$ and $\boldsymbol{\ell}^{\text{ok}} = \frac{1}{\sqrt{n\gamma}} \left(\sqrt{\lambda_0^{\text{ok}}}, \ldots, \sqrt{\lambda_{\bar{n}-1}^{\text{ok}}}\right)^{\top}$. Since power invariance requires $\left\|\boldsymbol{\ell}^{\text{ok}}\right\| = \left\|\boldsymbol{\ell}^{\text{ko}}\right\| = 1$, we set $\boldsymbol{\ell}^{\text{ko}} = \boldsymbol{R}_{\bar{n},\theta} \boldsymbol{\ell}^{\text{ok}}$ for some random $\bar{n} \times \bar{n}$ matrix $\boldsymbol{R}_{\bar{n},\theta}$ encoding a rotation of an angle θ. The altered eigenvalues are then retrieved from $\boldsymbol{\ell}^{\text{ko}}$ by squaring and scaling.

- Principal subspace alteration sets $\boldsymbol{U}^{\text{ko}} = \boldsymbol{R}_{n,\theta} \boldsymbol{U}^{\text{ok}}$, where $\boldsymbol{R}_{n,\theta}$ is an $n \times n$ random matrix encoding a rotation of an angle θ applied to the columns of $\boldsymbol{U}^{\text{ok}}$.

- Random rotations are built assuming that n and $\bar{n}$ are even, with

$$\boldsymbol{R}_{\theta} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{r}_{\theta} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \boldsymbol{r}_{\theta} \end{pmatrix} \boldsymbol{Q}^{\top}, \quad \boldsymbol{r}_{\theta} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{6.12}$$

  where $\boldsymbol{r}_{\theta}$ is the 2D rotation matrix and $\boldsymbol{Q}$ is generated by orthonormalizing an instance of the Ginibre ensemble [127].

- In saturation and dead-zone alterations the thresholds are adapted in each window to match the desired δ in (6.18) and (6.19).

- In principle, different effects can be combined to create an anomaly that emulates real-world anomalies manifesting with multiple superimposed effects. More details can be found in Appendix A.

6.2.5 Deviations

For each type of anomaly, Tab 6.1 reports the relationship between the parameters in the anomaly definition and the deviation δ. Those formulas allow to apply alterations posing a prescribed and uniform level of challenge to the detector to test.

Table 6.1: Expressions of deviation δ for each anomaly

Anomalies	δ	
Increasing	a^2	(6.13)
Mixing	$2\left[1-\sqrt{1-a^2}\right]$	(6.14)
Time warping	$2\left[1-\frac{1}{n}\mathrm{tr}\left(\boldsymbol{C}\boldsymbol{\Sigma}^{\mathrm{ok}}\right)\right]$	(6.15)
Spectral alt.	$2\gamma\left[1-\cos(\theta)\right]$	(6.16)
Principal subspace alt.	$2\left[1-\cos(\theta)\right]$	(6.17)
Saturation	$\sum_{\left\lvert x_j^{\mathrm{ok}}\right\rvert > x_{\mathrm{SAT}}} \left[x_j^{\mathrm{ok}} - x_{\mathrm{SAT}}\mathrm{sign}\left(x_j^{\mathrm{ok}}\right)\right]^2$	(6.18)
Dead-zone	$\sum_{\left\lvert x_j^{\mathrm{ok}}\right\rvert \leq x_{\mathrm{DZ}}} \left(x_j^{\mathrm{ok}}\right)^2$	(6.19)

In (6.15) the deviation is controlled by the parameter α inherent to $\boldsymbol{C}$, while in (6.16) deviation is caused by alterations $\boldsymbol{R}_{\bar{n},\theta}$. Considering that, principal subspace alterations caused by $\boldsymbol{R}_{n,\theta}$ are equivalent to rotating each $\boldsymbol{x}^{\mathrm{ok}}$ into the corresponding $\boldsymbol{x}^{\mathrm{ko}}$ by the same angle, (6.17) can be obtained by exploiting (6.4) and (6.16). Finally, (6.18) and (6.19) can be used to set respectively x_{SAT} and x_{DZ} so that the average deviation over the available dataset is the given δ.

6.2.6 Metrics

As anticipated in Section 5.1 when the detector is fed with an instance $\boldsymbol{x}$ that is either ok or ko, the output of a detector can be considered as a function $z(\boldsymbol{x})$ that assigns a score to the instance, such that larger scores correspond to more anomalous behaviours [83].

To obtain the final binary decision, the detection score is compared against a threshold determined by the specific requirements of the application. To ensure that the

evaluation remains independent of the threshold choice, we adopt the Area Under the ROC Curve (AUC) metric [128], as defined in (5.7). When one anomalous and one normal vectors are randomly picked, AUC $\in$ [0,1] is the probability that the score will be higher for the former than for the latter. What we use as a metric is a variation of AUC corresponding to the probability of correct detection

$$P_D = \begin{cases} \text{AUC} & \text{if AUC} \geq 0.5 \\ 1 - \text{AUC} & \text{if AUC} < 0.5 \end{cases} \tag{6.20}$$

P_D accounts for the fact that, when a detector consistently scores normal instances higher than anomalous ones, i.e., AUC < 0.5, it is still useful for detection, if its output is interpreted in a reversed manner.

6.3 Experimental setup

To prove the effectiveness of the proposed anomaly models, we evaluate numerically the performance of a set of detectors on two different datasets: Electrocardiogram (ECG) signals coming from a Health Monitoring application and accelerometers' (ACC) waveforms used for Structural Health Monitoring.

Table 6.2: Signal features and the corresponding score $z(x)$

Feature	$z(\boldsymbol{x})$	**Description**
pk-pk	$\max \boldsymbol{x} - \min \boldsymbol{x}$	Peak-to-peak value
energy	$\sum_{i=0}^{n-1} x_i^2$	Energy of the vector
TV	$\sum_{i=0}^{n-1} \lvert x_{i+1} - x_i \rvert$	Total variation [129]
ZC	$\frac{1}{2} \sum_{i=0}^{n-2} [1 - \text{sign}(x_i x_{i+1})]$	Number of zero-crossings

6.3.1 Detectors

Our approach is tested by deploying a variety of detectors based on different techniques that have been proposed in literature [84]. Specifically, we focus on the detectors described in Section 5.2, setting $k = \bar{n}$ and k corresponding to $\gamma = 0.999$, as defined in (6.1). DL-based methods are excluded from our analysis, since the aim is to validate the effectiveness of the proposed evaluation framework rather than to benchmark the performance of individual detectors.

In addition to the most common detectors, we also consider a set of representative signal features that can be efficiently computed. Each feature listed in Table 6.2 defines a distinct score, whose detection capability provides insight into the nature of the modeled anomalies.

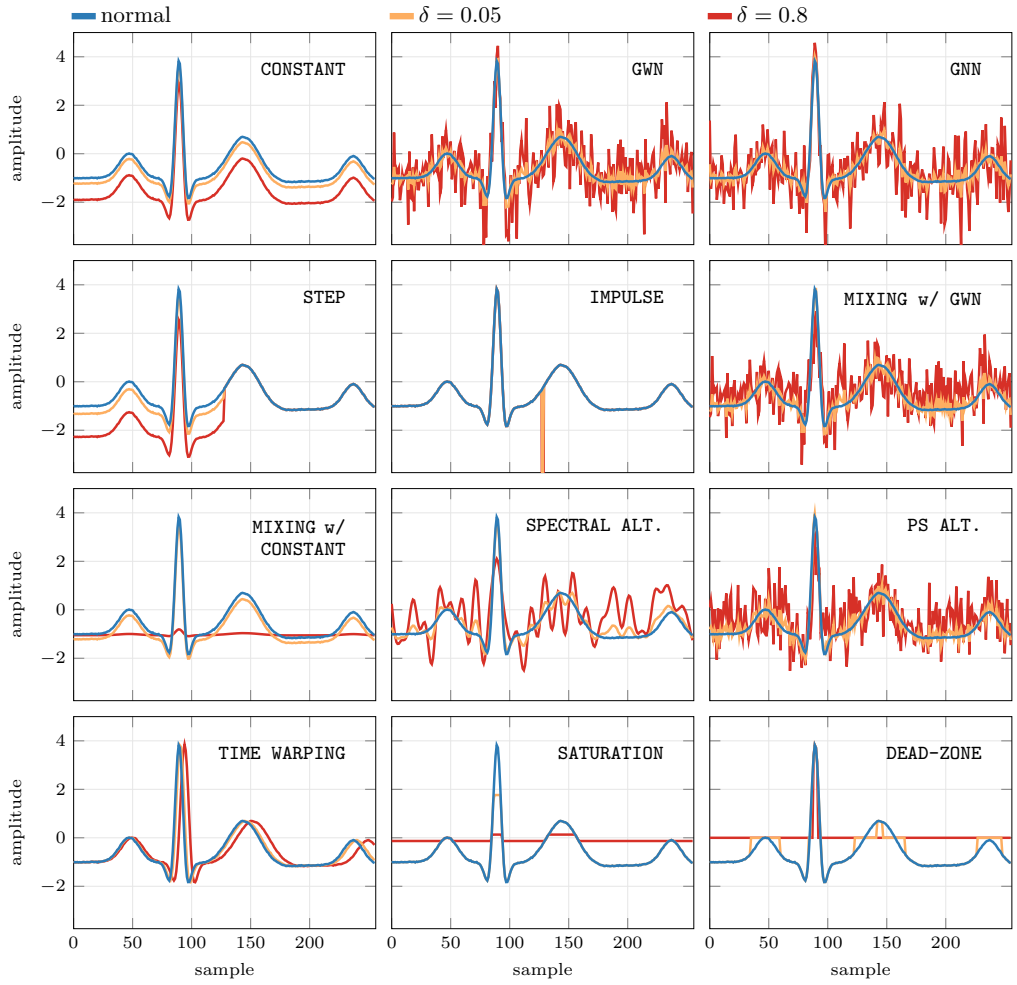

Figure 6.2: Examples of ECG anomalies for two different values of deviation $\delta = \{0.05, 0.8\}$.

6.3.2 Datasets

ECG signals

The reference ECG signals were generated using a realistic generator introduced in [130] with the setup described in [131]. In particular, the sampling rate is set to 256 sample/s and the heart-beat rate is uniformly drawn in the range 60-100 beat/min. We first generate 7.7×10^4 chunks of 2 s that are then randomly

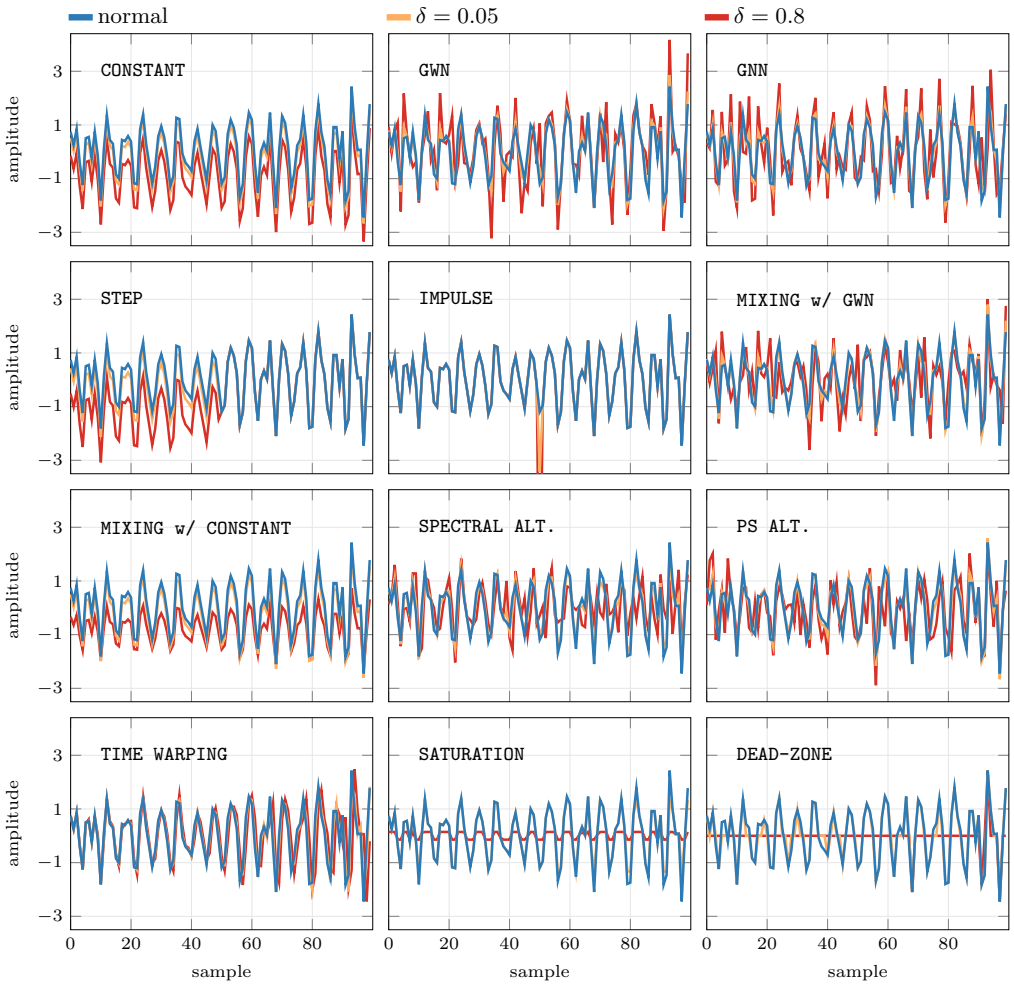

Figure 6.3: Examples of ACC anomalies for two different values of deviation $\delta = \{0.05, 0.8\}$.

split into non-overlapping windows of length $n = 256$ to create vectors $\boldsymbol{x}^{\text{ok}} \in \mathbb{R}^n$. Gaussian white noise is superimposed with a power guaranteeing an SNR of 40 dB. The resulting number of principal components is $\bar{n} = 52$. Finally, the dataset is split so that, 10^4 vectors are allocated to performance assessment during the testing phase and the remaining vectors are employed during the training phase.

ACC signals

Acceleration signals utilized in this work have been collected as a result of the Structural Health Monitoring of a viaduct located in Italy [102, 101]. The entire dataset comprises readings of 90 three-axis accelerometers acquired at a sampling rate of 100 sample/s and which represent the elastic response of the structure to

Table 6.3: Performance of different detectors (columns) in terms of P_D, working on ECG anomalies (rows) with a fixed deviation $\delta = 0.05$

		PCA-BASED				GD-BASED			ML-BASED			FEATURE-BASED			
	ANOMALY	SPE_{52}	SPE_{59}	T^2_{52}	T^2_{59}	AR_4	AR_{16}	MD	$OC_{poly,\ 0.01}$	LOF_5	IF_{250}	energy	TV	ZC	pk-pk
INCREASING	GWN	1.00	1.00	0.55	0.63	1.00	1.00	1.00	0.50	0.98	0.59	0.56	1.00	1.00	0.70
	IMPULSE	1.00	1.00	0.54	0.62	1.00	1.00	1.00	0.50	0.98	0.54	0.56	0.76	0.66	0.76
	STEP	0.54	0.60	0.65	0.63	0.96	0.99	0.99	0.50	0.89	0.58	0.56	0.51	0.53	0.59
	CONSTANT	0.50	0.50	0.52	0.51	0.50	0.51	0.51	0.50	0.63	0.56	0.55	0.50	0.54	0.50
	GNN	0.93	0.92	0.55	0.59	0.93	0.93	0.93	0.50	0.97	0.59	0.56	0.95	0.94	0.67
INVARIANT	MIXING w/ GWN	1.00	1.00	0.50	0.59	1.00	1.00	1.00	0.59	0.98	0.55	0.50	1.00	1.00	0.62
	MIXING w/ CONSTANT	0.52	0.51	0.53	0.53	0.56	0.56	0.57	0.59	0.62	0.52	0.50	0.54	0.54	0.60
	SPECTRAL ALT.	0.50	0.50	0.62	0.60	0.59	0.55	0.55	0.59	0.97	0.54	0.50	0.73	0.80	0.51
	PRINCIPAL SUBSPACE ALT.	1.00	1.00	0.50	0.58	1.00	1.00	1.00	0.59	0.98	0.55	0.50	1.00	1.00	0.61
	TIME WARPING	0.52	0.52	0.51	0.51	0.70	0.73	0.72	0.50	0.52	0.50	0.50	0.51	0.51	0.50
DEC.	SATURATION	0.86	0.99	0.79	0.81	1.00	1.00	1.00	0.83	0.81	0.61	0.70	0.69	0.50	0.97
	DEAD-ZONE	0.92	0.98	0.50	0.55	1.00	1.00	1.00	0.73	0.93	0.53	0.56	0.50	0.86	0.50

Table 6.4: Performance of different detectors (columns) in terms of P_D, working on ECG anomalies (rows) with a fixed deviation $\delta = 0.8$

		PCA-BASED				GD-BASED			ML-BASED			FEATURE-BASED			
	ANOMALY	SPE_{52}	SPE_{59}	T^2_{52}	T^2_{59}	AR_4	AR_{16}	MD	$OC_{poly,\ 0.01}$	LOF_5	IF_{250}	energy	TV	ZC	pk-pk
INCREASING	GWN	1.00	1.00	0.94	1.00	1.00	1.00	1.00	0.51	1.00	0.99	0.99	1.00	1.00	0.99
	IMPULSE	1.00	1.00	0.93	1.00	1.00	1.00	1.00	0.53	1.00	0.55	0.99	1.00	0.66	1.00
	STEP	0.90	0.98	1.00	1.00	1.00	1.00	1.00	0.52	1.00	0.98	0.96	0.56	0.82	0.81
	CONSTANT	0.50	0.50	0.75	0.71	0.55	0.66	0.59	0.51	0.99	0.96	0.94	0.50	0.89	0.50
	GNN	0.94	0.93	0.70	0.74	0.96	0.96	0.97	0.50	1.00	0.99	0.97	0.99	0.97	0.96
INVARIANT	MIXING w/ GWN	1.00	1.00	0.52	0.96	1.00	1.00	1.00	1.00	1.00	0.85	0.51	1.00	1.00	0.52
	MIXING w/ CONSTANT	0.79	0.77	0.91	0.91	1.00	1.00	1.00	1.00	0.98	0.77	0.50	0.94	0.97	1.00
	SPECTRAL ALT.	0.52	0.51	0.96	0.94	0.91	0.81	0.85	0.99	1.00	0.73	0.53	1.00	1.00	0.72
	PRINCIPAL SUBSPACE ALT.	1.00	1.00	0.50	0.93	1.00	1.00	1.00	1.00	1.00	0.80	0.50	1.00	1.00	0.50
	TIME WARPING	0.59	0.58	0.54	0.55	0.77	0.79	0.78	0.50	0.57	0.50	0.50	0.54	0.56	0.50
DEC.	SATURATION	0.86	0.58	1.00	1.00	0.72	0.87	0.87	1.00	1.00	1.00	1.00	1.00	0.50	1.00
	DEAD-ZONE	1.00	1.00	0.99	0.61	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.99

vehicles transit and environmental stimuli. Focusing only on a single axis of a particular sensor, we randomly split the readings into 3.77×10^5 windows of length $n = 100$ to create vectors $\boldsymbol{x}^{\text{ok}} \in \mathbb{R}^n$. The intrinsic noise of the dataset leads to $\bar{n} = 16$. Once again, 10^4 vectors have been allocated to the performance assessment, and the rest are dedicated to the training phase.

6.4 Results

For each signal class and anomaly type, anomalous instances are generated by corrupting the original input vectors as described in Section 6.2. Considering the two

Table 6.5: Performance of different detectors (columns) in terms of P_D, working on ACC anomalies (rows) with a fixed deviation $\delta = 0.05$

		PCA-BASED				GD-BASED			ML-BASED			FEATURE-BASED			
	ANOMALY	SPE_{16}	SPE_{85}	T^2_{16}	T^2_{85}	AR_4	AR_{16}	MD	$OC_{RBF,\ 0.01}$	LOF_5	IF_{500}	energy	TV	ZC	pk-pk
INCREASING	GWN	0.80	0.98	0.55	0.87	0.76	0.93	1.00	0.64	0.85	0.59	0.58	0.59	0.57	0.60
	IMPULSE	0.80	0.99	0.55	0.88	0.76	0.94	1.00	0.64	0.86	0.55	0.58	0.54	0.51	0.73
	STEP	0.82	0.97	0.50	0.91	0.85	0.90	0.93	0.64	0.86	0.59	0.58	0.50	0.70	0.54
	CONSTANT	0.81	0.51	0.50	0.84	0.85	0.90	0.83	0.64	0.84	0.59	0.58	0.50	0.72	0.50
	GNN	0.78	0.80	0.54	0.83	0.71	0.85	0.88	0.63	0.84	0.58	0.58	0.58	0.55	0.59
INVARIANT	MIXING w/ GWN	0.79	0.98	0.54	0.87	0.75	0.93	1.00	0.63	0.85	0.58	0.58	0.58	0.57	0.60
	MIXING w/ CONSTANT	0.81	0.51	0.51	0.84	0.85	0.90	0.83	0.64	0.84	0.58	0.58	0.51	0.72	0.51
	SPECTRAL ALT.	0.50	0.50	0.51	0.51	0.50	0.51	0.51	0.50	0.53	0.50	0.50	0.50	0.52	0.50
	PRINCIPAL SUBSPACE ALT.	0.58	0.83	0.50	0.63	0.55	0.72	0.94	0.51	0.60	0.50	0.50	0.50	0.51	0.51
	TIME WARPING	0.50	0.50	0.50	0.50	0.50	0.51	0.51	0.50	0.51	0.50	0.50	0.50	0.51	0.50
DEC.	SATURATION	0.54	0.69	0.56	0.53	0.55	0.59	0.90	0.56	0.54	0.55	0.55	0.55	0.50	0.68
	DEAD-ZONE	0.58	0.81	0.50	0.62	0.55	0.72	0.93	0.50	0.61	0.51	0.51	0.51	0.82	0.50

Table 6.6: Performance of different detectors (columns) in terms of P_D, working on ACC anomalies (rows) with a fixed deviation $\delta = 0.8$

		PCA-BASED				GD-BASED			ML-BASED			FEATURE-BASED			
	ANOMALY	SPE_{16}	SPE_{85}	T^2_{16}	T^2_{85}	AR_4	AR_{16}	MD	$OC_{RBF,\ 0.01}$	LOF_5	IF_{500}	energy	TV	ZC	pk-pk
INCREASING	GWN	0.96	1.00	0.76	0.99	0.97	1.00	1.00	0.89	0.99	0.81	0.80	0.82	0.67	0.83
	IMPULSE	0.96	1.00	0.77	1.00	0.98	1.00	1.00	0.89	0.99	0.58	0.80	0.64	0.51	0.95
	STEP	0.97	1.00	0.52	1.00	1.00	1.00	1.00	0.90	0.99	0.82	0.80	0.51	0.92	0.69
	CONSTANT	0.97	0.69	0.51	0.98	1.00	1.00	0.98	0.90	0.99	0.83	0.80	0.50	0.93	0.50
	GNN	0.96	0.89	0.67	0.97	0.93	0.97	0.98	0.88	0.98	0.80	0.79	0.77	0.57	0.81
INVARIANT	MIXING w/ GWN	0.96	1.00	0.70	0.99	0.96	1.00	1.00	0.86	0.99	0.77	0.76	0.77	0.69	0.79
	MIXING w/ CONSTANT	0.96	0.58	0.63	0.97	1.00	0.99	0.97	0.87	1.00	0.79	0.76	0.64	0.96	0.65
	SPECTRAL ALT.	0.56	0.51	0.63	0.57	0.55	0.58	0.57	0.56	0.74	0.52	0.51	0.53	0.66	0.53
	PRINCIPAL SUBSPACE ALT.	0.78	0.98	0.54	0.87	0.73	0.93	0.99	0.59	0.93	0.51	0.50	0.52	0.65	0.53
	TIME WARPING	0.61	0.50	0.52	0.59	0.51	0.50	0.57	0.52	0.67	0.50	0.50	0.51	0.54	0.50
DEC.	SATURATION	0.89	0.69	0.94	0.88	0.90	0.74	0.65	0.93	0.83	0.92	0.92	0.94	0.50	0.98
	DEAD-ZONE	0.57	0.91	0.72	0.66	0.50	0.81	0.98	0.63	0.77	0.78	0.71	0.87	1.00	0.51

use cases, Fig. 6.2 and Fig. 6.3 illustrate the effect of each anomaly type on both a normal ECG and a normal ACC instance. In these figures, each subfigure corresponds to a different anomaly type and displays both the original and the altered signals instancev for two levels of deviation δ. As expected, a higher deviation leads to a more pronounced anomaly.

Original and corrupted vectors have been used to test several detectors with two different levels of deviation. Results expressed in terms of P_D for each defined anomaly are in Tables 6.3, 6.4, 6.5, and 6.6 where each column corresponds to a different detector. Achieved performance may identify the most effective detector for each type of anomaly, thus validating the tool under scrutiny at design time.

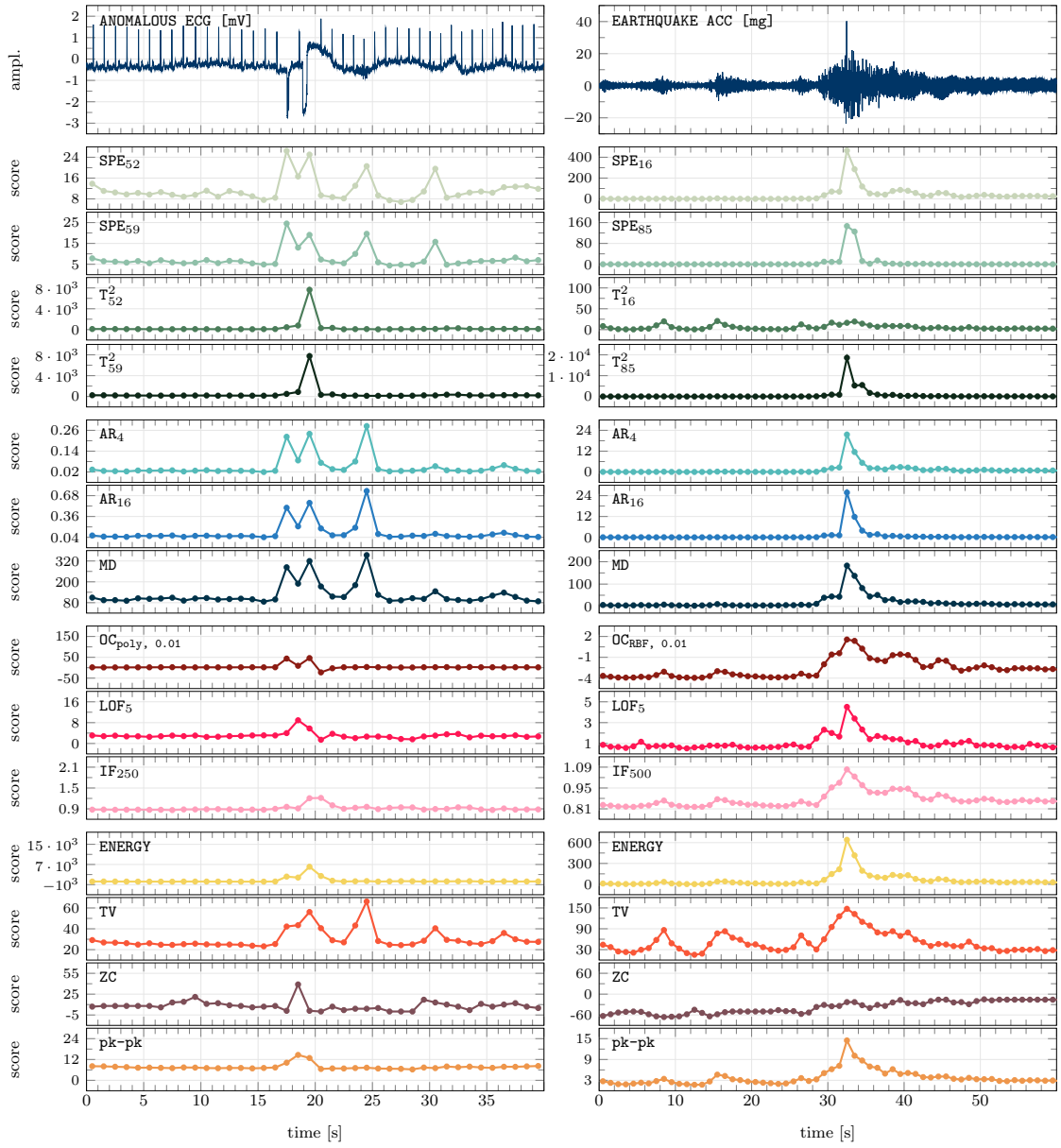

Figure 6.4: Scores trends (bottom) for four detectors working on anomalous real ECG time-series (right) and for four detectors applied on an ACC time series (left) containing an earthquake event.

In detail, Tables 6.3 and 6.4 provide an overview of the detectability in case of ECG signals with a deviation level of $\delta = 0.05$ for Table 6.3 and $\delta = 0.8$ for Table 6.4. From these results, it is evident how each type of anomaly has a different effect on the normal signal, leading to different detection performances. Additionally, some anomalies such as constant, and time warping, are difficult to detect using any of the considered detectors.

Moreover, comparing the detection performance in Table 6.3 and 6.4, as expected, increasing δ leads to better detection performance. This confirms the validity of δ as a parameter for generating anomalies with varying levels of severity[1].

The detection results for ACC signals are summarized in Table 6.5 and Table 6.6 for $\delta = 0.05$ and $\delta = 0.8$, respectively. As anticipated, the detection performances for the ACC signals are significantly different from those for ECG signals. For the same level of δ, P_D values are generally lower compared to the ECG case. Moreover, spectral alterations, time warping, and dead-zone are found to be the most challenging anomalies to detect in the ACC case. The results also indicate that feature-based detectors have inferior detection capabilities compared to more complex solutions.

To clarify how the framework evaluates detector performance within a specific signal class, it is important to note that the results in Table 6.3, Table 6.4, Table 6.5, and Table 6.6 should be interpreted column-wise. Each column summarizes the expected detection performance of the corresponding detector.

Fig. 6.4 further demonstrates the capability of our tool to predict the performance of detectors working on real-world anomalies. On the left-hand side, we present the scores of different detectors, which were trained on a reference synthetic ECG signal. These scores change based on the variations of a real-world ECG signal with anomalies [132] plotted at the top. While all detectors can identify the anomaly between 16 and 22 s, some, e.g., T^2_{52} and the energy-based detector, fail in detecting a noise increase between 23–26 s that does not affect the signal's power. This result, even when testing on a real ECG signal, aligns with the insights provided in Table 6.3, as it is able to anticipate the performance of 12 out of 14 considered detectors. In fact, with the exception of LOF_5 and ZC, the detectors failing with this real anomaly are the ones marked with a red cell in correspondence to the row MIXING w/ GWN in Table 6.3.

In addition, on the right side of Figure 6.4, the tool's ability to predict detector performance in identifying an earthquake occurring in the time interval of 28–38 s is demonstrated. Since an earthquake typically amplifies the energy of all signal components, it can be considered a GWN anomaly. Table 6.6 shows that SPE_{85}, AR_{16}, and MD are the most suitable for detecting this type of anomaly, while T^2_{16} and ZC perform poorly. This is consistent with the score trends shown in the figure, where SPE_{85}, AR_{16}, and MD exhibit sharp variations during the earthquake, while T^2_{16} and ZC show little deviation compared to their behavior during normal operation.

[1]This is detailed in Appendix A, where P_D for each detector/anomaly is plotted against δ.

Interestingly, in both cases, the most effective detectors are not machine learning-based, but rather simpler and less computationally demanding solutions.

6.5 Conclusion

The evaluation of anomaly detector performance is often hindered by the scarcity of anomalous data. In this work, we have presented a framework that enables the systematic assessment of anomaly detectors operating on time series. The proposed tool relies on a dictionary of anomalies modeled as alterations to the normal signal, reflecting non-nominal behaviors in the monitored system or acquisition chain, such as aging, wear, abrupt condition changes, hardware faults, or external disturbances. Each anomaly model is parameterized by a deviation factor that controls its departure from normality, thereby regulating the level of detection difficulty.

The effectiveness of the proposed framework has been demonstrated in two real-world scenarios: human health monitoring and structural health monitoring, using electrocardiogram (ECG) and acceleration (ACC) signals, respectively. The framework allows for the evaluation of multiple candidate detectors, highlighting the types of anomalies for which each performs best or worst. Empirical evidence confirms that the framework can anticipate detector performance on real-world anomalies, such as artifacts in ECG recordings and an earthquake in the structural monitoring of a bridge.

Conclusion

This dissertation presented a systematic investigation of data-driven methodologies for Prognostics and Health Management (PHM), focusing on two fundamental aspects of the discipline: Remaining Useful Life (RUL) estimation and Anomaly Detection (AD). The main objective was to advance the development of frameworks capable of addressing both diagnostics and prognostics, emphasizing real-world applicability.

The first part of the thesis focused on the prognostic task through the construction of a unified pipeline for RUL estimation. The study began with the assessment of different Health Indicators (HIs) to identify the most informative degradation representations and progressively evolved toward the comparison of multiple RUL estimation models in terms of both prognostic accuracy and computational efficiency. Within this progression, Autoencoder (AE)-based architectures became the core components for feature extraction and representation learning of system degradation patterns. The first preliminary study, conducted on satellite telemetry data, established the methodological foundation that was later extended to enable AEs to infer HIs directly from raw sensor measurements. Building upon these learned features, similarity and degradation-based RUL estimation approaches were developed, validated on the C-MAPSS dataset, and deployed on edge. This enabled a systematic analysis of the trade-offs between predictive capability and computational constraints inherent to resource-limited environments. Unlike much of the existing literature, where RUL estimation remains limited to model development and simulation studies, this work emphasizes practical deployability, bridging the gap between algorithmic research and real-world implementation on edge systems.

The second part of the dissertation proposed a general framework for the systematic evaluation of anomaly detection algorithms on temporal signals. The framework formalized a set of anomaly classes and synthetic generation processes to objectively assess detector performance under controlled conditions. Through comparative experiments across multiple algorithmic families, it was shown that this methodology can serve as a valuable instrument for benchmarking and selecting detection strategies in PHM applications.

Future Work

Future work will extend this research in several directions. A primary objective is the integration of the anomaly detection framework with RUL estimation, enabling detectors to identify deviations from nominal behavior to trigger or support the prognostic process. Achieving this goal will require extending the current AD framework to multivariate time series and overlapping windows, thus supporting its integration into the multi-class RUL prediction framework. Another important direction concerns the limitations associated with the use of simulated prognostic datasets, such as C-MAPSS, which are commonly adopted for RUL estimation. While these datasets provide a valuable benchmark for the development and comparison of prognostic algorithms, they rely on idealized degradation trajectories and simplified operational assumptions. A necessary step towards real industrial deployment therefore consists in transitioning from simulation-based and synthetic datasets to real-world prognostic data, where degradation behavior reflects realistic operating and environmental variability. From a modeling perspective, future research will focus on extending the current methodologies beyond purely data-driven inference. In particular, hybrid architectures that integrate physical knowledge will be explored in domains such as the prognostics of battery systems and bearing components, where the underlying degradation mechanisms are well understood and can be effectively modeled. This integration is expected to enhance the robustness and reliability of prognostic models, providing more physically consistent representations of degradation processes. On the implementation side, forthcoming developments will aim to consolidate the portability of the proposed frameworks through deployment on FPGA and other edge-oriented platforms. Such implementations will enable a systematic exploration of the trade-offs between predictive accuracy, computational efficiency, and energy consumption, which are central to the design of embedded PHM solutions.

In summary, this work contributed to the development of scalable PHM methodologies by proposing experimentally validated frameworks for both RUL estimation and anomaly detection. These methods provide a consistent foundation for the design of intelligent maintenance systems that effectively combine diagnostic awareness with reliable prognostic capability in industrial contexts.

Appendix A

A Framework for the Assessment of Time-Series Anomaly Detectors

This appendix presents a collection of additional figures that complement the main results, further demonstrating the effectiveness of the proposed framework for systematically evaluating the anomaly detectors introduced in Chapter 6. The analyses are carried out for the two considered use cases: Electrocardiogram (ECG) data and accelerometer (ACC) telemetry acquired from a structural health monitoring system. A summary of the adopted labels is reported in Tables A.1 and A.2

Table A.1: Description of the anomaly labels.

Anomaly family	**Label**	**Description**
Power-increasing	GWN	Gaussian white noise anomaly
	IMPULSE	Impulse anomaly
	STEP	Step anomaly
	CONSTANT	Constant anomaly
	GNN	Gaussian narrowband noise anomaly
Power-invariant	MIXING w/ GWN	Mixing with Gaussian white noise disturbance
	MIXING w/ CONSTANT	Mixing with a constant disturbance
	TIME WARPING	Time warping anomaly
	SPECTRAL ALT.	Spectral alteration anomaly
	PS ALT.	Principal subspace alteration anomaly
Power-decreasing	SATURATION	Saturation anomaly
	DEAD-ZONE	Dead-zone anomaly

Table A.2: Description of the detector labels.

Detector family	Label	Description
PCA-based	SPE_k	Squared prediction error in case of k principal components
	T^2_k	Hotelling's T-squared test in case of k principal components
GD-based	AR_p	Autoregressive model of order p
	MD	Mahalanobis distance
ML-based	$OC_{kernel,\nu}$	One-Class Support Vector Machine with a non-linear mapping defined by *kernel* and the number of support vectors determined by ν
	LOF_h	Local Outlier Factor considering h neighbors
	IF_q	Isolation Forest with q estimators
Feature-based	energy	Energy of the vector
	TV	Total variation
	ZC	Number of zero-crossings
	pk-pk	Peak-to-peak value

A.1 Detection Performance

Figures A.1–A.6 show the performance of the considered detectors (PCA-based, ML-based, GD-based, and feature-based) in terms of P_D as a function of the deviation δ. These results extend the analysis presented in Tables 6.3, 6.4, 6.5, and 6.6 of Chapter 6, where P_D values were provided for two discrete deviation levels, $\delta = \{0.05, 0.8\}$. Each figure focuses on a different class of anomaly and on a different class of the normal signal. Figs. A.1-A.3 respectively show the performance of detectors for power-increasing, power-invariant, power-decreasing anomalies in case of ECG signals; while Figs. A.4-A.6 depict results for the ACC case.

Figure A.1 illustrates the detection performance, expressed in terms of P_D, of power-increasing anomalies as a function of δ for ECG signals. Each row refers to a different anomaly and each column corresponds to a different family of detectors. As expected, the detection performance increases with δ in all cases. The CONSTANT anomaly appears to be the most difficult to detect, especially for low values of δ. On the other hand, the GWN anomaly is the easiest to be detected. No single detector consistently outperforms the other ones for every anomaly and δ. For example, in the first column, SPE_{59} is the best with GWN, GNN, and IMPULSE and is in the average for the other two anomalies CONSTANT and STEP; while T^2_{52} is the best in CONSTANT and STEP and the worst in GWN, GNN, and IMPULSE. However, on average, detectors as SPE_{59}, LOF_5, MD tend to perform better than the other detectors of the same family. Feature-based detectors show high variability since their performance strongly depends on the type of anomaly.

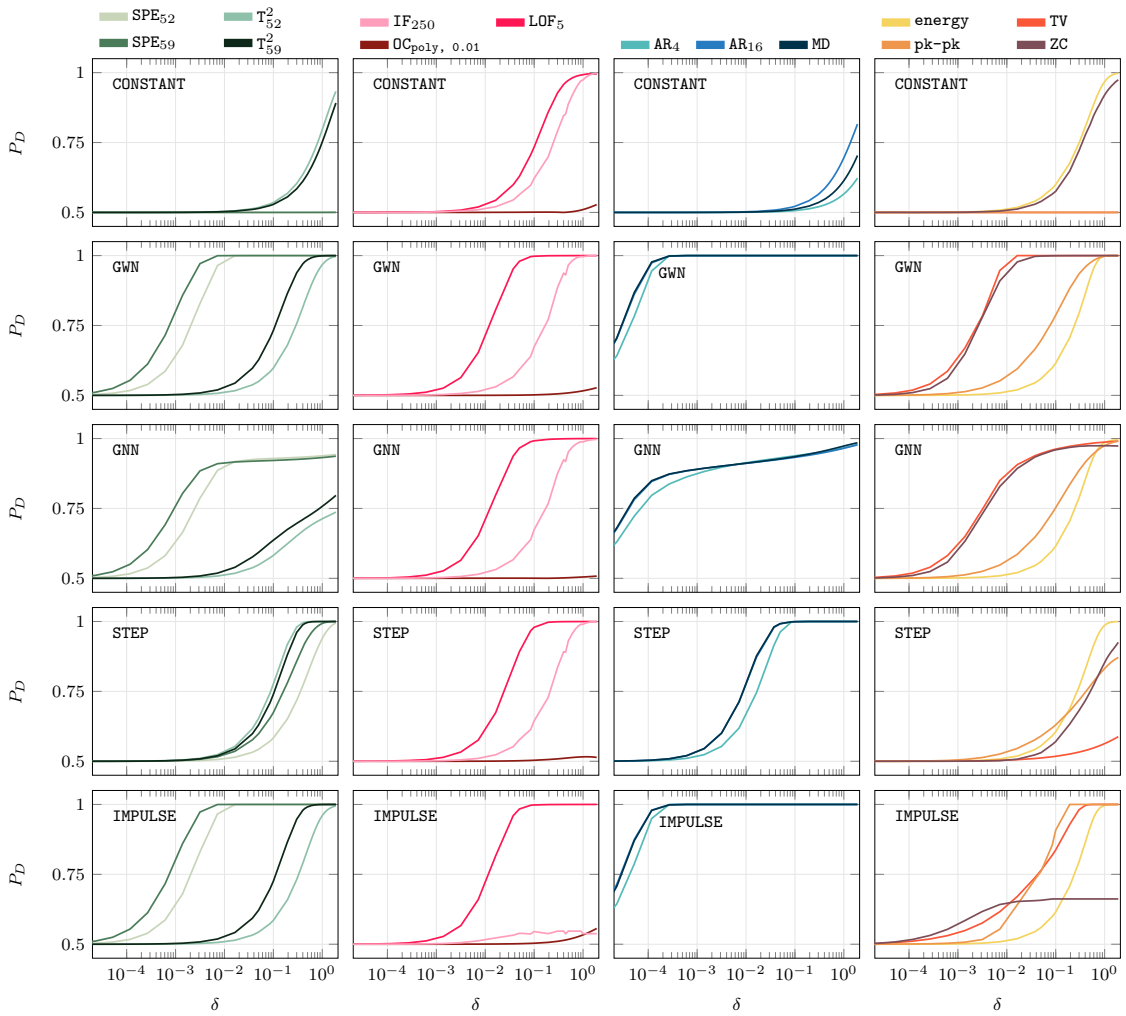

Figure A.1: Performance in terms of P_D of four families of detectors (PCA-based, ML-based, GD-based, feature-based) against deviation δ in case of ECG power-increasing anomalies.

Fig.A.2 shows the detection performance expressed in terms of P_D of power-invariant anomalies as a function of the deviation δ in the case of ECG signals. Each row illustrates the performance for a different anomaly and each column refers to a different family of detectors. As expected, in most cases the detection performance increases with δ. There is no strict monotonicity when GD-based detectors face a TIME WARPING anomaly and when pk-pk detector encounters PS ALT. and MIXING w/ GWN anomalies. As previously noted, the latter two anomalies pose similar challenges for the majority of detectors. Additionally, MIXING w/ GWN and PS ALT are the easiest to detect, especially for GD-based detectors. In contrast, the TIME WARPING anomaly appears to be the most difficult to detect,

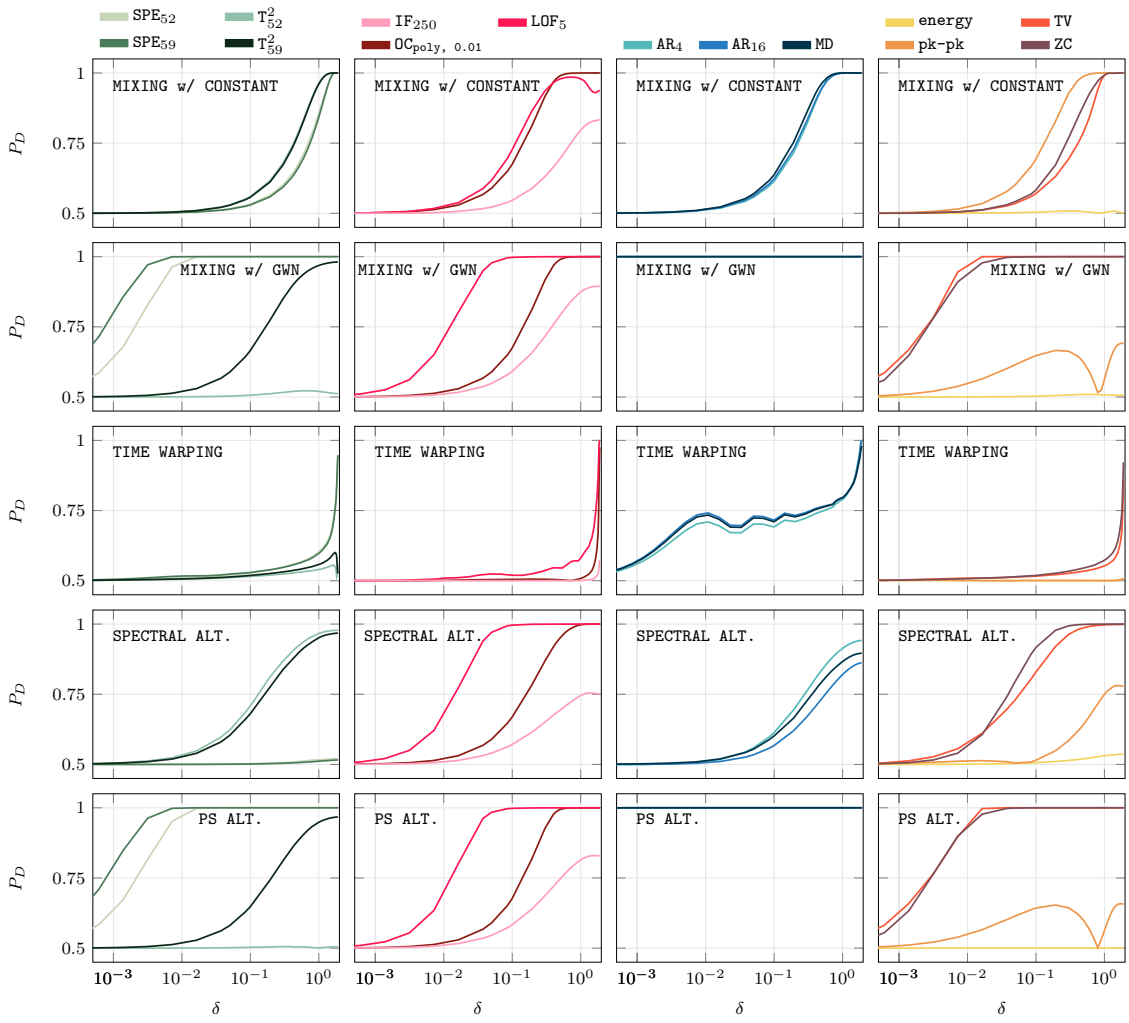

Figure A.2: Performance in terms of P_D of four families of detectors (PCA-based, ML-based, GD-based, feature-based) against deviation δ in case of ECG power-invariant anomalies.

especially for low values of δ. As before, among different detectors, there is no winner for every anomaly and δ. For example, T^2_{52}, completely fails at identifying PS ALT., TIME WARPING, and MIXING w/ GWN, while it outperforms other PCA-based detectors in identifying SPECTRAL ALT.. LOF_5 and MD tend to perform better than other detectors of the same family. Regarding feature-based detectors, as expected, energy-based detector fails at identifying power-invariant anomalies, while ZC, on average, tends to outperform other detectors of this family.

Fig.A.3 depicts the detection performance in terms of P_D of power-decreasing anomalies in case of ECG signals w.r.t. the level of deviation δ. Each row shows the performance for the two anomalies and each column shows the performance

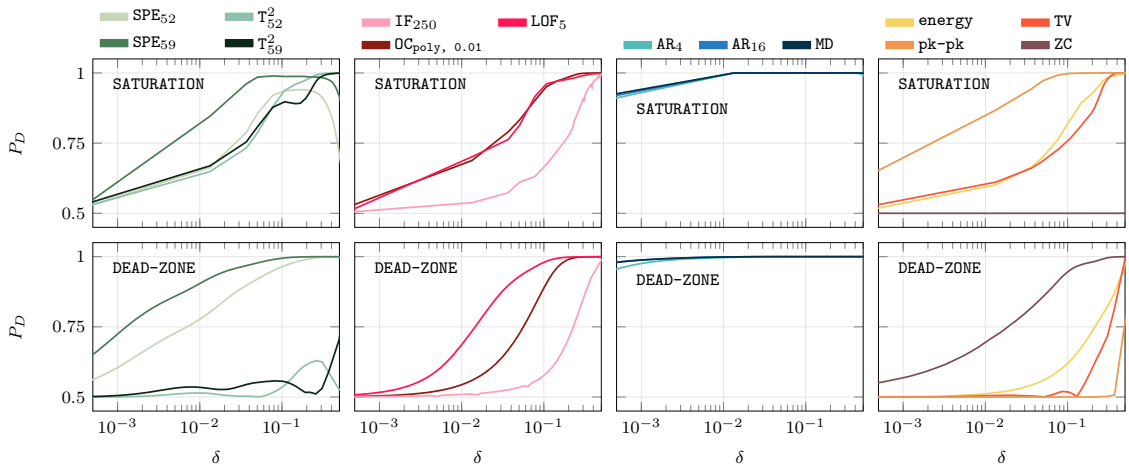

Figure A.3: Performance in terms of P_D of four families of detectors (PCA-based, ML-based, GD-based, feature-based) against deviation δ in case of ECG power-decreasing anomalies.

for different families of detectors. As expected, in most cases the detection performance increases with δ. There is no strict monotonicity for some PCA-based detectors. All GD-based detectors perform similarly and outperform other families of detectors for both anomalies. ZC-based detector completely fails at identifying SATURATION anomalies, while it is the best among feature-based detectors in case of DEAD-ZONE anomaly. In contrast, while pk-pk detector struggles at identifying DEAD-ZONE anomaly, it is the best among feature-based detectors in the case of SATURATION anomaly.

Fig.A.4 shows the detection performance in terms of P_D of power-increasing anomalies depending on the level of deviation δ in the case of ACC signals. Each row highlights the performance for a different anomaly and each column illustrates the performance for different families of detectors. As expected, in all cases the detection performance increases with δ. In contrast to ECG as in Fig.A.1, the CONSTANT and STEP anomalies present a comparable level of challenge to the detectors w.r.t. the other anomalies. T^2_{16} and TV detectors completely fail at detecting the CONSTANT and STEP anomalies. While ZC is good at detecting these two anomalies, it completely fails for the IMPULSE anomaly.

Fig.A.5 shows the detection performance in terms of P_D of power-invariant anomalies in the case of ACC signals w.r.t. the level of deviation δ. Each row refers to a different anomaly and each column corresponds to a different family of detectors. As expected, in most cases the detection performance increases with δ. The TIME WARPING and SPECTRAL ALT. anomalies appear to be the most difficult to detect, even for high values of δ. Feature-based detectors perform relatively well

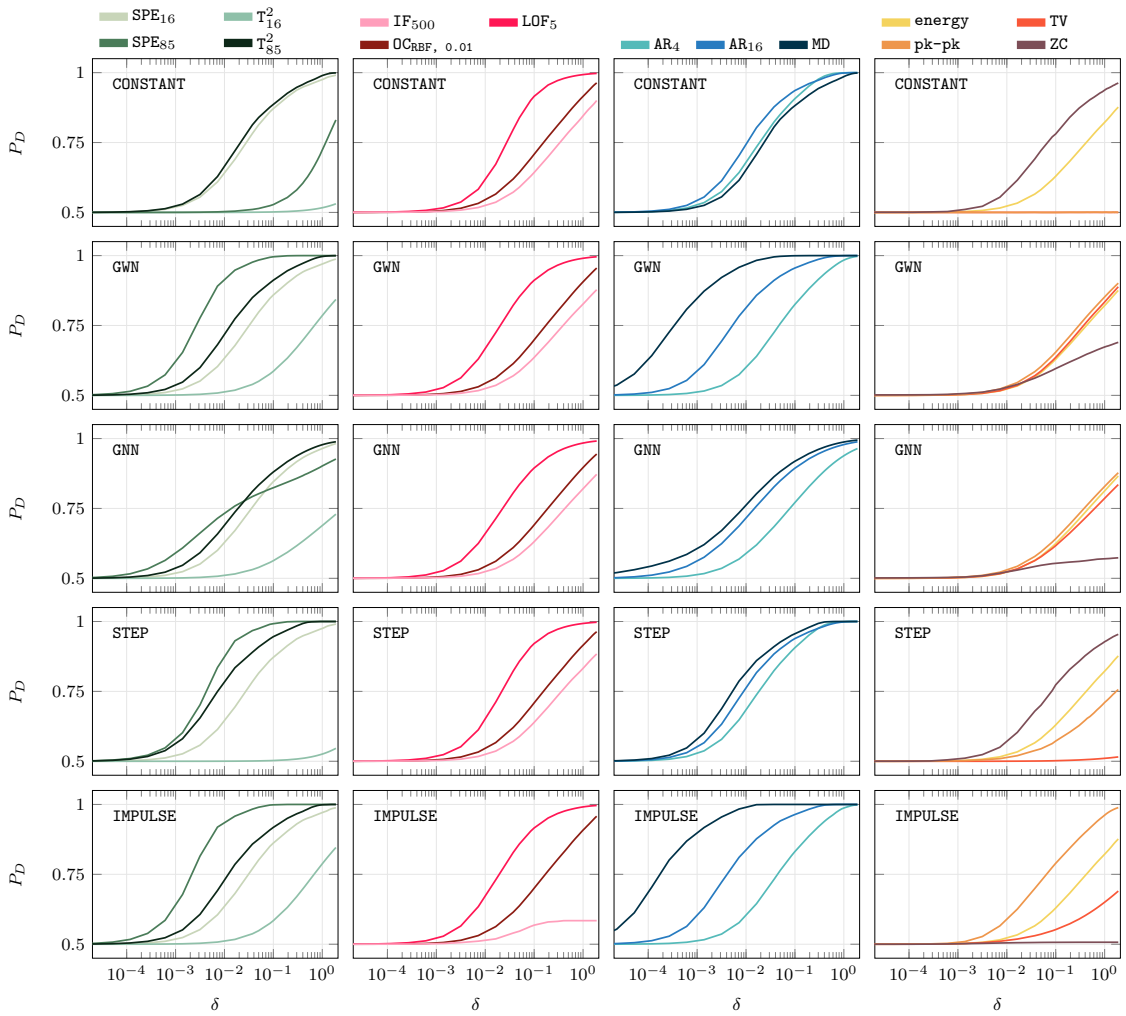

Figure A.4: Performance in terms of P_D of four families of detectors (PCA-based, ML-based, GD-based, feature-based) against deviation δ in case of ACC power-increasing anomalies.

only for MIXING w/ CONSTANT anomaly and struggle for the other anomalies. Note that, conversely to the ECG case (Fig.A.2), MIXING w/ GWN and PS. ALT. anomalies pose different challenges to the detectors.

Fig.A.6 depicts the detection performance in terms of P_D of power-decreasing anomalies in case of ACC signals w.r.t. the level of deviation δ. The two rows refer to the two types of power-decreasing anomalies and each column corresponds to a different family of detectors. ZC-based detector completely fails at identifying SATURATION anomalies, while it is the best among feature-based detectors in case of DEAD-ZONE anomaly. In contrast, while pk-pk detector struggles at identifying DEAD-ZONE anomalies, it is the best among feature-based detectors

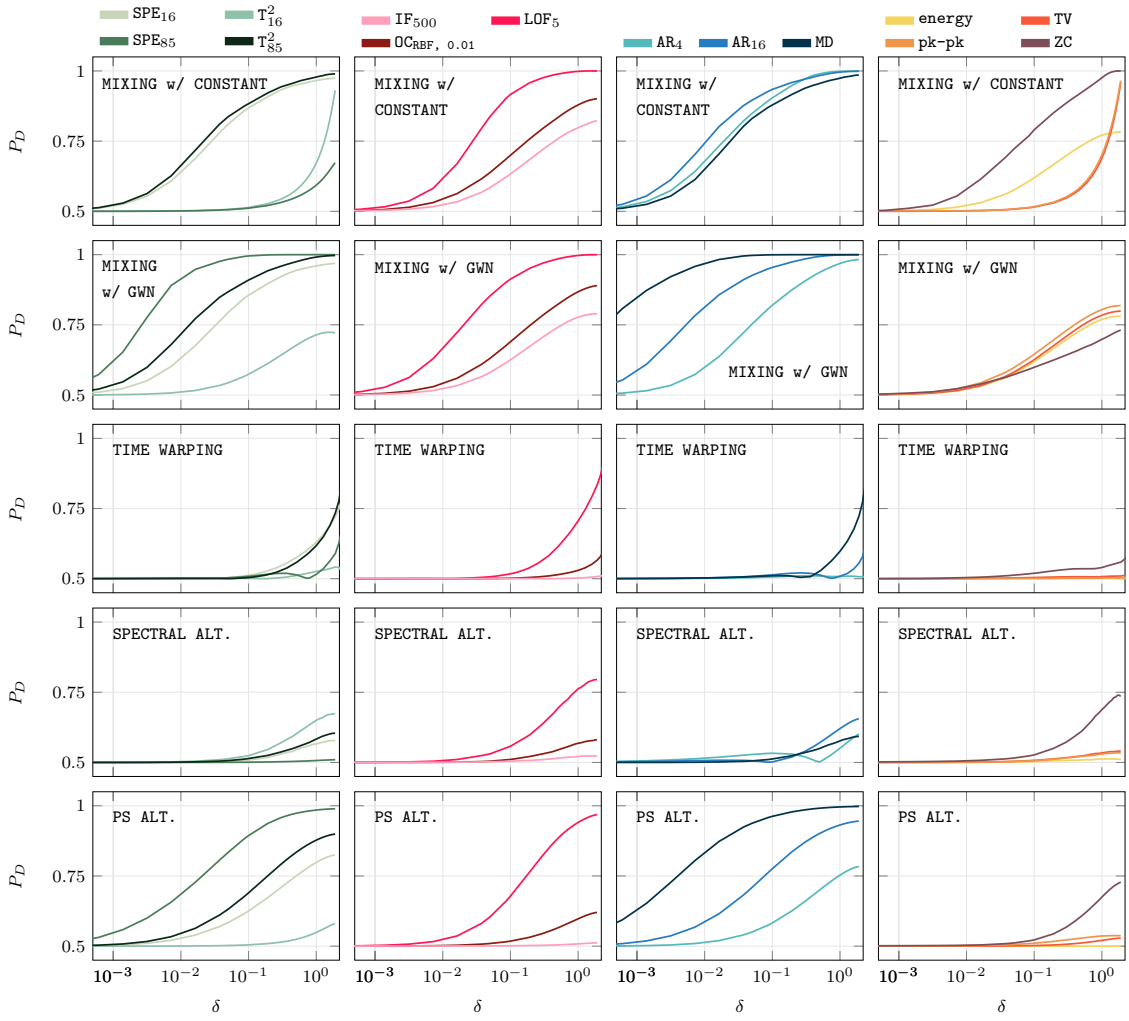

Figure A.5: Performance in terms of P_D of four families of detectors (PCA-based, ML-based, Gaussian distribution-based, feature-based) against deviation δ in case of ACC power-invariant anomalies.

in the case of the SATURATION anomaly.

A.2 Anomalies Mixtures

In this section, we present examples illustrating how different effects can be combined to generate anomalies that emulate real-world behaviors, where multiple effects are often superimposed. This follows the general anomaly model introduced in Equation (6.2) of the thesis.

$$\boldsymbol{x}^{\mathrm{ko}} = c\left(\boldsymbol{x}^{\mathrm{ok}}\right) + \boldsymbol{d}, \tag{A.1}$$

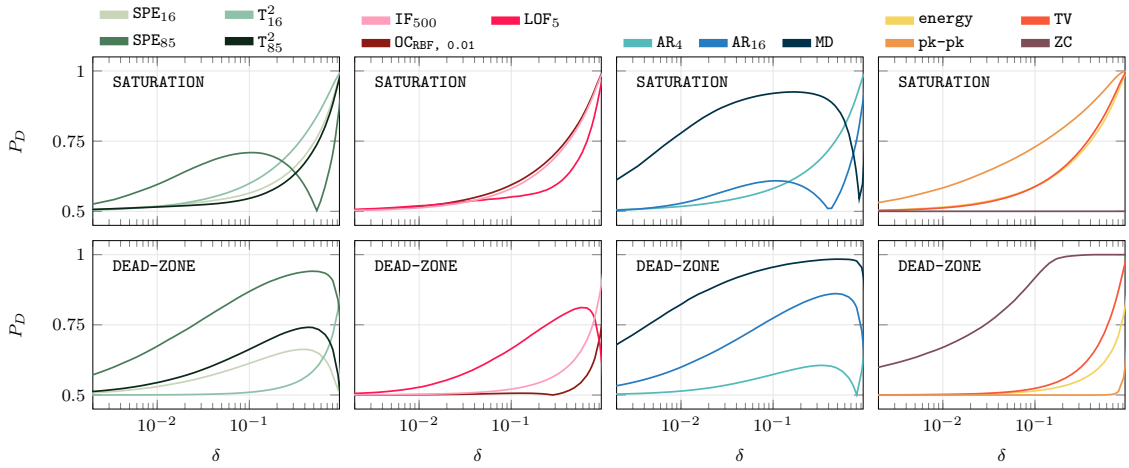

Figure A.6: Performance in terms of P_D of four families of detectors (PCA-based, ML-based, GD-based, feature-based) against deviation δ in case of ACC power-decreasing anomalies.

as already discussed, we can distinguish between anomalies obtained as an alteration through $c(\cdot)$ of the normal signal or anomalies that arise when an independent disturbance signal $\boldsymbol{d}$ is added to the normal signal. When we consider the superimposition of two different anomalies, we can have three cases: two intra-category combinations of $\boldsymbol{d}$-like anomalies or $c(\cdot)$-like anomalies, and an inter-category combination of one $c(\cdot)$ and one $\boldsymbol{d}$-like anomalies. While intra-category cases can be treated similarly, the inter-category case requires additional considerations. To provide an idea, we focus on *i*) CONSTANT (power-increasing) and SATURATION (power-decreasing) anomalies mixture and *ii*) SPECTRAL ALT. and PS ALT. power-invariant anomalies mixture.

A.2.1 Constant + saturation

According to the model in (A.1), to mix CONSTANT and SATURATION anomalies, a window of normal signal $\boldsymbol{x}^{\text{ok}}$ has to be first altered by a SATURATION anomaly and then by a CONSTANT anomaly. With this, it is sufficient to set

$$c\left(x_j^{\text{ok}}\right) = \begin{cases} x_{\text{SAT}} \text{sign}\left(x_j^{\text{ok}}\right) & \text{if } \left|x_j^{\text{ok}}\right| > x_{\text{SAT}} \\ x_j^{\text{ok}} & \text{otherwise} \end{cases} \tag{A.2}$$

$$d_j = \pm a \quad \text{for} \quad j = 0, \ldots, n-1. \tag{A.3}$$

Parameters x_{SAT} and a are chosen to satisfy a given level of deviation

$$\delta = \beta\delta + (1-\beta)\,\delta = \delta_{\text{SATUR.}} + \delta_{\text{CONST.}} \tag{A.4}$$

where β is the weight parameter controlling the relative intensity of the two anomalies and

$$\delta_{\text{SATUR.}} = \beta\delta = \sum_{\left|x_j^{\text{ok}}\right| > x_{\text{SAT}}} \left[x_j^{\text{ok}} - x_{\text{SAT}} \text{sign}\left(x_j^{\text{ok}}\right)\right]^2 \tag{A.5}$$

$$\delta_{\text{CONST.}} = (1-\beta)\,\delta = a^2 \tag{A.6}$$

The equation (A.4) is valid since SATURATION and CONSTANT anomalies are independent.

We provide examples of SATURATION-CONSTANT mixture for ECG signal, $\delta = 0.8$ and five values of β in Fig. A.7. It is evident how the combination transitions from the CONSTANT anomaly ($\beta = 0$) to SATURATION anomaly ($\beta = 1$) only. For intermediate β values, the mixture of anomalies appears as a weighted combination of the two single anomalies.

Note that, this example can be generalized to any other mixture where the first anomaly distorts $\boldsymbol{x}^{\text{ok}}$ through $c(\cdot)$, e.g., SPECTRAL ALT., DEAD-ZONE, and the second is an anomaly arising form a disturbance $\boldsymbol{d}$, e.g., GWN, GNN; or, alternatively, to a mixture where both anomalies arise from two different disturbances $\boldsymbol{d}$.

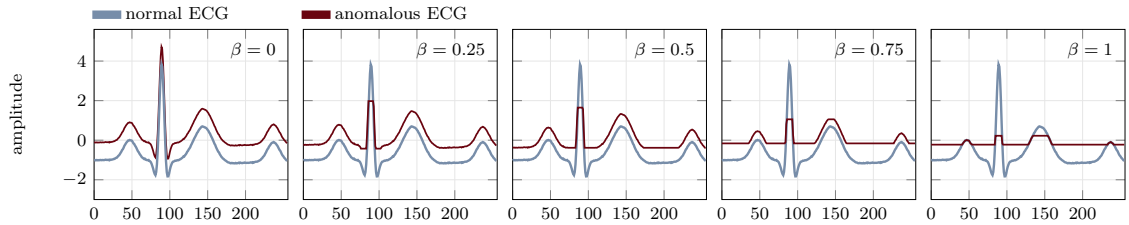

Figure A.7: Examples of SATURATION-CONSTANT mixture anomaly in case of ECG signal for five different values of weight $\beta = \{0, 0.25, 0.5, 0.75, 1\}$ increasing from left to right.

A.2.2 Sprectral alteration + PS alteration

To generate a mixture of SPECTRAL ALT. and PS ALT. anomalies, starting from (A.1), it is enough to set $c(\cdot)$ as a composition of SPECTRAL ALT. and PS ALT. anomalies and $\boldsymbol{d} = \boldsymbol{0}$:

$$c\left(\boldsymbol{x}^{\text{ok}}\right) = \boldsymbol{C}\boldsymbol{x}^{\text{ok}} = \boldsymbol{U}^{\text{ko}}\sqrt{\boldsymbol{\Lambda}^{\text{ko}}}\sqrt{\boldsymbol{\Lambda}^{\text{ok}}}^{-1}\boldsymbol{U}^{\text{ok}\top}\boldsymbol{x}^{\text{ok}} \tag{A.7}$$

$$d_j = 0 \quad \text{for} \quad j = 0, \ldots, n-1. \tag{A.8}$$

where $\boldsymbol{U}^{\text{ko}} = \boldsymbol{R}_{n,\theta_{\text{PS ALT.}}}\boldsymbol{U}^{\text{ok}}$ and $\sqrt{\boldsymbol{\Lambda}^{\text{ko}}}$ is built such that

$$\sqrt{\lambda_j^{\text{ko}}} \quad \text{for} \quad j = 0, \dots, \bar{n} \quad \text{are s.t.} \quad \boldsymbol{\ell}^{\text{ko}} = \boldsymbol{R}_{\bar{n},\theta_{\text{SPECTR. ALT.}}} \boldsymbol{\ell}^{\text{ok}} \tag{A.9}$$

$$\sqrt{\lambda_j^{\text{ko}}} = \sqrt{\lambda_j^{\text{ok}}} \quad \text{for} \quad j = \bar{n}, \dots, n-1 \tag{A.10}$$

with $\boldsymbol{\ell}^{\text{ok}} = \frac{1}{\sqrt{n\gamma}} \left(\sqrt{\lambda_0^{\text{ok}}}, \dots, \sqrt{\lambda_{\bar{n}-1}^{\text{ok}}} \right)^\top$ and $\boldsymbol{\ell}^{\text{ko}} = \frac{1}{\sqrt{n\gamma}} \left(\sqrt{\lambda_0^{\text{ko}}}, \dots, \sqrt{\lambda_{\bar{n}-1}^{\text{ko}}} \right)^\top$. The angles $\theta_{\text{SPECTR. ALT.}}$ and $\theta_{\text{PS ALT.}}$ are fixed to satisfy

$$\delta = g_\beta \left[\delta'\right] = g_\beta \left[\beta\delta' + (1-\beta)\,\delta'\right] = g_\beta \left[\delta_{\text{SPECTR. ALT.}} + \delta_{\text{PS ALT.}}\right] \tag{A.11}$$

$$\delta_{\text{SPECTR. ALT.}} = \beta\delta' = 2\gamma \left[1 - \cos(\theta_{\text{SPECTR. ALT.}})\right] \tag{A.12}$$

$$\delta_{\text{PS ALT.}} = (1-\beta)\,\delta' = 2\left[1 - \cos(\theta_{\text{PS ALT.}})\right] \tag{A.13}$$

with $g_\beta = 2\left[1 - \frac{1}{n}\text{tr}\left(\boldsymbol{C}_{\beta,\delta'}\right)\right]$ the function that connects the effective deviation δ with the imposed one δ'.

Examples of SPECTRAL ALT.-PS ALT. mixture for ECG signal, $\delta = 0.05$ and five values of β are shown in Fig. A.8. It is noticeable how by acting on β the mixture transitions from the PS ALT. anomaly ($\beta = 0$) to SPECTRAL ALT. anomaly ($\beta = 1$). For intermediate β values, the resulting anomaly appears as a weighted combination of the two.

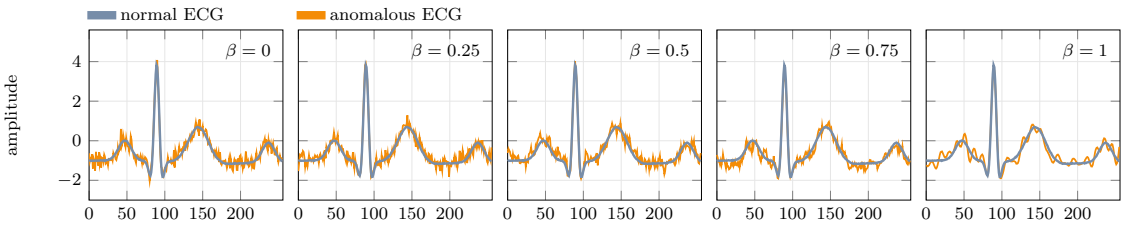

Figure A.8: Examples of SPECTRAL ALT.-PS ALT. mixture anomaly in case of ECG signal for five different values of weight $\beta = \{0, 0.25, 0.5, 0.75, 1\}$ increasing from left to right.

Note that, the anomaly deriving from SPECTRAL ALT.-PS ALT. mixture can be further mixed with disturbances following a procedure similar to the one illustrated in the previous example.

Bibliography

[1] Xiongzi Chen, Jinsong yu, Diyin Tang, and Wang Yingxun. Remaining useful life prognostic estimation for aircraft subsystems or components: A review. *Proceedings - IEEE 2011 10th International Conference on Electronic Measurement and Instruments, ICEMI 2011*, 2, 08 2011. doi: 10.1109/ICEMI.2011.6037773.

[2] Yixin Zhao, Baoping Cai, Valerio Cozzani, and Yiliu Liu. Failure dependence and cascading failures: A literature review and research opportunities. *Reliability Engineering & System Safety*, 256:110766, 2025. ISSN 0951-8320. doi: 10.1016/j.ress.2024.110766.

[3] Andrew K.S. Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, 2006. ISSN 0888-3270. doi: 10.1016/j.ymssp.2005.09.012.

[4] Luca Biggio and Iason Kastanis. Prognostics and health management of industrial assets: Current progress and road ahead. *Frontiers in Artificial Intelligence*, Volume 3 - 2020, 2020. ISSN 2624-8212. doi: 10.3389/frai.2020.578613.

[5] Marek Molęda, Bożena Małysiak-Mrozek, Weiping Ding, Vaidy Sunderam, and Dariusz Mrozek. From corrective to predictive maintenance—a review of maintenance approaches for the power industry. *Sensors*, 23(13), 2023. ISSN 1424-8220. doi: 10.3390/s23135970.

[6] Roberto Sala, Emmanuel Francalanza, and Simone Arena. A review on three decades of manufacturing maintenance research: past, present and future directions. *Production & Manufacturing Research*, 13, 02 2025. doi: 10.1080/21693277.2025.2469037.

[7] Prashant Kumar, Izaz Raouf, and Heung Soo Kim. Review on prognostics and health management in smart factory: From conventional to deep learning perspectives. *Engineering Applications of Artificial Intelligence*, 126:107126, 2023. ISSN 0952-1976. doi: 10.1016/j.engappai.2023.107126.

[8] Jay Lee, Fangji Wu, Wenyu Zhao, Mahsa Ghaffari, Linxia Liao, and David Siegel. Prognostics and health management design for rotary machinery systems—reviews, methodology and applications. *Mechanical Systems and Signal Processing*, 42:314–334, 01 2014. doi: 10.1016/j.ymssp.2013.06.004.

[9] Barry Dowdeswell, Roopak Sinha, and Stephen G. MacDonell. Finding faults: A scoping study of fault diagnostics for industrial cyber–physical systems. *Journal of Systems and Software*, 168:110638, 2020. ISSN 0164-1212. doi: 10.1016/j.jss.2020.110638.

[10] Martin Törngren and Ulf Sellgren. *Complexity Challenges in Development of Cyber-Physical Systems: Essays Dedicated to Edward A. Lee on the Occasion of His 60th Birthday*, pages 478–503. Springer, Cham, 07 2018. ISBN 978-3-319-95245-1. doi: 10.1007/978-3-319-95246-8_27.

[11] Xiao-sheng Si, Wenbin Wang, Changhua hu, and Dong-Hua Zhou. Remaining ueful life estimation—a review on the statistical data driven approaches. *European Journal of Operational Research*, 213:1–14, 08 2011. doi: 10.1016/j.ejor.2010.11.018.

[12] Carl Byington, M. Watson, D. Edwards, and P. Stoelting. A model-based approach to prognostics and health management for flight control actuators. *IEEE Aerospace Conference Proceedings*, 6:3551 – 3562 Vol.6, 04 2004. doi: 10.1109/AERO.2004.1368172.

[13] Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. Machinery health prognostics: A systematic review from data acquisition to rul prediction. *Mechanical Systems and Signal Processing*, 104:799–836, 05 2018. doi: 10.1016/j.ymssp.2017.11.016.

[14] Yitong Liu, Jiarui Wen, and Guoqiang Wang. A comprehensive overview of remaining useful life prediction: From traditional literature review to scientometric analysis. *Machine Learning with Applications*, 21:100704, 09 2025. doi: 10.1016/j.mlwa.2025.100704.

[15] Yong Yu, Changhua Hu, Xiao-sheng Si, and Jianxun Zhang. Degradation data-driven remaining useful life estimation in the absence of prior degradation knowledge. *Journal of Control Science and Engineering*, 2017:1–11, 12 2017. doi: 10.1155/2017/4375690.

[16] Jianxin Lei, Wenbo Zhang, Zhinong Jiang, and Zhilong Gao. A review: Prediction method for the remaining useful life of the mechanical system. *Journal of Failure Analysis and Prevention*, 22, 11 2022. doi: 10.1007/s11668-022-01532-4.

[17] Tianyi Wang, Jianbo Yu, David Siegel, and Jay Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 International Conference on Prognostics and Health Management*, pages 1–6, 2008. doi: 10.1109/PHM.2008.4711421.

[18] Wennian Yu, II Yong Kim, and Chris Mechefske. Remaining useful life estimation using a bidirectional recurrent neural network based autoencoder scheme. *Mechanical Systems and Signal Processing*, 129:764–780, 2019. ISSN 0888-3270. doi: 10.1016/j.ymssp.2019.05.005.

[19] Jianghong Zhou, Jiahong Yang, and Yi Qin. A systematic overview of health indicator construction methods for rotating machinery. *Engineering Applications of Artificial Intelligence*, 138:109356, 2024. ISSN 0952-1976. doi: 10.1016/j.engappai.2024.109356.

[20] Qinghua Hu, Weiwei Pan, Lei Zhang, David Zhang, Yanping Song, Maozu Guo, and Daren Yu. Feature selection for monotonic classification. *IEEE Transactions on Fuzzy Systems*, 20(1):69–81, 2012. doi: 10.1109/TFUZZ.2011.2167235.

[21] Liang Guo, Yaoxiang Yu, Andongzhe Duan, Hongli Gao, and Jiangquan Zhang. An unsupervised feature learning based health indicator construction method for performance assessment of machines. *Mechanical Systems and Signal Processing*, 167:108573, 2022. ISSN 0888-3270. doi: 10.1016/j.ymssp.2021.108573.

[22] Pengfei Lin and Jizhong Tao. A novel bearing health indicator construction method based on ensemble stacked autoencoder. In *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pages 1–9, 2019. doi: 10.1109/ICPHM.2019.8819405.

[23] Adrian Cubillo, Suresh Perinpanayagam, and Manuel Esperon-Miguez. A review of physics-based models in prognostics: Application to gears and bearings of rotating machinery. *Advances in Mechanical Engineering*, 8(8): 1687814016664660, 2016. doi: 10.1177/1687814016664660.

[24] Naipeng Li, Yaguo Lei, Tao Yan, Ningbo Li, and Tianyu Han. A wiener-process-model-based method for remaining useful life prediction considering unit-to-unit variability. *IEEE Transactions on Industrial Electronics*, PP:1–1, 05 2018. doi: 10.1109/TIE.2018.2838078.

[25] Xuefeng Chen, Zhongjie Shen, Zhengjia He, Chuang Sun, and Zhiwen Liu. Remaining life prognostics of rolling bearing based on relative features and multivariable support vector machine. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 227 (12):2849–2860, 2013. doi: 10.1177/0954406212474395.

[26] Kailong Liu, Xiaosong Hu, Zhongbao Wei, Yi Li, and Yan Jiang. Modified gaussian process regression models for cyclic capacity prediction of lithium-ion batteries. *IEEE Transactions on Transportation Electrification*, PP:1–1, 09 2019. doi: 10.1109/TTE.2019.2944802.

[27] Dongdong Li and Lin Yang. Remaining useful life prediction of lithium battery using convolutional neural network with optimized parameters. In *2020 5th Asia Conference on Power and Electrical Engineering (ACPEE)*, pages 840–844, 2020. doi: 10.1109/ACPEE48638.2020.9136289.

[28] Ji-Yan Wu, Min Wua, Zhenghua Chen, Xiaoli li, and Ruqiang Yan. Degradation-aware remaining useful life prediction with lstm autoencoder. *IEEE Transactions on Instrumentation and Measurement*, PP:1–1, 02 2021. doi: 10.1109/TIM.2021.3055788.

[29] Yuhang Duan, Honghui Li, Mengqi He, and Dongdong Zhao. A bigru autoencoder remaining useful life prediction scheme with attention mechanism and skip connection. *IEEE Sensors Journal*, 21(9):10905–10914, 2021. doi: 10.1109/JSEN.2021.3060395.

[30] Ansi Zhang, Honglei Wang, Shaobo Li, Yuxin Cui, Zhonghao Liu, Guanci Yang, and Jianjun Hu. Transfer learning with deep recurrent neural networks for remaining useful life estimation. *Applied Sciences*, 8(12), 2018. ISSN 2076-3417. doi: 10.3390/app8122416.

[31] Xinyuan Liao, Shaowei Chen, Pengfei Wen, and Shuai Zhao. Remaining useful life with self-attention assisted physics-informed neural network. *Advanced Engineering Informatics*, 58, oct 2023. ISSN 1474-0346. doi: 10.1016/j.aei.2023.102195.

[32] Yi Wu, Wei Li, Youren Wang, and Kai Zhang. Remaining useful life prediction of lithium-ion batteries using neural network and bat-based particle filter. *IEEE Access*, 7:54843–54854, 2019. doi: 10.1109/ACCESS.2019.2913163.

[33] Fan Li and Jiuping Xu. A new prognostics method for state of health estimation of lithium-ion batteries based on a mixture of gaussian process models and particle filter. *Microelectronics Reliability*, 55, 04 2015. doi: 10.1016/j.microrel.2015.02.025.

[34] Abhinav Saxena, Jose Celaya, Edward Balaban, Kai Goebel, Bhaskar Saha, Sankalita Saha, and Mark Schwabacher. Metrics for evaluating performance of prognostic techniques. In *2008 International Conference on Prognostics and Health Management*, pages 1–17, 2008. doi: 10.1109/PHM.2008.4711436.

[35] Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*, pages 1–9, 2008. doi: 10.1109/PHM.2008.4711414.

[36] Pankaj Malhotra, Vishnu Tv, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Multi-sensor prognostics using an unsupervised health index based on lstm encoder-decoder. *1st SIGKDD Workshop on Machine Learning for Prognostics and Health Management*, 08 2016. doi: 10.48550/arXiv.1608.06154.

[37] Chinedu I. Ossai. Prognosis and remaining useful life estimation of lithium-ion battery with optimal multi-level particle filter and genetic algorithm. *Batteries*, 4(2), 2018. ISSN 2313-0105. doi: 10.3390/batteries4020015.

[38] Bahareh Tajiani and Jørn Vatn. Adaptive remaining useful life prediction framework with stochastic failure threshold for experimental bearings with different lifetimes under contaminated condition. *International Journal of System Assurance Engineering and Management*, 14, 06 2023. doi: 10.1007/s13198-023-01979-0.

[39] Abhinav Saxena, Jose Celaya, Bhaskar Saha, Sankalita Saha, and Kai Goebel. Metrics for offline evaluation of prognostic performance. *International Journal of Prognostics and Health Management*, 1:2153–2648, 01 2010. doi: 10.36001/ijphm.2010.v1i1.1336.

[40] Marcia L. Baptista, Elsa M.P. Henriques, and Kai Goebel. More effective prognostics with elbow point detection and deep learning. *Mechanical Systems and Signal Processing*, 146:106987, 2021. ISSN 0888-3270. doi: 10.1016/j.ymssp.2020.106987.

[41] K. Etienne, S. Bosse, T. Laloix, and A Cabarbaye. Prognosis and health monitoring applications in satellite systems. In *Safety, Reliability and Risk Analysis*, pages 1679–1684, 09 2013. ISBN 9780429227714. doi: 10.1201/b15938-255.

[42] Qinyi Li and Xin Peng. Application of large-data-driven phm technology in satellite test and on-orbit management. In *2017 Prognostics and System Health Management Conference (PHM-Harbin)*, pages 1–5, 2017. doi: 10.1109/PHM.2017.8079214.

[43] Kezhen Song, Chen Zhao, Jie Liu, Hao Zhang, and Zhiyu Li. Research ON satellites fault diagnosis method based on artificial intelligence. *Journal of Physics: Conference Series*, 2005(1):012102, aug 2021. doi: 10.1088/1742-6596/2005/1/012102.

[44] Yuhuang Zheng. Predicting remaining useful life based on hilbert–huang entropy with degradation model. *Journal of Electrical and Computer Engineering*, 2019:1–11, 02 2019. doi: 10.1155/2019/3203959.

[45] Lei Xiao, Junxuan Tang, Xinghui Zhang, Eric Bechhoefer, and Siyi Ding. Remaining useful life prediction based on intentional noise injection and feature reconstruction. *Reliability Engineering & System Safety*, 215:107871, 2021. ISSN 0951-8320. doi: 10.1016/j.ress.2021.107871.

[46] Takehisa Yairi, Naoya Takeishi, Tetsuo Oda, Yuta Nakajima, Naoki Nishimura, and Noboru Takata. A data-driven health monitoring method for satellite housekeeping data based on probabilistic clustering and dimensionality reduction. *IEEE Transactions on Aerospace and Electronic Systems*, 53(3):1384–1401, 2017. doi: 10.1109/TAES.2017.2671247.

[47] Cong Chen, Tianhua Xu, Guang Wang, and Bo Li. Railway turnout system rul prediction based on feature fusion and genetic programming. *Measurement*, 151:107162, 2020. ISSN 0263-2241. doi: 10.1016/j.measurement.2019.107162.

[48] Shixin Ji, Xuehao Han, Yichun Hou, Yong Song, and Qingfu Du. Remaining useful life prediction of airplane engine based on pca–blstm. *Sensors*, 20(16), 2020. ISSN 1424-8220. doi: 10.3390/s20164537.

[49] Miguel Martínez-García, Yu Zhang, Jiafu Wan, and Jason McGinty. Visually interpretable profile extraction with an autoencoder for health monitoring of industrial systems. In *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 649–654, 2019. doi: 10.1109/ICARM.2019.8834281.

[50] Bang Xiang Yong, Yasmin Fathy, and Alexandra Brintrup. Bayesian autoencoders for drift detection in industrial environments. In *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*, pages 627–631, 2020. doi: 10.1109/MetroInd4.0IoT48571.2020.9138306.

[51] Erkki Oja and Juha Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106(1):69–84, 1985. ISSN 0022-247X. doi: 10.1016/0022-247X(85)90131-3.

[52] Laura Balzano, Yuejie Chi, and Yue M. Lu. Streaming pca and subspace tracking: The missing data case. *Proceedings of the IEEE*, 106(8):1293–1310, 2018. doi: 10.1109/JPROC.2018.2847041.

[53] Kai Zhong, Min Han, and Bing Han. Data-driven based fault prognosis for

industrial systems: a concise overview. *IEEE/CAA Journal of Automatica Sinica*, PP:1–16, 12 2019. doi: 10.1109/JAS.2019.1911804.

[54] Yuxin Wen, Md. Fashiar Rahman, Honglun Xu, and Tzu-Liang Bill Tseng. Recent advances and trends of predictive maintenance from data-driven machine prognostics perspective. *Measurement*, 187:110276, 2022. ISSN 0263-2241. doi: 10.1016/j.measurement.2021.110276.

[55] C. Okoh, R. Roy, J. Mehnen, and L. Redding. Overview of remaining useful life prediction techniques in through-life engineering services. *Procedia CIRP*, 16:158–163, 2014. ISSN 2212-8271. doi: 10.1016/j.procir.2014.02.006.

[56] Farzaneh Ahmadzadeh and Jan Lundberg. Remaining useful life estimation: review. *International Journal of System Assurance Engineering and Management*, 5, 12 2013. doi: 10.1007/s13198-013-0195-0.

[57] Gaowei Xu, Min Liu, Jingwei Wang, Yumin Ma, Jian Wang, Fei Li, and Weiming Shen. Data-driven fault diagnostics and prognostics for predictive maintenance: A brief overview *. In *2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)*, pages 103–108, 08 2019. doi: 10.1109/COASE.2019.8843068.

[58] Shuai Yang, Chaoqin Liu, Xue Zhou, Wei Liang, and Qiang Miao. Investigation on data-driven life prediction methods. In *2012 International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, pages 674–680, 2012. doi: 10.1109/ICQR2MSE.2012.6246322.

[59] Narendhar Gugulothu, Vishnu Tv, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Predicting remaining useful life using time series embeddings based on recurrent neural networks. *International Journal of Prognostics and Health Management*, 9, 08 2017. doi: 10.48550/arXiv.1709.01073.

[60] Chong Zhang, Pin Lim, A. K. Qin, and Kay Chen Tan. Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2306–2318, 2017. doi: 10.1109/TNNLS.2016.2582798.

[61] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018. ISSN 0951-8320. doi: 10.1016/j.ress.2017.11.021.

[62] Kwok-Leung Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, and Wenbin Wang. Prognostics and health management: A review on data driven approaches.

Mathematical Problems in Engineering, 2015:1–17, 05 2015. doi: 10.1155/2015/793161.

[63] Zhicheng Xia, Qingluan Guan, Yifan Gao, Xiaoying Chen, and Xiaojie Zhai. Review on remaining useful life prediction methods of bearing. In *2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)*, pages 429–433, 2020. doi: 10.1109/PHM-Jinan48558.2020.00083.

[64] Youdao Wang, Yifan Zhao, and Pavan Addepalli. Remaining useful life prediction using deep learning approaches: A review. In *Proceedings of the 8th International Conference on Through-Life Engineering Services – TESConf 2019*, volume 49, 10 2019. doi: 10.1016/j.promfg.2020.06.015.

[65] Amine Abouaomar, Soumaya Cherkaoui, Zoubeir Mlika, and Abdellatif Kobbane. Resource provisioning in edge computing for latency sensitive applications. *IEEE Internet of Things Journal*, PP:1–1, 01 2021. doi: 10.1109/JIOT.2021.3052082.

[66] Hongbo Liu, Ping Song, Youtian Qie, and Yifan Li. Real-time prediction method of remaining useful life based on tinyml. In *2022 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 693–698, 2022. doi: 10.1109/RCAR54675.2022.9872225.

[67] Partha Pratim Ray. A review on tinyml: State-of-the-art and prospects. *Journal of King Saud University - Computer and Information Sciences*, 34 (4):1595–1623, 2022. ISSN 1319-1578. doi: 10.1016/j.jksuci.2021.11.019.

[68] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016. doi: 10.1109/JIOT.2016.2579198.

[69] Georgios Athanasakis, Gabriel Filios, Ioannis Katsidimas, Sotiris Nikoletseas, and Stefanos H. Panagiotou. Tinyml-based approach for remaining useful life prediction of turbofan engines. In *2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8, 2022. doi: 10.1109/ETFA52439.2022.9921629.

[70] Abubakar Bala, Rahimi Rashid, Idris Ismail, Diego Oliva, Noryanti Muhammad, Sadiq Sait, Khaled Al-Utaibi, Temitope Ibrahim Amosa, and Kamran Memon. Artificial intelligence and edge computing for machine maintenance-review. *Artificial Intelligence Review*, 57, 04 2024. doi: 10.1007/s10462-024-10748-9.

[71] Mohammed Zubair Shamim. Tinyml model for classifying hazardous volatile

organic compounds using low-power embedded edge sensors: Perfecting factory 5.0 using edge ai. *IEEE Sensors Letters*, PP:1–4, 09 2022. doi: 10.1109/LSENS.2022.3201398.

[72] I Nyoman Kusuma Wardana, Suhaib A. Fahmy, and Julian W. Gardner. Tinyml models for a low-cost air quality monitoring device. *IEEE Sensors Letters*, 7(11):1–4, 2023. doi: 10.1109/LSENS.2023.3315249.

[73] Lukas Stäcker, Juncong Fei, Philipp Heidenreich, Frank Bonarens, Jason Rambach, Didier Stricker, and Christoph Stiller. Deployment of deep neural networks for object detection on edge ai devices with runtime optimization. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1015–1022, 2021. doi: 10.1109/ICCVW54120.2021.00118.

[74] Christian Jay C. Adducul, John Rufino I. Macasaet, and Nestor Michael C. Tiglao. Edge-based battery remaining useful life estimation using deep learning. In *2023 International Conference on Smart Applications, Communications and Networking (SmartNets)*, pages 1–6, 2023. doi: 10.1109/SmartNets58706.2023.10215733.

[75] Lei Ren, Shixiang Li, Yuanjun Laili, and Lin Zhang. Btcan: A binary trend-aware network for industrial edge intelligence and application in aero-engine rul prediction. *IEEE Transactions on Instrumentation and Measurement*, 73:1–10, 2024. doi: 10.1109/TIM.2024.3428643.

[76] Zijie Chen, Yiming Gao, and Junrui Liang. Lopdm: A low-power on-device predictive maintenance system based on self-powered sensing and tinyml. *IEEE Transactions on Instrumentation and Measurement*, 72:1–13, 2023. doi: 10.1109/TIM.2023.3308251.

[77] Silvia Onofri, Alex Marchioni, Gianluca Setti, Mauro Mangia, and Riccardo Rovatti. Multi-class similarity-based approach for remaining useful life estimation. In *2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 01–06, 2024. doi: 10.1109/I2MTC60896.2024.10560572.

[78] Kangze Su, Biao Deng, Shengjin Tang, Xiaoyan Sun, Pengya Fang, Xiaosheng Si, and Xuebing Han. Remaining useful life prediction of lithium-ion batteries based on a cubic polynomial degradation model and envelope extraction. *Batteries*, 9:441, 08 2023. doi: 10.3390/batteries9090441.

[79] Silvia Onofri, Andriy Enttsel, Livia Manovi, Alex Marchioni, Salvatore Cognetta, Francesco Corallo, Carlo Ciancarelli, Mauro Mangia, Riccardo Rovatti, and Gianluca Setti. Autoencoder-based features extraction for the health monitoring in the space domain. In *2023 IEEE 66th International*

Midwest Symposium on Circuits and Systems (MWSCAS), pages 458–462, 2023. doi: 10.1109/MWSCAS57524.2023.10406124.

[80] Felix O. Heimes. Recurrent neural networks for remaining useful life estimation. In *2008 International Conference on Prognostics and Health Management*, pages 1–6, 2008. doi: 10.1109/PHM.2008.4711422.

[81] Xin Wang, Yi Li, Yaxi Xu, Xiaodong Liu, Tao Zheng, and Bo Zheng. Remaining useful life prediction for aero-engines using a time-enhanced multi-head self-attention model. *Aerospace*, 10:80, 01 2023. doi: 10.3390/aerospace10010080.

[82] Michael Pecht and Rubyca Jaai. A prognostics and health management roadmap for information and electronics-rich systems. *Microelectronics Reliability*, 50(3):317–323, 2010. ISSN 0026-2714. doi: 10.1016/j.microrel.2010.01.006.

[83] Charu C. Aggarwal. *Outlier Analysis*. Springer Cham, 2017. ISBN 9978-3-319-47577-6. doi: 10.1007/978-3-319-47578-3.

[84] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), jul 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882.

[85] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, 54:1–33, 04 2021. doi: 10.1145/3444690.

[86] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A. Zuluaga. Do deep neural networks contribute to multivariate time series anomaly detection? *Pattern Recognition*, 132:108945, 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2022.108945.

[87] Jessica Lin, Eamonn Keogh, Ada Fu, and Helga Van Herle. Approximations to magic: finding unusual medical time series. In *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pages 329–334, 2005. doi: 10.1109/CBMS.2005.34.

[88] Kevin Ni, Nithya Ramanathan, Mohamed Nabil Hajj Chehade, Laura Balzano, Sheela Nair, Sadaf Zahedi, Eddie Kohler, Greg Pottie, Mark Hansen, and Mani Srivastava. Sensor network data fault types. *ACM Trans. Sen. Netw.*, 5(3), 2009. ISSN 1550-4859. doi: 10.1145/1525856.1525863.

[89] Abhishek Sharma, Leana Golubchik, and Ramesh Govindan. On the prevalence of sensor faults in real-world deployments. In *2007 4th Annual IEEE*

Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pages 213–222, 2007. doi: 10.1109/SAHCN.2007.4292833.

[90] Sana Ullah Jan, Young-Doo Lee, Jungpil Shin, and Insoo Koo. Sensor fault classification based on support vector machine and statistical time-domain features. *IEEE Access*, 5:8682–8690, 2017. doi: 10.1109/ACCESS.2017.2705644.

[91] Han Qiu Zhang and Yong Yan. A wavelet-based approach to abrupt fault detection and diagnosis of sensors. *IEEE Transactions on Instrumentation and Measurement*, 50(5):1389–1396, 2001. doi: 10.1109/19.963215.

[92] Quoc Khanh Dang and Young Soo Suh. Sensor saturation compensated smoothing algorithm for inertial sensor based motion tracking. *Sensors*, 14 (5):8167–8188, 2014. ISSN 1424-8220. doi: 10.3390/s140508167.

[93] Cong Huang, Bo Shen, Lei Zou, and Yuxuan Shen. Event-triggering state and fault estimation for a class of nonlinear systems subject to sensor saturations. *Sensors*, 21(4), 2021. ISSN 1424-8220. doi: 10.3390/s21041242.

[94] Nithya Ramanathan, Tom Schoellhammer, Deborah Estrin, Mark Hansen, Tom Harmon, Eddie Kohler, and Mani Srivastava. The final frontier: Embedding networked sensors in the soil. Cens technical report 68, UCLA Center for Embedded Network Sensing (CENS), November 2006.

[95] Sinvaldo R. Moreno, Leandro dos Santos Coelho, Helon V.H. Ayala, and Viviana Cocco Mariani. Wind turbines anomaly detection based on power curves and ensemble learning. *IET Renewable Power Generation*, 14(19): 4086–4093, 2020. doi: 10.1049/iet-rpg.2020.0224.

[96] Stanley J. Wenndt and Andrew J. Noga. Narrow-band interference cancellation for enhanced speaker identification. In *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. WASPAA'99 (Cat. No.99TH8452)*, pages 123–126, 1999. doi: 10.1109/ASPAA.1999.810865.

[97] Dingkun Liang, Ning Sun, Yiming Wu, Gendi Liu, and Yongchun Fang. Fuzzy-sliding mode control for humanoid arm robots actuated by pneumatic artificial muscles with unidirectional inputs, saturations, and dead zones. *IEEE Transactions on Industrial Informatics*, 18(5):3011–3021, 2022. doi: 10.1109/TII.2021.3111655.

[98] Lei Ren, Jiabao Dong, Xiaokang Wang, Zihao Meng, Li Zhao, and M. Jamal Deen. A data-driven auto-cnn-lstm prediction model for lithium-ion battery

remaining useful life. *IEEE Transactions on Industrial Informatics*, 17(5): 3478–3487, 2021. doi: 10.1109/TII.2020.3008223.

[99] Kailong Liu, Yunlong Shang, Quan Ouyang, and Widanalage Dhammika Widanage. A data-driven approach with uncertainty quantification for predicting future capacities and remaining useful life of lithium-ion battery. *IEEE Transactions on Industrial Electronics*, 68(4):3170–3180, 2021. doi: 10.1109/TIE.2020.2973876.

[100] Ebru Karakose, Muhsin Tunay Gencoglu, Mehmet Karakose, Ilhan Aydin, and Erhan Akin. A new experimental approach using image processing-based tracking for an efficient fault diagnosis in pantograph–catenary systems. *IEEE Transactions on Industrial Informatics*, 13(2):635–643, 2017. doi: 10.1109/TII.2016.2628042.

[101] Alessio Burrello, Alex Marchioni, Davide Brunelli, Simone Benatti, Mauro Mangia, and Luca Benini. Embedded streaming principal components analysis for network load reduction in structural health monitoring. *IEEE Internet of Things Journal*, 8(6):4433–4447, 2021. doi: 10.1109/JIOT.2020.3027102.

[102] Alex Marchioni, Mauro Mangia, Fabio Pareschi, Riccardo Rovatti, and Gianluca Setti. Subspace energy monitoring for anomaly detection @sensor or @edge. *IEEE Internet of Things Journal*, 7(8):7575–7589, 2020. doi: 10.1109/JIOT.2020.2985912.

[103] Hongzu Li and Pierre Boulanger. A survey of heart anomaly detection using ambulatory electrocardiogram (ecg). *Sensors*, 20(5), 2020. ISSN 1424-8220. doi: 10.3390/s20051461.

[104] Luis Martí, Nayat Sanchez-Pi, José Manuel Molina, and Ana Cristina Bicharra Garcia. Anomaly detection based on sensor data in petroleum industry applications. *Sensors*, 15(2):2774–2797, 2015. ISSN 1424-8220. doi: 10.3390/s150202774.

[105] Ian T. Jolliffe. *Principal Component Analysis*. Springer New York, NY, 2002. ISBN 978-0-387-22440-4. doi: 10.1007/b98835.

[106] Shen Yin, Steven X. Ding, Xiaochen Xie, and Hao Luo. A review on basic data-driven approaches for industrial process monitoring. *IEEE Transactions on Industrial Electronics*, 61(11):6418–6428, 2014. doi: 10.1109/TIE.2014.2301773.

[107] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, 2000. ISSN 0163-5808. doi: 10.1145/335191.335388.

[108] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008*

Eighth IEEE International Conference on Data Mining, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17.

[109] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press.

[110] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis*, pages 4–11, 12 2014. doi: 10.1145/2689746.2689747.

[111] Jie Ding, Vahid Tarokh, and Yuhong Yang. Model selection techniques: An overview. *IEEE Signal Processing Magazine*, 35(6):16–34, 2018. doi: 10.1109/MSP.2018.2867638.

[112] Mononito Goswami, Cristian Ignacio Challu, Laurent Callot, Lenon Minorics, and Andrey Kan. Unsupervised model selection for time series anomaly detection. In *The Eleventh International Conference on Learning Representations*, 2023.

[113] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Grégoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Müller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, 2021. doi: 10.1109/JPROC.2021.3052449.

[114] Sebastian Schmidl, Phillip Wenig, and Thorsten Papenbrock. Anomaly detection in time series: A comprehensive evaluation. *Proc. VLDB Endow.*, 15 (9):1779–1797, may 2022. ISSN 2150-8097. doi: 10.14778/3538598.3538602.

[115] Renjie Wu and Eamonn J. Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2421–2429, 2023. doi: 10.1109/TKDE.2021.3112126.

[116] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

[117] Abhishek B. Sharma, Leana Golubchik, and Ramesh Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sen. Netw.*, 6(3), 2010. ISSN 1550-4859. doi: 10.1145/1754414.1754419.

[118] Kwei-Herng Lai, Daochen Zha, Junjie Xu, Yue Zhao, Guanchu Wang, and Xia Hu. Revisiting time series outlier detection: Definitions and benchmarks. In

Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1), 2021.

[119] Georg Steinbuss and Klemens Böhm. Benchmarking unsupervised outlier detection with realistic synthetic data. *ACM Trans. Knowl. Discov. Data*, 15 (4), apr 2021. ISSN 1556-4681. doi: 10.1145/3441453.

[120] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. ADBench: Anomaly detection benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[121] John Paparrizos, Yuhao Kang, Paul Boniol, Ruey S. Tsay, Themis Palpanas, and Michael J. Franklin. Tsb-uad: An end-to-end benchmark suite for univariate time-series anomaly detection. *Proc. VLDB Endow.*, 15(8):1697–1711, apr 2022. ISSN 2150-8097. doi: 10.14778/3529337.3529354.

[122] Andriy Enttsel, Filippo Martinini, Alex Marchioni, Mauro Mangia, Riccardo Rovatti, and Gianluca Setti. Second-order statistic deviation to model anomalies in the design of unsupervised detectors. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10095287.

[123] Julien Bonnel, Aaron Thode, Dana Wright, and Ross Chapman. Nonlinear time-warping made simple: A step-by-step tutorial on underwater acoustic modal separation with a single hydrophone. *The Journal of the Acoustical Society of America*, 147(3):1897–1926, 2020. doi: 10.1121/10.0000937.

[124] Rajesh K. Sharma and Jon W. Wallace. Improved spectrum sensing by utilizing signal autocorrelation. In *VTC Spring 2009 - IEEE 69th Vehicular Technology Conference*, pages 1–5, 2009. doi: 10.1109/VETECS.2009.5073595.

[125] Michael Unser. Splines: a perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, 1999. doi: 10.1109/79.799930.

[126] Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is $4/\sqrt{3}$. *IEEE Transactions on Information Theory*, 60(8):5040–5053, 2014. doi: 10.1109/TIT.2014.2323359.

[127] Jean Ginibre. Statistical ensembles of complex, quaternion, and real matrices. *Journal of Mathematical Physics*, 6(3):440–449, 1965. doi: 10.1063/1.1704292.

[128] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.10.010.

[129] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation

based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1): 259–268, 1992. ISSN 0167-2789. doi: 10.1016/0167-2789(92)90242-F.

[130] Patrick E. McSharry, Gari D. Clifford, Lionel Tarassenko, and Leonard A. Smith. A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering*, 50(3):289–294, 2003. doi: 10.1109/TBME.2003.808805.

[131] Mauro Mangia, Riccardo Rovatti, and Gianluca Setti. Rakeness in the design of analog-to-information conversion of sparse and localized signals. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 59(5):1001–1014, 2012. doi: 10.1109/TCSI.2012.2191312.

[132] Ary L. Goldberger, Luís A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen C. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 2000. doi: 10.1161/01.cir.101.23.e215.

Finanziato dall'Unione europea- Next Generation EU, Missione 4, Componente 2, Investimento 3.3 (D.M. 352/2022) CUP J33C22001440009.