Faculdade de Engenharia da Universidade do Porto

Mobile System for the Study and Application of Predictive Maintenance in Industrial Equipment

Gonçalo Geada Silveira de Bessa Pacheco

Master's Degree in Mechanical Engineering
Automation

Supervisor: Prof. Dr. António Mendes Lopes

Second Supervisor: Eng. Silvino Machado

July 5, 2026

Mobile System for the Study and Application of Predictive Maintenance in Industrial Equipment

Gonçalo Geada Silveira de Bessa Pacheco

Master's Degree in Mechanical Engineering
Automation

July 5, 2026

Abstract

In the contemporary industrial landscape, maintenance policies represent a crucial factor in operational costs and planning, and so Predictive Maintenance (PdM) has emerged as a highly effective strategy to minimize unplanned downtime and maintenance overhead. This study focused on developing a mobile monitoring system capable of performing machine health diagnostics in industrial equipment.

With this in mind, thorough research was carried out on maintenance practices, data acquisition, and Artificial Intelligence (AI) classification. A literature review was also conducted to evaluate the state-of-the-art implementations of PdM in academic and industrial applications. The information gathered by this research allowed the creation of a structured methodology for the development of this study.

In the development stage, an experimental workbench comprising two inertia-wheel test rigs, one fixed-axis and another equipped with a translational carriage, was devised to evaluate the mobile system's performance across distinct operating states. This system was designed as an edge device to acquire data in real time and run fault detection algorithms on different machinery operational scenarios. To that end, a custom frequency-domain pre-processing pipeline was developed to allow the system to analyze vibration data consistently across different rotational speeds. Additionally, eight distinct AI models, including Machine Learning (ML), Deep Learning (DL), and clustering techniques, were benchmarked in a series of tests of increasing complexity, spanning fault detection, multi-class classification, unseen faults, and cross-domain generalization, to assess the models' applicability and scope.

The results of this experiment show that the devised system is able to perform fault detection across all considered scenarios using real-time edge inference. Furthermore, it was able to detect unknown, untrained faults in both interpolation and extrapolation scenarios. Finally, for individual anomaly classification, it shows solid results in controlled conditions, with some room for further improvement for complex, noisy cross-domain scenarios. Across this trial sequence, a custom-designed Dual-branch Convolutional Neural Network (CNN) and eXtreme Gradient Boosting (XGBoost) achieved the most promising results, proving to be the most consistent algorithms across the range of tested conditions.

Keywords: Predictive Maintenance, AI, Machine Learning, Anomaly Detection, Neural Network, Programmable Logic Controller (PLC), Edge Device

Resumo

No panorama industrial contemporâneo, as políticas de manutenção representam um fator crucial nos custos operacionais e no planeamento, pelo que a Manutenção Preditiva surgiu como uma estratégia altamente eficaz para minimizar paragens não planeadas e os seus respetivos custos. Este estudo focou-se no desenvolvimento de um sistema de monitorização móvel capaz de diagnosticar o estado de equipamentos industriais.

Neste sentido, foi realizada uma investigação aprofundada sobre práticas de manutenção, aquisição de dados e classificação por inteligência artificial. Foi igualmente conduzida uma revisão da literatura para avaliar as implementações do estado da arte da Manutenção Preditiva em aplicações académicas e industriais. A informação reunida nesta investigação permitiu a elaboração de uma metodologia estruturada para o desenvolvimento deste estudo.

Na fase de desenvolvimento, foi concebida uma bancada experimental composta por duas plataformas de ensaio com rodas de inércia: uma com eixo fixo e outra com deslocamento translacional, de modo a avaliar o desempenho do sistema em diferentes estados de operação. Concebido como um dispositivo *edge*, o sistema adquire dados em tempo real e executa algoritmos de deteção de falhas em diferentes cenários de operação. Para tal, foi desenvolvido um pipeline de pré-processamento no domínio da frequência, permitindo que o sistema analise os dados de vibração de forma consistente a diferentes velocidades de rotação. Foram ainda avaliados oito modelos de AI, incluindo técnicas de ML, DL e *clustering*, numa série de ensaios de complexidade crescente, abrangendo deteção de falhas, classificação multiclasse, falhas não treinadas e generalização *cross-domain*, de forma a avaliar a sua aplicabilidade e alcance.

Os resultados demonstram que o sistema desenvolvido realiza deteção de falhas em todos os cenários considerados, com inferência *edge* em tempo real. Foi ainda capaz de detetar falhas desconhecidas e não treinadas, tanto em interpolação como em extrapolação. Na classificação individual de anomalias, apresenta resultados sólidos em condições controladas, com margem para melhorias em cenários *cross-domain* complexos e ruidosos. Nesta sequência de ensaios, os modelos Dual-branch CNN e XGBoost obtiveram os resultados mais promissores, sendo os mais consistentes na gama de condições testadas.

Palavras-chave: Manutenção Preditiva, Inteligência Artificial, Aprendizagem Automática, Deteção de Anomalias, Rede Neuronal, PLC, Dispositivo Edge

Acknowledgements

I would like to acknowledge all of those who were essential to the realization of this dissertation, whether with direct or indirect impact, without whom this project would not be possible.

Firstly, I would like to thank Eng. Silvino Machado for his continuous support and guidance. His knowledge and experience were crucial to this project, and I'm grateful for the opportunity to take part in this research.

I would also like to recognize Prof. Dr. António Mendes Lopes and all the help given in this dissertation. I express my gratitude for sharing ideas and expertise, which were indispensable in improving my work on this project.

To my family, namely my parents and brother, I would like to express my heartfelt appreciation for all the encouragement, love, and affection you have given over the years; without your unconditional support and belief in me, none of this would be possible.

To my friends, a special thanks for all the support, belief, and shared experiences that brought me here.

Last but not least, I express my gratitude to INEGI for hosting my research and giving me the opportunity to collaborate on this project.

Gonçalo Geada Silveira de Bessa Pacheco

AI Declaration

This dissertation occasionally uses AI for grammar correction and phrasing improvement, with all content undergoing extensive review. As AI served strictly as a writing aid, all ideas remain entirely original.

"Not everything that counts can be counted,
and not everything that can be counted counts."

William Bruce Cameron

Contents

List of Figures

List of Tables

Acronyms and Abbreviations list

ADC	Analog-to-Digital Converter
AI	Artificial Intelligence
AIoT	Artificial Intelligence of Things
ARIMA	Auto-regressive Integrated Moving Average
CIM	Computer-Integrated Manufacturing
CNC	Computer Numerical Control
CNN	Convolutional Neural Network
CPS	Cyber-Physical Systems
CSV	Comma-Separated Values
DAQ	Data Acquisition
DHL	Deep Hybrid Learning
DL	Deep Learning
DNN	Deep Neural Network
ETR	Experimental Test Rig
FFT	Fast Fourier Transform
HMI	Human-Machine Interface
HMM	Hidden Markov Models
IEPE	Integrated Electronics Piezo-Electric
IIoT	Industrial Internet of Things
INEGI	Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial
IoT	Internet of Things
IPC	Industrial PC
KNN	K-Nearest Neighbors

LSTM	Long Short-Term Memory
M2M	Machine-to-Machine
MES	Manufacturing Execution Systems
ML	Machine Learning
MMS	Mobile Monitoring System
MQTT	Message Queuing Telemetry Transport
NN	Neural Network
ONNX	Open Neural Network Exchange
PCA	Principal Component Analysis
PdM	Predictive Maintenance
PLC	Programmable Logic Controller
PRISMA	Preferred Reporting Items for Systematic Review and Meta-Analyses
RF	Random Forest
RMS	Root Mean Square
RNN	Recurrent Neural Network
RUL	Remaining Useful Life
SCADA	Supervisory Control and Data Acquisition
SHAP	SHapley Additive exPlanations
SLR	Systematic Literature Review
SQL	Structured Query Language
STFT	Short-Time Fourier Transform
SVM	Support Vector Machines
TFR	Time-Frequency representation
TSDB	Time Series Database
VFD	Variable Frequency Drive
XAI	Explainable AI
XGBoost	eXtreme Gradient Boosting

Chapter 1

Introduction

The present document was composed by Gonçalo Geada Silveira de Bessa Pacheco during the second semester of the 2025/2026 academic year. This is a dissertation work to obtain a Master's degree in Mechanical Engineering - Specialization in Automation, developed at Faculdade de Engenharia da Universidade do Porto, supervised by Prof. Dr. António Mendes Lopes and Eng. Silvino Machado.

1.1 Context

In the modern industrial landscape, enterprises have made an increased commitment to maintenance planning. To optimize revenue, industries aim to minimize breakdowns, increase machinery Remaining Useful Life (RUL), and improve maintenance schedules.

For this purpose, Predictive Maintenance (PdM) has established itself as the state-of-the-art approach for optimized maintenance planning. PdM leverages advances in Artificial Intelligence (AI) models to create autonomous systems that analyze large volumes of sensor data and detect complex patterns with minimal human intervention. These systems allow the detection of faults before they affect the machinery's operation, which would otherwise be impossible with visual inspection or classical statistical models.

To implement this framework, robust hardware and software need to be developed to monitor data in real time and provide immediate diagnostics of emerging problems. PdM integration requires high-resolution vibration sensors operating at high sampling frequencies, paired with devices with sufficient processing power to generate actual machinery diagnostics from the monitored data.

The main limitation of this technology remains domain adaptability. Industrial equipment constantly shifts its operational conditions, making it difficult for AI models to generalize machinery behavior across the varying operating conditions encountered in real-world deployment.

1.2 Objectives

This work's main goal is to devise a Mobile Monitoring System (MMS) capable of executing machinery health diagnosis. This system must be able to monitor industrial equipment in real time as an edge device. In response to the contemporary limitations of PdM, this framework should also be able to detect unknown faults in machinery under shifting operational conditions. The programming languages to be used during development are Python and Structured Text.

1.3 Motivation

With the emergence of AI over the last decade, this thesis's topic has become an extremely relevant theme in the contemporary technological landscape. It works at the intersection of AI and applied engineering, offering the opportunity to simultaneously study academic innovation and industrial applications.

1.4 Structure

The dissertation is divided into five chapters to structure the document in an organized fashion that facilitates reading and understanding.

- **Chapter 1** This chapter contextualizes the main topics and establishes the main objectives of this dissertation. It then presents the motivation behind this project and the document outline.
- **Chapter 2** This chapter presents an overview of the principal concepts concerning PdM, along with context for the most recent state-of-the-art solutions, in the form of a comprehensive literature review.
- **Chapter 3** This chapter introduces the methodology and development of the dissertation. A PdM experimental framework is defined, comprising all the hardware and software used, as well as the trials conducted to validate the proposed system.
- **Chapter 4** This chapter provides the results of the framework proposed in Chapter 3. The model's accuracy and inference time are evaluated in different scenarios to ascertain the system's capability for condition monitoring. The results are then discussed in detail, and the final conclusions are presented in the next chapter.
- **Chapter 5** This chapter provides the closing considerations of this dissertation, comparing the results obtained in Chapter 4 with the objectives delineated in Chapter 1. Furthermore, this thesis's future developments are discussed.

Chapter 2

Literature Review and Preliminary Concepts

2.1 Literature Review Methodology

The Systematic Literature Review (SLR) was performed using the Preferred Reporting Items for Systematic Review and Meta-Analyses (PRISMA) 2020 criteria, in order to achieve comprehensive coverage, clarity of the procedure, and a rigorous analysis of the current state of technological research in PdM.

The literature review was conducted using the Scopus database, which was chosen due to its comprehensive collection of peer-reviewed scientific publications. After an evaluation of the most relevant keywords, a query was devised to limit the results of this search solely to the most pertinent topics, namely predictive maintenance, AI, and industrial applications. Additional search parameters were added to restrict the search to recent and technically relevant studies, as shown in Table 2.1.

Search String (Scopus)

```
( "predictive maintenance" OR "Condition-based maintenance" ) AND ( "AI" OR
"Artificial Intelligence" OR "Machine Learning" ) AND industr*
```

Table 2.1: Systematic Literature Review: Search Parameters and Constraints.

Parameter	**Specification**
Database	Scopus
Search Fields	Title, Abstract, Keywords
Timeframe	2021 – 2026
Document Type	Articles (Peer-reviewed)
Language	English
Subject Areas	Engineering; Computer Science

This query yielded 1050 different records on Scopus. A two-stage screening process was then applied. Firstly, the records were filtered by confining the title content to the relevant topics of this research. This was done using a set of exclusion criteria to consistently eliminate non-pertinent papers. The following title-based exclusion criteria (TC) were applied to the selected articles' titles:

TC1: Domain Irrelevance: Records focusing on non-industrial sectors such as healthcare, civil infrastructure, or out-of-scope engineering fields, such as energy, automotive, etc., that are not relevant to this research.

TC2: Economical Focus: Titles suggesting the articles take a management or economic analysis approach to PdM.

TC3: Generic AI Applications: Articles using AI for general optimization, such as logistics or energy consumption, that did not address anomaly detection implementations.

After the first stage, of the original 1050 papers, 153 were chosen that complied with the selected criteria. While the title-level screening served to eliminate broad domain irrelevance, the abstract-level filtering focused on technical rigor. At this phase, the abstracts of the remaining records were evaluated against specific Inclusion Criteria (IC) and Exclusion Criteria (EC) to ensure the final selection of 45 articles provided comprehensive validation on industrial applications, AI model architecture benchmarking, and Data Acquisition (DAQ). The criteria were applied as stated in Table 2.2.

Table 2.2: Inclusion and Exclusion Criteria for Abstract and Full-text Screening.

Category	**Inclusion Criteria (IC)**	**Exclusion Criteria (EC)**
Technical Depth	Proposes or implements a specific DL/ ML architecture (e.g., CNN, LSTM, KNN).	Conceptual papers or surveys without original AI modeling implementation or relying on logic-based traditional algorithms.
Validation	Uses experimental data or high-fidelity simulated datasets.	Studies with insufficient data results or a lack of performance metrics.
Context	Focused on component-level health (e.g., bearings, motors) or industrial processes.	Applications in non-industrial sectors or irrelevant industries.
Scope	Includes details on data acquisition, data storage, or signal processing.	Focused on industry efficiency or revenue maximization without PdM as the main focus.

After the bibliographic analysis of the 45 selected articles, the context of the researched papers was evaluated to ensure this literature review is globally relevant in the contemporary technological landscape. This verification shows a peak in this topic in 2025, revealing a growing trend in recent years, as shown in Figure 2.1. In addition, the global distribution of these articles shows that the implementation of PdM strategies has reached a worldwide scale. Furthermore, Europe and Asia, namely India and China, have been at the forefront of research, accounting for over 75% of international reviews on PdM, among the selected articles, as shown in Figure 2.2.

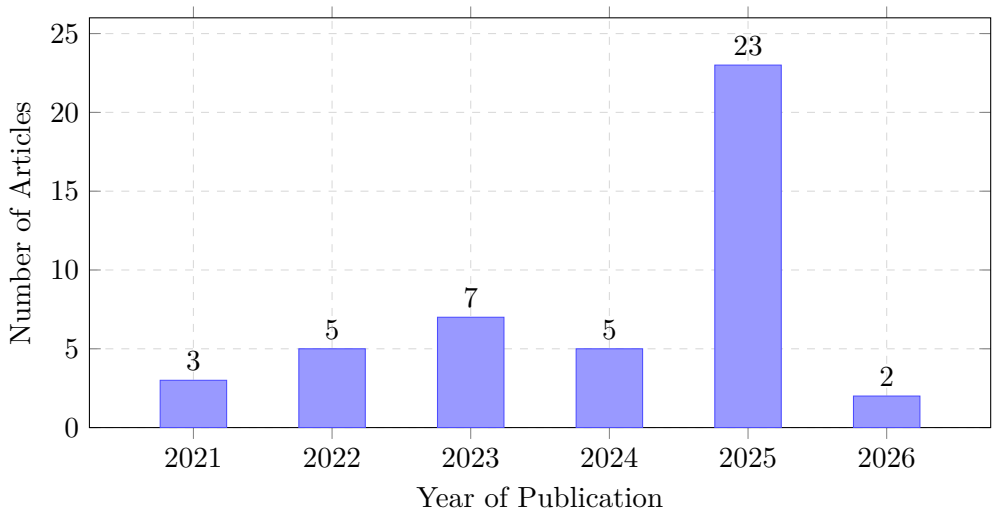


Figure 2.1: Temporal distribution of the 45 selected articles (2021–02/2026).

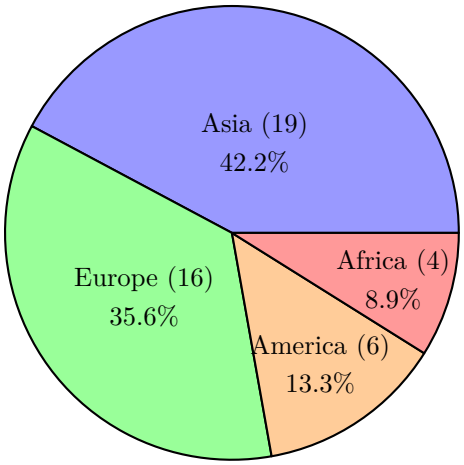


Figure 2.2: Geographical distribution of selected articles by global region ($n = 45$).

Following a systematic analysis of the selected literature, Table 2.3 outlines the core methodologies and primary limitations of each study. This overview highlights the boundaries of the current frameworks, thereby establishing the research gap addressed by this thesis.

Table 2.3: Summary of Reviewed Literature.

Article	**Description**	**Main Limitations**
[1]	Collects multi-sensor data from AC motors under various fault conditions to train a DNN on a cloud server for fault classification.	Uses a microcontroller as an edge device for data acquisition and only tests the algorithm against known anomalies.
[2]	Uses LSTM autoencoders to estimate RUL of industrial equipment based on real-world manufacturing data, evaluated specifically in a steel industry use case.	Lacks pre-processing methodology and focuses only on LSTM algorithms.
[3]	Presents a scalable PdM system architecture utilizing AutoML for data-driven fault classification, validated on an autonomous transfer vehicle and an electric motor.	Does not provide evaluation metrics of the proposed method.
[4]	Evaluates six machine learning algorithms for predicting machine conditions using vibration, temperature, and machine status data.	Does not identify the root cause of the anomalous state; only indicates whether the machine needs maintenance.
[5]	Combines gradient boosting for failure prediction and SHAP values for explainability to generate optimal maintenance schedules that minimize downtime.	The proposed method relies on manual feature engineering and is computationally heavy.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[6]	Develops an adaptive framework utilizing online LSTM and KNN models to process continuous data streams and dynamically adjust motorized system speeds to manage faults in real time.	This model can cause computational overhead as the KNN dataset grows and is more susceptible to error propagation due to its pipeline architecture.
[7]	Tests various machine learning, deep learning, and hybrid models on a synthetic dataset, highlighting Deep Forest and Gradient Boosting algorithms for superior early defect detection.	Uses a synthetic dataset and lacks algorithm parameter optimization; as such, the results of the benchmarked models are underwhelming.
[8]	Validates a hybrid CNN-LSTM framework across 18 months of large-scale industrial sensor data, proving significant economic benefits and high accuracy in predicting failures 48 hours in advance.	The model is implemented on a very complex multi-sensor dataset and focuses on long-term prognosis rather than instant diagnostics.
[9]	Investigates the performance of multiple machine learning models for fault detection in the commercial refrigeration systems of a brewing company.	Uses highly specific multi-sensor data only applicable to this particular system.
[10]	Highlights the critical impact of data preparation techniques and proposes a mixed-integer linear programming method to prevent deterioration in KNN predictive accuracy during gas turbine decay.	The model is very sensitive to preprocessing; slight variations in the preparation steps can cause a severe impact on the model's efficiency.

Continued on next page

Table 2.3 continued from previous page

Article	**Description**	**Main Limitations**
[11]	Classifies 249 publications to provide a systematic overview of PdM in Industry 4.0, identifying core challenges like the lack of relevant benchmark datasets and system complexity.	Lacks hardware validation.
[12]	Integrates IoT sensors, the MQTT protocol, and machine learning (specifically Random Forest) to analyze operational data for anomaly detection and proactive maintenance of electrical motors.	Uses a microcontroller as an edge device for data acquisition. The proposed method lacks scalability for industrial applications and relies on manual feature extraction.
[13]	Leverages Digital Twin Technology and reinforcement learning models to simulate operational behaviors, detect anomalies, and optimize maintenance schedules in smart manufacturing.	Uses multiple Machine Learning (ML) and Deep Learning (DL) approaches simultaneously, causing considerable computational overhead and overlapping diagnostics.
[14]	Combines Random Forest for fault classification and LSTM networks for estimating remaining useful life in an AI application tailored specifically for optimizing CNC machine maintenance.	Does not specify which sensors are being used, nor what type of variables are being monitored.
[15]	Proposes a hybrid machine-learning ensemble utilizing Local Outlier Factor, One-Class SVM, and Autoencoders to improve real-time anomaly detection, successfully tested on air-blowing machines.	Focuses on anomaly detection over fault diagnosis. Relies on complex hyperparameter tuning.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[16]	Investigates the integration of IIoT and four distinct machine learning models to optimize PdM procedures and reduce industrial downtime.	The compared models achieve below-average accuracy results.
[17]	Introduces a "Triplet Fault Injection Algorithm" for rigorous testing and employs the XGBoost algorithm for superior early fault detection in IoT-enabled equipment.	Uses data from a highly complex multi-sensor dataset. Uses a micro-controller as an edge device for data processing.
[18]	Proposes an advanced IoT-driven system utilizing federated learning and XAI to securely and transparently predict failures in heavy vehicle operations.	Focuses on federated learning, gathering data from different factories, which makes it susceptible to communication overhead and accuracy trade-offs in relation to centralized models.
[19]	A systematic literature review of 44 studies focusing specifically on the practical application of machine learning for mechanical fault diagnosis in real-world manufacturing.	Lacks hardware validation. Uses inconsistent metrics across different studies.
[20]	Provides a comprehensive survey of PdM planning models and machine learning techniques, detailing stages from data preparation to decision modeling.	Only provides a theoretical approach and concludes that there is no concrete pipeline for PdM, as different pre-processing and decision-making models can lead to different outcomes in distinct scenarios.
[21]	Focuses on optimizing feature selection in wind turbine blades to filter out redundant condition indicators, enabling simpler machine learning models to perform accurately.	The devised method only affects ML models and does not account for variations in the baseline, making the originally selected features outdated in the long term.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[22]	Develops and evaluates a hybrid model combining LSTM and SVM to enhance the accuracy of machine failure prediction.	Only tests the data against known anomalies. This hybrid model has decreased interpretability and does not explain the pre-processing steps used to obtain its results.
[23]	Compares Random Forest and LSTM models for predicting motor conditions in a raw mill.	The two chosen models differ significantly in efficiency, indicating that the model's performance relies heavily on manual feature extraction.
[24]	Proposes an adaptive, online machine learning pipeline (cycle extraction, feature learning, health detection) for continuous PdM, validated in the railway industry.	The model is trained for a specific working condition of speed and load, and might need retraining when new conditions arise.
[25]	Investigates the combination of highly sensitive Fiber Bragg Grating vibration sensors with machine learning models to detect anomalies in rotating machinery.	The model relies on image-based conversion to detect exclusively predefined fault classes in a controlled dataset.
[26]	Optimizes bearing fault diagnosis by transforming 1D vibration signals into 2D scalograms and applying transfer learning via a pre-trained SqueezeNet neural network.	The data acquisition depends on high-frequency sensors placed in the gearbox housing within a vibration-isolated environment.
[27]	Uses IoT sensor networks to predict whether electric motor temperature or vibration levels will exceed predefined thresholds within a ten-minute window.	The tested models reveal that optimized algorithms for a specific working condition get significantly lower diagnostic precision in alternative conditions.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[28]	Introduces a Brokenstick Regression-based multi-objective dragonfly predictive optimization algorithm to improve fault prediction accuracy and reduce cloud execution time.	This study uses cloud-based processing paired with complex processing models that can struggle in real-time monitoring applications.
[29]	Combines advanced vibration analysis techniques, like wavelet transforms, with a suite of machine learning models to detect hidden faults in noisy rotating machinery environments.	Relies on frequency-domain analysis and can be susceptible to not detecting transient mechanical events.
[30]	Proposes a hybrid CNN-LSTM framework for multivariate time-series forecasting, leveraging CNNs for feature extraction and LSTMs for long-term dependency patterns.	Uses a complex model whose objective is achieving maximum accuracy with no real-time deployment constraints.
[31]	Presents a distributed digital twin architecture utilizing fog computing to enable real-time condition monitoring and predictive analytics, validated on a wind turbine case study.	The proposed ML models depend on the selected parameters, yielding different results for each class, without a solution that optimizes overall efficiency.
[32]	A scoping review of the past five years that provides practical guidelines on data processing, information sources, and algorithm selection for applying ML and DL in PdM.	Reveals difficulties in comparing results from different studies that use different datasets and evaluation metrics.
[33]	Combines an ensemble Gradient Boosting classifier with SHAP to create a highly accurate and interpretable fault diagnosis framework for industrial gear systems.	Assumes that frequency content does not change in each measurement window, potentially leading to significant diagnosis loss in transient features.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[34]	Examines the role of data-driven AI models in Industry 4.0 PdM, emphasizing the need for multi-criteria evaluation to select suitable prognostic models.	Absence of empirical industrial validation.
[35]	Implements a digital twin architecture utilizing an unsupervised deep learning hybrid model (Autoencoder and LSTM) for anomaly detection on a monofilament winding machine.	Uses feature extraction and anomaly thresholds specific to winding machines and does not give information about the model's latency or inference speed.
[36]	Utilizes RNNs to extract features from electrical current signatures, enabling near real-time wear prediction at the edge for production plants.	The use of STFTs for pre-processing can lead to computational overhead. The model lacks interpretability.
[37]	Compares SVM, KNN, and NN models for diagnosing bearing defects under varying loads and speeds.	The model's objective is anomaly detection, without a focus on fault diagnosis.
[38]	Applies multi-label classification using Random Forest on multi-modal sensor data (vibration, temperature, ultrasound) to simultaneously detect multiple faults in industrial suction fans.	The model relies on supervised learning limited to trained classes, losing efficiency in the presence of noisy signals and unseen anomalies.
[39]	Proposes a framework for a large-scale, annotated universal vibration analysis dataset (starting with bearing data) to facilitate efficient transfer learning in PdM.	This study focuses exclusively on bearing data, using optimized spectrograms that might not correlate with other fault states in machinery.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[40]	Evaluates over 20 different ML, DL, and deep hybrid learning algorithms on a synthetic dataset, demonstrating that Deep Forest and Gradient Boosting achieve the highest average accuracy.	The models are tested using synthetic data, and, as such, lack validation and interpretability in empirical industrial data.
[41]	Enhances an ML-based fault detection approach (Random Forest and K-means) with XAI tools to improve the interpretability and root-cause diagnosis of industrial assets.	The study focuses on diagnostic explainability with minimal regard for real-time diagnostic implementation.
[42]	A PRISMA-compliant systematic review of 34 articles that explores the integration of PdM and Digital Twin technologies across dynamic fields like healthcare, utilities, and agriculture.	Focuses on a theoretical framework without taking into account operational restrictions and industrial implementation.
[43]	Benchmarks twelve ML and DL models across two manufacturing datasets, finding that Random Forest provides the most stable performance and effectively handles imbalanced sensor data.	Uses downsampling pre-processing techniques that might not correlate to other datasets, since it uses datasets without vibration data.
[44]	Proposes "T-PdM", a novel framework that uniquely integrates regression, classification, and association rule mining into a single platform.	This study introduces a model with high computational overhead, both in pre-processing and machine learning, unsuited for real-time fast cycles in industrial monitoring.

Continued on next page

Table 2.3 continued from previous page

Article	Description	Main Limitations
[45]	Investigates the Teager-Kaiser Energy Operator for advanced signal pre-processing, proving it enhances the accuracy of bearing fault classification models by capturing dynamic variations.	The model is optimized to detect bursts of energy and is highly sensitive to noisy data. This model is unsuited for cyclical contextual anomalies.

Alongside the articles chosen with the PRISMA method, 29 other references were selected from cross-reference articles, image search, standards, and textbooks. Figure 2.3 shows a flow diagram of the entire PRISMA selection criteria and results.

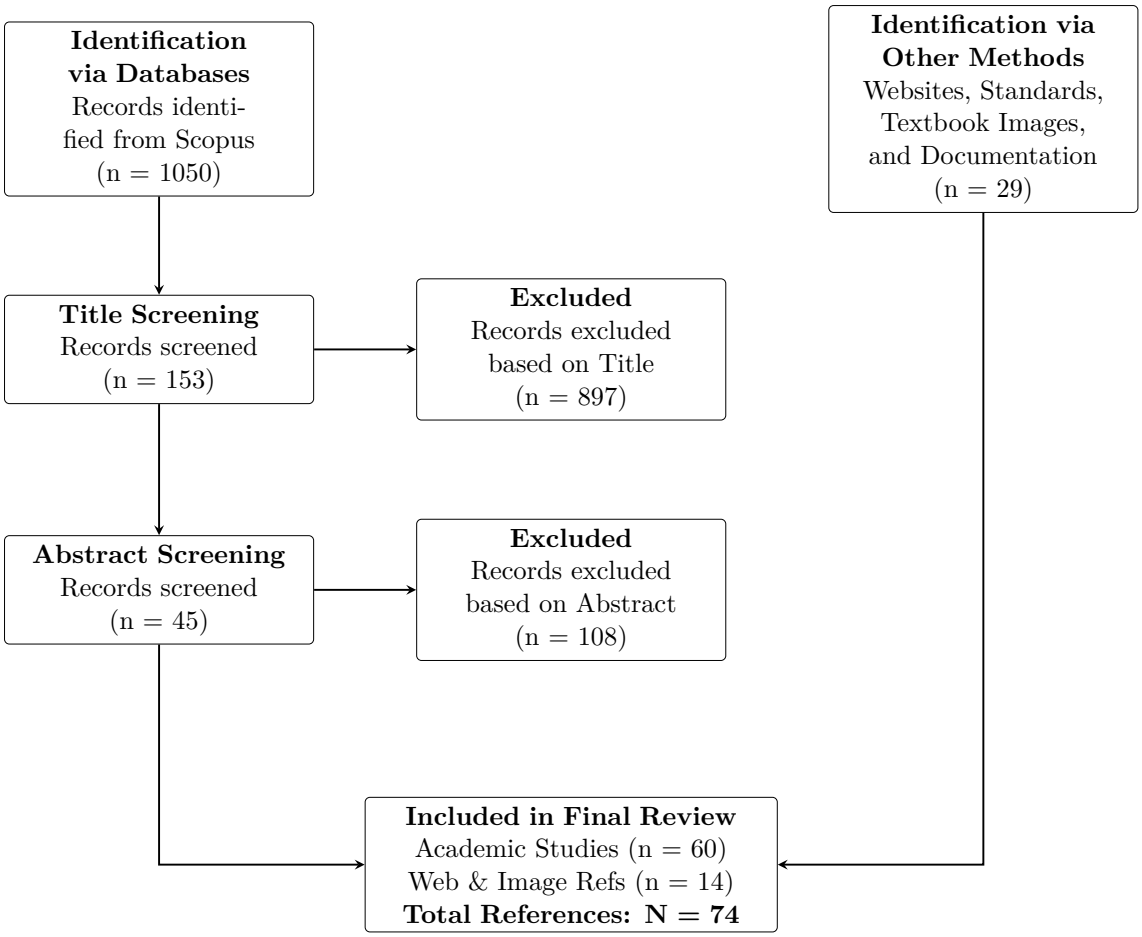


Figure 2.3: PRISMA flow diagram.

Selected articles referenced in this Literature Review are cited in detail in the following sections to contextualize and elaborate on the core concepts of PdM.

2.2 Industrial Automation

Industrial automation refers to the implementation of control systems, such as programmable controllers, robotics, and machine vision, to manage industrial processes with minimal human intervention. Historically, its primary objectives have been to reduce operational costs, increase productivity, and improve product quality, while, at the same time, removing human operators from hazardous or repetitive tasks [46]. In this sense, automation has solidified itself as a core principle of modern manufacturing.

The origins of industrial automation can be traced to the introduction of machinery into production during the Industrial Revolution of the late 18th century, which introduced steam-powered equipment and the organized division of labor. Drawing a comparison to the modern era of automation, the paradigm shift toward prioritizing flexibility in manufacturing environments led to the introduction of the PLC in the late 1960s, which replaced hardwired relay-based control logic with software-configurable systems [47].

This trend evolved into the adoption of Supervisory Control and Data Acquisition (SCADA) systems in the 1970s onwards, which enabled considerable advances in industrial processes by implementing real-time centralized monitoring of entire industrial environments. The integration of these technologies into the framework of Computer-Integrated Manufacturing (CIM) architectures through the end of the 20th century materialized the beginning of the path towards modern industrial digitalization.

From an architectural perspective, industrial automation systems are typically structured into hierarchical functional layers. This framework was devised by the International Standard IEC 62264 (Enterprise-Control System Integration), which categorized the different industrial levels, from Level 0, regarding field-level physical instrumentation, to Level 4, which concerns planning and logistics, as shown in Figure 2.4. At the lower levels, sensors and actuators interface directly with physical processes, feeding data into consecutive higher levels through PLC and SCADA systems to Manufacturing Execution Systems (MES) and, ultimately, to enterprise management software. This multi-tier model established a common automation basis that enables the use of an analogous system across industries [48].

The usage within industrial environments of digital networking, generally referred to as the IIoT, has transformed the automation paradigm over the past two decades. With the integration of low-cost sensors, edge computing, and industrial communication protocols such as OPC-UA and MQTT, it is possible to collect and communicate considerable volumes of operational data between floor-level equipment and monitoring devices. The automation of this process enables uninterrupted monitoring, opening the way to novel approaches to maintenance strategies in which equipment health is inferred from real-time data patterns rather than fixed inspection intervals. The implications of this transition for maintenance practice, as well as the general concepts of this shifting industrial paradigm, are addressed in the subsections that follow.

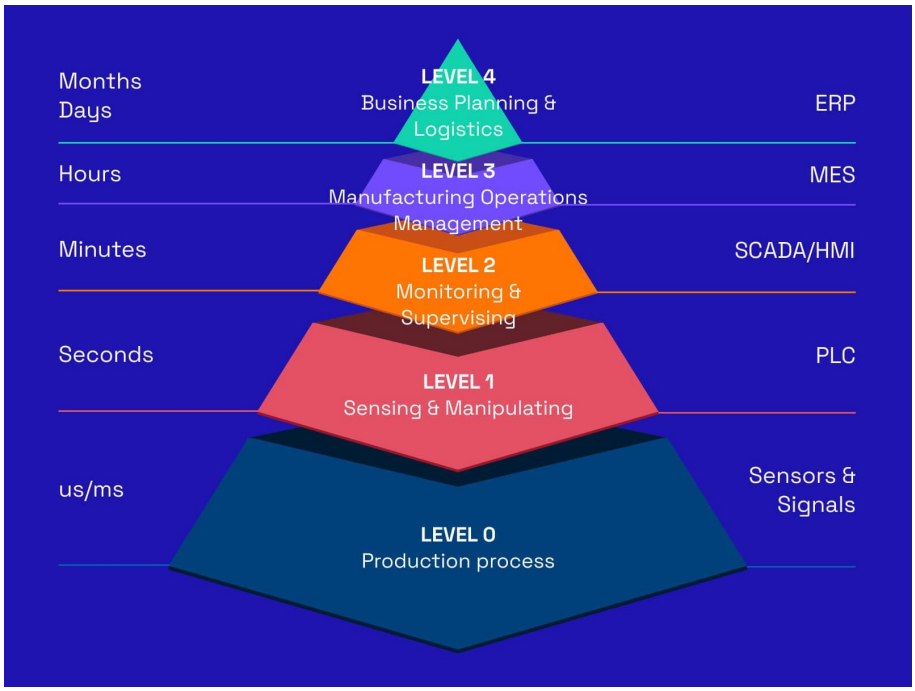


Figure 2.4: The IEC 62264 Industrial Automation Hierarchy [49].

2.2.1 Industry 4.0 and 5.0

The fourth industrial revolution, commonly referred to as Industry 4.0, is defined by unprecedented connectivity and data-driven decision-making at the industrial level, and it is materialized by the incorporation of sensors for real-time monitoring and AI-controlled logic. Built on the concept of Industry 3.0, which was characterized by embedding automation and electronics in industrial manufacturing, Industry 4.0 took a step forward, integrating several technologies, including Cyber-Physical Systems (CPS), the IoT, and big data analytics, leading to a 10–30% decrease in manufacturing and planning costs [6].

The latest industrial paradigm, Industry 5.0, focuses on integrating industrial technologies with human expertise. The European Commission's Industry 5.0 initiative emphasizes the collaborative relationship between human decision-making and intelligent automated systems, namely Artificial Intelligence of Things (AIoT)-enabled platforms, cobots, and AI-driven manufacturing [50]. Aligned with these principles, sustainability and flexibility are the core elements of the modern industrial paradigm. Enterprises keep lean logistics as a focal area of attention, reducing their environmental impact and enabling adaptation and recovery to changes, whether economic, environmental, or technological [25].

2.2.2 Internet of Things

The IoT is a network of physical devices embedded with sensors and data communication technology, enabling data to be collected and transferred to other devices and the network. This technology also enables autonomous remote control of these devices over the network, based on the data feed information [16].

IoT devices, also known as smart sensors, are found across multiple industries, ranging from smart home devices to advanced industrial machinery. The ability to communicate between different devices, and by extension, multiple sensors, creates an opportunity to improve the efficiency of various tasks, including monitoring system conditions, controlling machines using data input from other devices, or even tracking inventory and facilitating shipping [51].

The inclusion of AI in the decision process of these systems has proven to be a decisive factor in the growing relevance of this technology. This development, called AIoT, reflects the necessity for real-time autonomous processing in order to handle the increasing growth in data volume and complexity collected by interconnected systems [52]. AIoT systems have the capacity to process data, identify patterns, and make decisions despite the usage of resource-limited IoT devices in the network, making these systems extremely useful for both commercial use in smart devices in a cost-controlled manner, and as industrial complex solutions, as shown in Figure 2.5.

The inclusion of this technology in industrial machinery takes it one step further. With the integration of big data, edge computing, and machine learning, these systems excel at making rapid decisions by transforming large volumes of raw data into valuable information, enabling a new level of flexibility across industries [16]. IIoT systems are especially useful for remote monitoring, resource allocation, energy management, and the implementation of Digital Twins [8, 51].

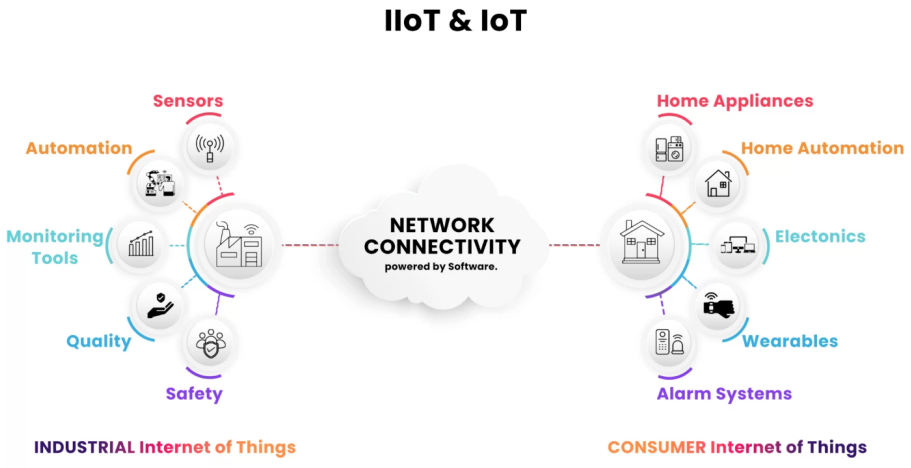


Figure 2.5: Comparison between IoT and IIoT [53].

2.2.3 Digital Twins

A Digital Twin is a virtual representation of a physical asset that uses real-time data from IoT sensors to monitor and optimize the performance of target systems [13]. It functions as an identical, live digital copy of the system and, as such, can simulate multiple scenarios using the physical machine data to monitor the device's state, predict its future behavior, and support operations with decision support. As a consequence of its applications, systems can reduce maintenance costs, improve production efficiency, refine systems' consistency and reliability, and simulate operation under diverse adverse conditions to develop prevention plans [31, 35]. Figure 2.6 illustrates the framework of Digital Twins.

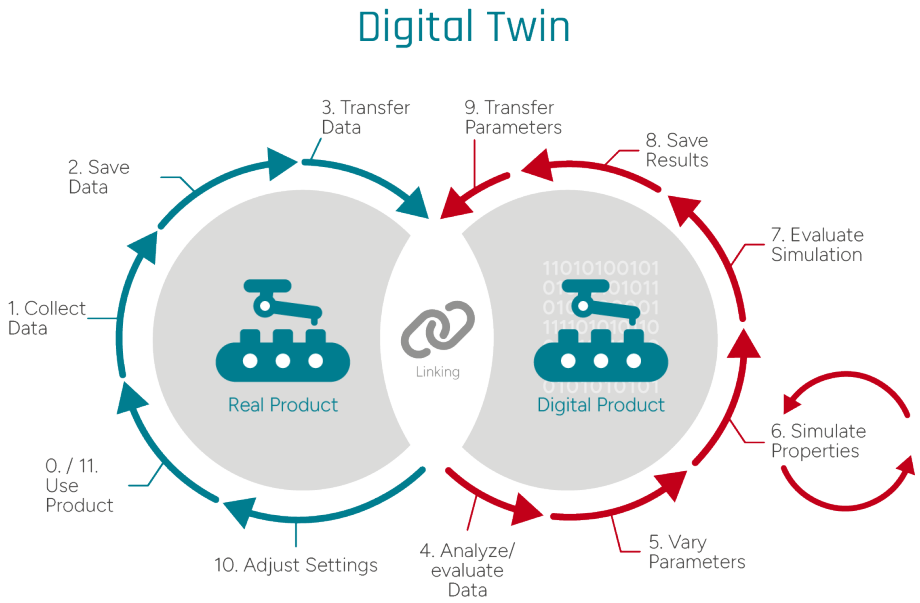


Figure 2.6: Digital Twin concept [54].

2.3 Industrial Maintenance

Present-day industries rely on effective maintenance to reduce downtime and maximize revenue. Several maintenance strategies have been used over the years, which can typically be divided into three categories: Corrective Maintenance, Preventive Maintenance, and Predictive Maintenance [7]. Figure 2.7 represents the basic function of each method.

Corrective Maintenance, also known as Reactive Maintenance, occurs after a failure, malfunction, or damage that diminishes a machine's working condition. This approach follows a run-to-failure management, which means after a breakdown, the machine can be repaired, but there is always downtime due to the unexpected stoppage [11]. This method incurs the lowest upfront costs but can lead to several halts in the production process, especially in modern industries, where some machinery may be irreparable and thus result in a much larger long-term financial impact [40].

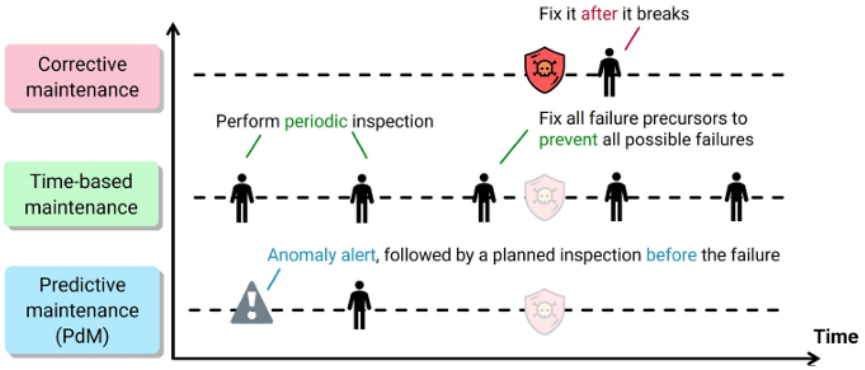


Figure 2.7: Maintenance strategies [24].

On the other hand, Preventive Maintenance consists of pre-scheduled maintenance activities based on the manufacturer's estimation of equipment lifetime. This method can also be complemented by visual inspection of floor-level operators, also known as Condition-Based Maintenance, which can schedule extraordinary maintenance based on an assessment of equipment status. Even though this method incurs additional costs for these scheduled tasks, the risk of fatal failure is greatly reduced. However, unexpected errors can still be undetected, causing severe constraints to the production plant [1]. In order to minimize downtime as well as reduce maintenance activities to the utmost minimum, PdM is emerging as an industrial trend.

2.3.1 Predictive Maintenance

PdM is considered a proactive maintenance strategy aimed at anticipating equipment failure using IoT sensors. Analyzing this data, while also having access to historical data, enables estimation of failure risk and RUL [5]. Traditional approaches rely mostly on statistical analysis, but with the inclusion of ML and IoT devices, the accuracy of these models has vastly increased to detect future equipment failure, leading to optimized maintenance schedules, extension of machinery RUL, and minimization of operational costs [4, 19]. Using PdM, predictive correlations are converted into actionable maintenance steps, thereby improving the safety and efficiency of maintenance tasks for floor-level operators [6].

2.3.1.1 Condition Monitoring

Condition monitoring is the fundamental basis for PdM. The implementation of sensors that continuously monitor machinery operation allows the establishment of a baseline for the normal behavior of its main physical components. The comparison between the live data feed and the normal baseline enables the detection of faults, defects, and overall anomalous behavior.

In order to infer the equipment's health, there are several variables that can be monitored:

- Vibration: The analysis of vibration frequency is a critical indicator to assess machine components' condition, particularly in rotating machinery. Each component has a signature frequency response, and new patterns might indicate misalignment, wear, or imbalance in gears, shafts, bearings, or other loose parts [6].
- Temperature: Excessive heat can reduce components' lifespan, as it represents a common symptom of motor overloading, bearing wear, and lubricant issues. Continuously monitoring this variable leads to fewer unexpected system failures and safety hazards for the operators [55].
- Electric current: Electrical properties are a reflection of the motor's state. Sudden current peaks are an effect of motor overload, insulation breakdown, phase imbalance, or mechanical binding [56].
- Oil analysis: Lubricant condition provides three categories of information: lubricant characteristics (fluid viscosity and oxidation); external contamination (water, dirt, etc.); and wear debris (mechanical parts wear) [57].

2.3.1.2 Anomalies

Anomalies manifest as deviations in the system state from the machine's normal baseline behavior, which is created by taking into account the machine components' historical data as well as the system monitoring sensor data [24].

Anomalies can be classified into three main categories, as established by Chandola et al. [58]:

- Point anomalies: A single data point deviates from the entire dataset (e.g., an electric current spike).
- Contextual anomalies: The data is considered anomalous only in a specific scenario. In this case, there are no outliers in the data; it is the context of the data that creates the anomaly (e.g., a rise in the motor's current is not an anomaly if the motor is equipped with a Variable Frequency Drive (VFD) that is controlling the speed; otherwise, that rise without any change in the output speed is considered a contextual anomaly).

- Collective anomalies: The collection of related data appears normal individually, but their sequence is abnormal in comparison to the total data (e.g., a slight rise in the current values of the motor, which are within the normal values, but these changes occur periodically, cyclically, and simultaneously with the engagement of a damaged gear tooth).

The combination of multiple monitored operating parameters could also help identify the concrete anomaly. For example, when analyzing the electric current and temperature of a motor, any variable change would be considered an anomaly, but the correlation between the parameters could facilitate identifying the cause of the faulty state [15, 56]. In this case, detecting each factor, individually and in combination, provides a diagnostic framework for identifying specific failure modes, as shown in Table 2.4.

Table 2.4: Condition Monitoring: Thermal and Electrical Symptom Analysis.

Symptom	**Primary Root Causes**
High Current	Overloading, Power Quality Issues, Broken Rotor Bars, Starting Issues
Overheating	Lubrication Failure, Cooling System Failure, High Ambient Temperature
Current + Heat	Mechanical Binding, Insulation Breakdown, Phase Unbalance

2.4 Data Acquisition

Data acquisition is the first step in machinery prognosis and comprises the process of acquiring and storing measurement signals obtained through monitoring sensors. In order to analyze machinery's physical characteristics, sensors are used to convert these variables into electrical signals that can be perceived by electronic devices, such as computers or PLCs [59]. That conversion is accomplished by the use of an Analog-to-Digital Converter (ADC), an electronic circuit that produces a digital output proportional to the analog input. The two main characteristics of an ADC are the resolution and conversion rate. The resolution determines how closely the analog signal is represented by the digital number. The conversion rate characterizes how quickly the ADC converts the analog value into a digital one. Depending on the context and application of the system, correctly defining these parameters on an ADC is essential for creating an accurate digital representation of the detected analog signal [60].

Another important concept to take into account is sampling. A sample is taken periodically when converting an analog signal to digital at a fixed rate, generally referred to as the sampling frequency, as shown in Figure 2.8.

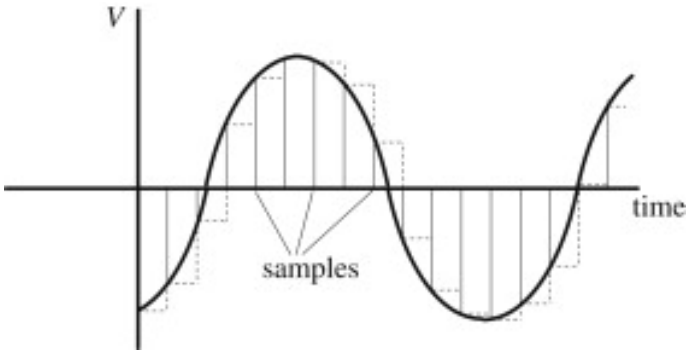


Figure 2.8: Sampling frequency [60].

For the signal to be correctly represented, the sampling frequency needs to be calculated based on the maximum frequency of the analog signal being converted, which, according to the Nyquist theorem, must be at least twice the maximum frequency.

2.4.1 Data Acquisition Systems

In order to correctly acquire data in an industrial setting, a cohesive hardware chain system is implemented to not only guarantee the correct conversion in the ADC, but also to ensure the correct time alignment between different sensor data, as well as the implementation of the correct sensorization for each analog target signal [18, 60].

A Data Acquisition System comprises 5 main components: sensors/transducers, signal conditioning, ADCs, communication, and storage. Firstly, the correct sensors must be chosen based on both the application and the dynamic response and frequency range of the monitored signal and its possible failure modes. The generated raw signal undergoes isolation, denoising, and filtering to produce a 'clean signal' ready for processing by the ADC, as previously discussed. The acquired data is then stored via communication protocols. There are multiple solutions regarding data transfer and storage, which will be discussed in the following sections.

To guarantee the success of this process as well as its efficiency, there are some important concepts to consider, namely real-time synchronicity and buffered packaging. For the purpose of analyzing the captured data, all sensor data must have the correct temporal alignment, so that if something is detected by a particular sensor, the data from other sensors is perfectly aligned and ready to be analyzed accordingly. In this case, even a millisecond difference between the different sensors' data can be interpreted as a different phase and compromise the analysis with erroneous results.

Furthermore, computer efficiency is another important parameter to take into account. A data acquisition system with multiple sensors can generate substantial amounts of data, easily creating bottlenecks in the pipeline. Typical industrial controllers have task cycle times of 1 ms, enabling them to execute fast real-time operations effectively while balancing the system's performance and computational overhead. However, industrial sensors

operate with a frequency response up to 10 kHz, which, under the Nyquist criterion, would require a minimum sampling frequency of 20 kHz, equivalent to a 50 μs sampling period, an interval that is significantly faster than the previously stated cycle time of 1 ms. To prevent processor overloading during such a short task time, a technique called oversampling can be implemented.

With this methodology, even though the terminals run at a faster 50 μs rate, the controller still runs at 1 ms, and the samples are packaged into an array only after 1 ms of data is aggregated. This allows a much faster sample acquisition rate, while still maintaining the controller at a typical task cycle time [61]. Figure 2.9 exemplifies a simplified version of a data acquisition system.

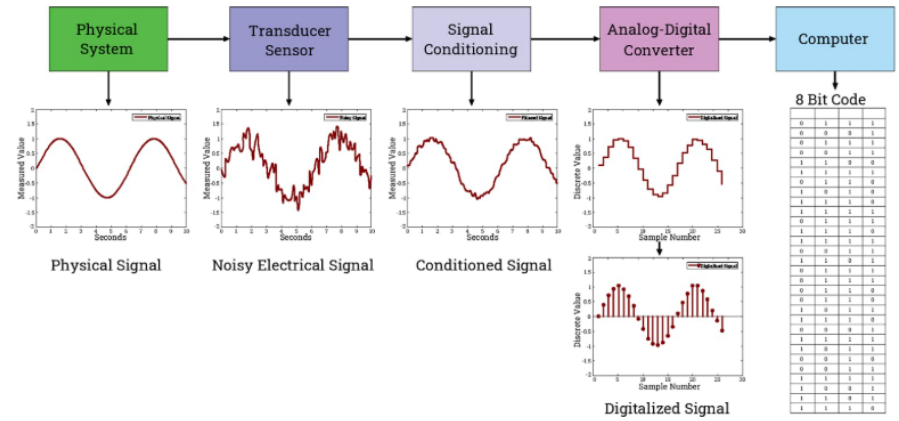


Figure 2.9: Data Acquisition System [62].

2.4.2 Communication Protocols

Communication protocols enable industrial devices to exchange data reliably. These standardized protocols are implemented at different levels of the system's architecture, as shown in Figure 2.4, and, depending on the characteristics and requirements of each level, different communication protocols are used.

At the lowest level of a data acquisition hardware chain, the protocol must ensure that data is transmitted from the ADC to the PLC with minimal latency and jitter. For this purpose, deterministic fieldbus protocols, such as EtherCAT, Profinet IRT, and others, are implemented to ensure that data is sent not only synchronously but also at the exact same time each cycle. These protocols also reduce the number of needed cables between field devices and provide device health information and process data.

At a higher level, with the aim of facilitating communication between the PLC and the monitoring and manufacturing levels, the industry-standard protocol is OPC UA. This protocol offers a universal interface and encryption-level security, being ideal for IIoT

and Machine-to-Machine (M2M) communication and enabling data transmission between PLCs and SCADA, MES, and local monitoring [31].

For lightweight communication for AIoT and storage applications, MQTT is the preferred protocol. This protocol uses a Publish-Subscribe model mediated by a broker, which then distributes the data to subscribers. This is particularly effective for storage solutions, since the protocol provides low-bandwidth, low-latency operation with minimal resource requirements [12].

2.4.3 Data Storage

The main principle of a data storage system is to ensure the database's structural integrity and organization. For PdM applications, the system must be able to receive large bursts of data via MQTT, prevent data loss during network failures or power outages, and maintain long-term historical records for analysis or future needs.

There are 3 main architecture solutions for data storage management: edge, cloud, and hybrid approaches. Edge storage consists of using PLCs, Industrial PCs (IPCs), or local servers in order to store the data locally, enabling fast, high-frequency raw data transfers that are too large to send over the internet. Cloud storage solutions leverage global storage to enable general data access and off-site, computationally demanding Big Data analysis. Hybrid approaches are common in PdM and combine edge and cloud storage, storing data locally for a short time after acquisition, enabling real-time monitoring and other applications, and at the same time using cloud storage for long-term storage and AI model training [63].

Another important aspect to take into consideration is how the data is saved and organized. For small datasets, simple solutions such as using CSV files can be effective, even though this strategy is semi-structured and lacks indexing. For more complex and organized datasets, one common solution is the implementation of SQL databases, which create relational databases organized into tables with columns and rows, where relationships are established between stored values. Another option is to employ Time Series Databases (TSDBs). This solution has the advantage of being designed for time-series and time-stamped data, enabling features that are helpful for PdM, such as high-volume ingestion, data downsampling, and compression [64].

2.5 Data Processing

While the previous section focused on the acquisition and storage of the physical variables as digital data, this chapter's main objective is to define the methodology to convert high-dimensional, noisy data obtained through monitoring sensors into an actionable evaluation of machinery status.

The processing phase comprises three main stages:

- Data pre-processing
- Anomaly detection
- Evaluation and interpretation

2.5.1 Data Pre-processing

Data pre-processing can be divided into two phases: data preparation and feature extraction. The first step in this method is data cleaning. In this phase, the main objective is to remove noise and outliers as well as replace missing values.

Digital filters, such as low-pass, high-pass, and band-pass filters, are applied in order to remove electrical noise and vibration from adjacent machinery, isolating a more faithful digital representation of the monitored data. The following operation removes outliers, which in analytics represent data points that significantly deviate from the other values in the dataset. Techniques such as the Hampel filter are used to remove these outliers without altering the signal shape. Addressing missing values in the dataset is crucial to guarantee data continuity. Using interpolation techniques, such as spline and polynomial algorithms, fills the gaps in an incomplete dataset, turning it into a reliable database for further processing [23].

The final process within data preparation to take into consideration is normalization. This technique scales the data into a standard range, reducing the bias of characteristics with considerable numerical impact and, consequently, balancing the influence of all characteristics. This method can improve further processing, especially for AI-based algorithms, thereby enhancing efficiency and accuracy. The following methods, shown in Equations 2.1 and 2.2, are two of the most common approaches to normalization [8, 10]:

- **Min-Max normalization** is performed using the following formula:

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_max_A - new\_min_A) + new\_min_A \tag{2.1}$$

  where:

  - A is the feature being normalized.
  - v is the original value of the feature.
  - v' is the newly scaled value.

- **Z-score normalization** uses the mean of all the values of the feature and its standard deviation, as stated below:

$$v' = \frac{v - mean_A}{stand\_dev_A} \tag{2.2}$$

After cleaning the dataset, the resulting data is transformed into a compact set of numerical indicators. This process is called feature extraction and allows a simplification of the data, resulting in more efficient processing models and better predictive prognosis. Various parameters can be calculated and are generally divided into two categories: time-domain features and frequency-domain transformations.

Features calculated in the time domain are commonly referred to as health or condition indicators. These are computationally efficient parameters that represent the overall condition of machinery and, in PdM, usually represent vibration data [21]. The most common and pertinent features are presented below in Equations 2.3 through 2.9:

- **Peak-to-peak**: Difference between maximum and minimum value of the signal x.

$$\mathrm{PP} = max(x) - min(x) \tag{2.3}$$

- **Root Mean Square (RMS)**: Reflects the vibration amplitude and energy of the signal in the time domain.

$$\mathrm{RMS} = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} x_i^2} \tag{2.4}$$

  where:
  N— number of samples,
  i— sample index,
  x_i— i-th signal sample.

- **Crest Factor**: Peak value divided by the root mean square. The crest factor can provide an early warning of faults, since they often first manifest as changes in a signal's peakedness before they appear in the energy.

$$\mathrm{CF} = \frac{max(x)}{\mathrm{RMS}_x} \tag{2.5}$$

- **Standard Deviation**: Standard measure of signal dispersion, measures the amount of variation from the mean $\bar{x}$ of the data set.

$$\mathrm{SD} = \sqrt{\frac{1}{N-1} \cdot \sum_{i=1}^{N} (x_i - \bar{x})^2} \tag{2.6}$$

- **Kurtosis**: Provides a measure of the peakedness of the signal, as developing faults can increase the frequency and severity of outliers. It is calculated as the fourth-order normalized moment of a given signal.

$$\mathrm{K} = \frac{N \cdot \sum_{i=1}^{N}(x_i - \bar{x})^4}{\left(\sum_{i=1}^{N}(x_i - \bar{x})^2\right)^2} \tag{2.7}$$

- **Skewness**: Measures the asymmetry of a probability distribution around its mean. Depending on how the data points deviate from the normal distribution, the distribution can be right- or left-skewed.

$$\mathrm{S} = \frac{N \cdot \sum_{i=1}^{N}(x_i - \bar{x})^3}{\left(\sum_{i=1}^{N}(x_i - \bar{x})^2\right)^{3/2}} \tag{2.8}$$

Whereas time-domain indicators greatly facilitate processing models in the identification of anomalies, they struggle to help identify the origin of said anomaly. Shifting from the time to the frequency domain accomplishes that. For that purpose, the use of Fast Fourier Transform (FFT) allows decomposing complex vibration signals into their sine-wave components, yielding spectral lines that represent the amplitude at specified frequencies [29]. This algorithm is calculated using the following formula:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-j\frac{2\pi kn}{N}} \tag{2.9}$$

where:
X_k—complex amplitude of the k-th spectral line,
x_n—n-th signal sample in the time domain,
N—total number of samples,
k—frequency index ($k = 0, 1, \ldots, N-1$),
n—time index ($n = 0, 1, \ldots, N-1$),
j — imaginary unit,
$e^{-j\frac{2\pi}{N}kn}$—Twiddle factor.

For machinery operating at varying speeds or loads, the resulting signal is not stationary, so a STFT is used instead. This algorithm is implemented by splitting the vibration signal into overlapping frames, in which localized spectral analysis can be performed on individual segments [37].

Using the data's frequency representation, spectral features can be calculated to reduce the data to selected parameters that improve prediction efficiency. Equations 2.10, 2.11 and 2.12 represent the most relevant features:

- **Mean Frequency**: indicates the vibration energy in the frequency domain.

$$\mathrm{MF} = \frac{1}{N} \cdot \sum_{k=1}^{N} X_k \tag{2.10}$$

- **Frequency Center**: calculates the dominant frequency or characteristic central frequency of the signal.

$$\mathrm{FC} = \frac{\sum_{k=1}^{N} f_k \cdot X_k}{\sum_{k=1}^{N} X_k} \tag{2.11}$$

- **Velocity RMS**: broadband indicator, calculates the velocity spectrum from vibration signals without the noise amplification typical of time-domain integration.

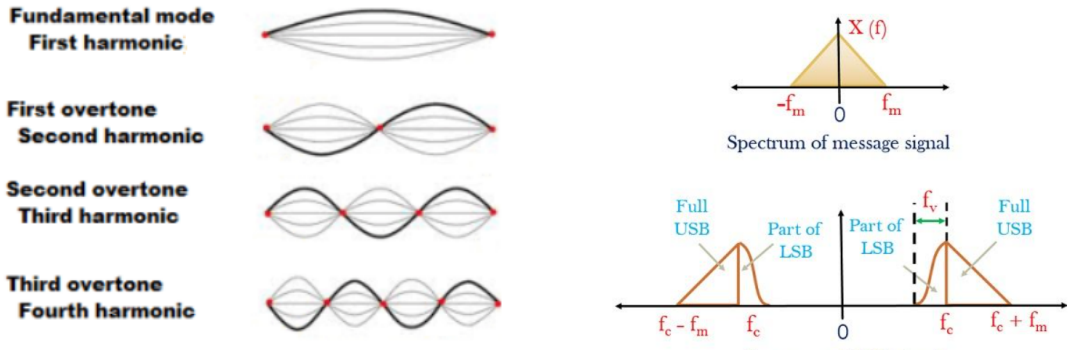
$$\mathrm{RMS}_v = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^{N} v_i^2} \tag{2.12}$$

where:
v_i—i-th sample of the velocity signal via frequency-domain integration.

Analyzing the signal in the frequency domain is particularly helpful for isolating the contributions of different mechanical components. Each component not only has a known signature frequency based on its rotational speed, which appears at certain intervals, but also typical faults will generate frequencies that have been pre-established for each category. For example, if a bearing has a certain defect, based on the detected frequency and historical data, we can determine whether the defect is on the inner ring, outer ring, rolling element, or bearing cage. This significantly improves anomaly detection and maintenance planning [36].

Furthermore, with increased mechanical wear, the vibration pattern becomes more complex, and its harmonic components and sidebands can be observed in the frequency spectrum for deeper analysis. Harmonics represent integer multiples of the signal's base frequency and can reveal issues such as mechanical looseness and misalignment at higher harmonics [57]. Sidebands are a result of modulation in signal processing and manifest as peaks on either side of the center frequency. This phenomenon indicates damage that occurs periodically in a revolution, such as a cracked tooth in a gear.

Examples of these signal modulations can be observed in Figure 2.10:



(a) Vibration harmonics [65]. (b) Vibration sidebands [66].

Figure 2.10: Comparison of physical vibration modes and signal frequency modulation.

Another common solution for analyzing vibration data is to use spectrograms. These are designed to capture the varying frequency components of a signal over time. Spectrograms are often chosen due to their ability to show both time and frequency information in the same domain, where the horizontal axis denotes time, the vertical axis denotes frequency, and different colors represent the energy of the Time-Frequency representation (TFR) [39]. The representation of the spectrogram can be written as equation 2.13.

$$S(\tau,\omega) = \int x(t)g(t-\tau)e^{-j\omega(t-\tau)}\,dt \tag{2.13}$$

where τ and ω represent time and frequency, $g(t)$ denotes the short window used to truncate the original signal $x(t)$.

As a final step before prediction analysis, after computing all features, Principal Component Analysis (PCA) can be applied to select principal components for anomaly detection, efficiently consolidating the data without meaningful data loss [23]. This dimensionality reduction technique selects new, uncorrelated variables that preserve maximum variance, reducing the overall number of variables and, as such, making data analysis faster.

2.5.2 Anomaly Detection

The most important step in PdM is detecting anomalies. The entire system depends on automatically identifying faults and predicting future problems to create an appropriate maintenance strategy. In this chapter, several methods for anomaly detection, failure prediction, and RUL estimation will be compared, with each method's uses, benefits, and limitations discussed.

2.5.2.1 Statistical Models

For decades, traditional statistical approaches were the industry standard. These adjust a predefined distribution to the given data, inferring, through statistical methods, whether an instance belongs to that model. Statistical algorithms work better with simple structured data, where failure mechanisms follow clear probabilistic behavior [43].

Statistical models can be divided into two different classes: parametric and non-parametric. Parametric algorithms require an underlying data distribution and are generally based on Gaussian or regression models. On the other hand, non-parametric techniques are preferred for their versatility, as they do not require a pre-established model structure [15].

These are some of the most relevant statistical algorithms:

- **Box-plots:** Standardized graphical method for displaying the distribution of numerical data based on a five-number summary: minimum, first quartile (Q_1), median, third quartile (Q_3), and maximum. They are effective for central-value estimation

and, consequently, for identifying outliers that fall outside the interquartile range of the monitored dataset.

- **Grubbs test:** This technique is designed for univariate data to identify a single outlier within a dataset that follows an approximate normal distribution. In machinery diagnostics, it can be utilized to detect sensor glitches, electrical surges, or isolated mechanical shocks.

- **Distribution Comparison Tests:** This category includes the Kolmogorov-Smirnov, Kruskal-Wallis, and Wilcoxon signed-rank tests. These methods mathematically compare vibration probability distributions to detect alterations in relation to the healthy baseline.

- **Auto-regressive Models:** These time-series models capture temporal dependencies of the combination of previous values and are capable of forecasting future values. Techniques like ARIMA combine autoregressive models with integration and moving averages, resulting in a more robust approach that also accounts for past failed predictions. While useful for trend analysis, they often lack consistency when subjected to the non-linear or abruptly shifting behaviors common in real-time maintenance scenarios.

- **Hidden Markov Models (HMM):** HMMs assume that a system moves through hidden states over time, such as "healthy", "worn", and "failed", and each hidden state produces an observable output based on certain probabilities. One of its main limitations is computational inefficiency. Despite the introduction of the expectation-maximization algorithm, the implementation of these models for real-time applications remains a challenge.

These classical methods are, however, affected by several problems that limit their contemporary application:

- Dependence on the system being stationary and following a concrete data distribution;

- Difficulty in processing complex or high-dimensional data;

- Necessity of user intervention in feature engineering and domain-specific model calibration.

2.5.2.2 Artificial Intelligence Models

With the aim of solving statistical approach limitations, data-driven AI models are now the established methodology, and the incorporation of ML and DL into PdM has vastly improved the precision and efficiency of failure prognosis [13].

There are four main types of machine learning algorithms for anomaly detection: classification, regression, clustering, and similarity-based algorithms.

Classification learning is a supervised machine learning technique, meaning it requires assigning labels to the dataset during training. This is the first phase of the technique, where the algorithm learns from the available samples and generates a classifier. The second phase is validation, in which samples the classifier has not seen are used to measure the model's performance and tune its parameters. The last phase is testing, where a final blind test is performed to assess the model's accuracy and prevent memorization (overfitting). The model can then assign input data to predefined categories or labels. Classification-based techniques have a quick testing phase, since each instance is compared against the predefined model. It also has the advantage of being able to distinguish between observations that belong to different anomalies (instead of an overall class called "anomaly") [15].

Regression algorithms are supervised machine learning techniques used to predict continuous numerical output values from input features. This method analyzes incoming data and estimates future values using time-series analysis. Input sensor readings (y) are then compared with the model's predicted value ($\hat{y}$). The difference between them is the residual error ($e = |y - \hat{y}|$). If the machine starts deteriorating, its behavior becomes unpredictable to the model, causing the residual error to spike. Increased residual error values indicate anomalies in the machine state. Consequently, this technique is primarily useful for diagnosis and RUL prediction [22].

Clustering techniques are unsupervised machine learning techniques that divide unlabeled data into groups (clusters) based on shared similarities. These algorithms are generally split into two stages: first, the data is grouped into clusters using clustering algorithms; and then the deviation is analyzed relative to the clusters. As such, normal data points will be closer to the cluster centroid, while anomalous data points will be farther away. In addition, normal samples are more likely to be located in large, dense groups, whereas anomalies are more likely to be found near small, disparate clusters.

Similarity-based learning is a subclass of supervised machine learning that, rather than classify data points into fixed categories, measures the similarity between data observations. The data is mapped into an n-dimensional space where geometric distances indicate similarity. One of the biggest advantages of these techniques is that, since they do not try to identify which class each data point belongs to, they can recognize never-seen anomalies by analyzing how much each instance deviates from the normal state. At the same time, due to this architecture, this algorithm is not strongly affected by class imbalance (when the majority of data is deemed normal, with very few anomalies, as is typical of normal machinery sensory data), unlike the previously mentioned methods [40].

While these algorithm alternatives define the core mathematical objectives of anomaly detection, their practical implementation within industrial frameworks depends heavily on how data features are processed. ML models rely on feature engineering; that is, human

manual intervention is needed to establish the initial input parameters. On the other hand, DL, which is a subdivision of ML, assembles algorithms in layers to produce a NN that is capable of self-learning from raw data without the need for prior feature extraction. Deep Hybrid Learning (DHL) combines these two methodologies by extracting features using DL models and performing classification using ML algorithms [7].

2.5.2.3 Machine Learning Algorithms

Traditional ML algorithms represent a highly resilient paradigm for real-time PdM. These models possess low computational complexity and minimal memory usage, making them a solid option for edge computing. The next section details the main principles of specific ML models and their applications.

- **Decision Tree**: A Decision Tree is a supervised machine learning algorithm that is used for both classification and regression. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Each decision or branch in the tree is based on a specific feature or attribute of the data, and the leaf nodes provide the model's outcome or prediction. Decision Trees are straightforward to interpret, but for PdM, this algorithm tends to overfit the training data and miss evolving faults [19]. Figure 2.11 showcases an example of a decision tree.

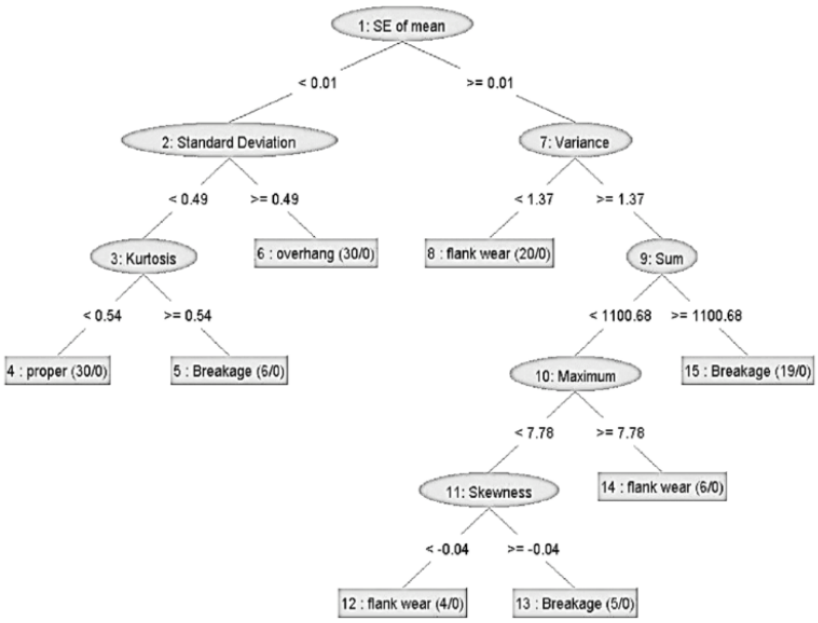


Figure 2.11: Decision tree example for vibration data analysis[67].

- **Random Forest**: A Random Forest is a collection of individual decision trees. Instead of trusting a single deep tree to classify machinery health, a Random Forest constructs a large collection of decorrelated decision trees during the training phase and combines their individual outputs through a majority voting protocol to determine the final fault classification. This algorithm is an established technique for fault prognosis and reduces the overfitting issues that affect decision trees [21]. Figure 2.12 shows the broad architecture of this algorithm.

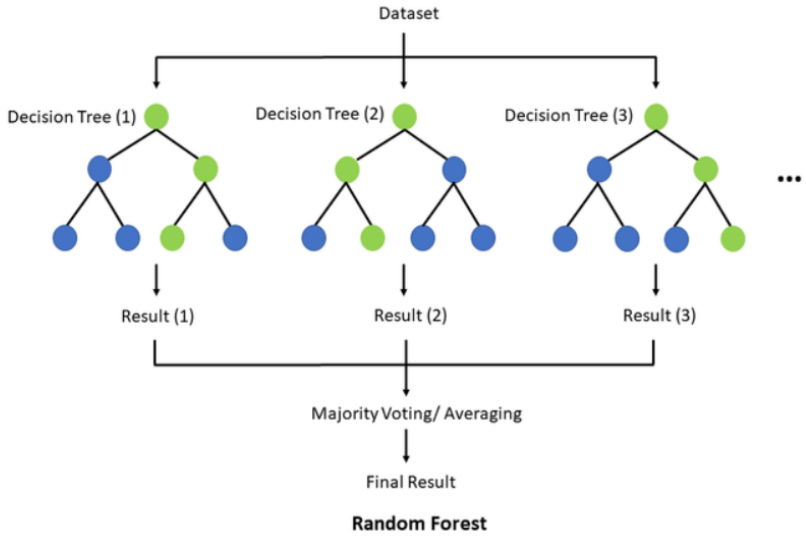


Figure 2.12: General architecture of a Random Forest classifier [68].

- **SVM**: SVM is a supervised machine learning algorithm used for classification and regression analysis. Data is mapped into a high-dimensional space, and points on the boundary form support vectors that define a hyperplane separating data points of different classes. This hyperplane represents the boundary that optimizes the margin, or the distance between the decision boundary and the nearest data points of each class. This algorithm can also create non-linear decision boundaries using kernel functions such as polynomial, radial basis function, or sigmoid function [43]. Figure 2.13 illustrates the difference between the linear and non-linear variants of SVM.

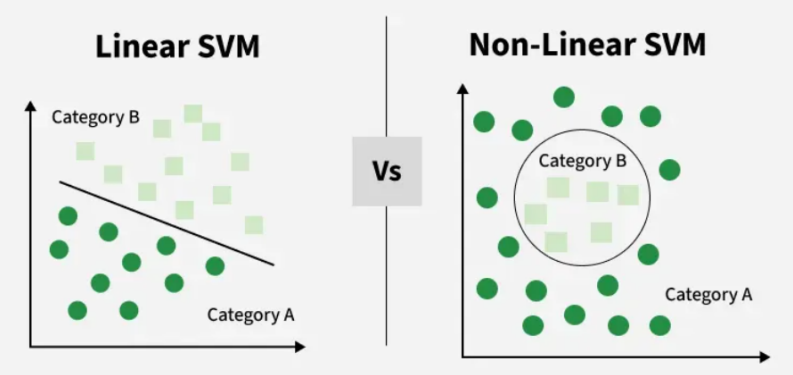


Figure 2.13: Linear vs Non-Linear SVM visual representation [69].

- **KNN**: KNN is a type of supervised machine learning algorithm that is used for solving classification or regression problems. This algorithm uses distance-based rules without assuming a fixed model, and new data points are classified based on the class of the K-nearest neighbors in the training set. To find the K-nearest neighbors, this technique calculates the distance between the new data point and each point in the training set using the Euclidean (most common), Manhattan, or Minkowski distances [40]. Figure 2.14 depicts a visual representation of this technique.

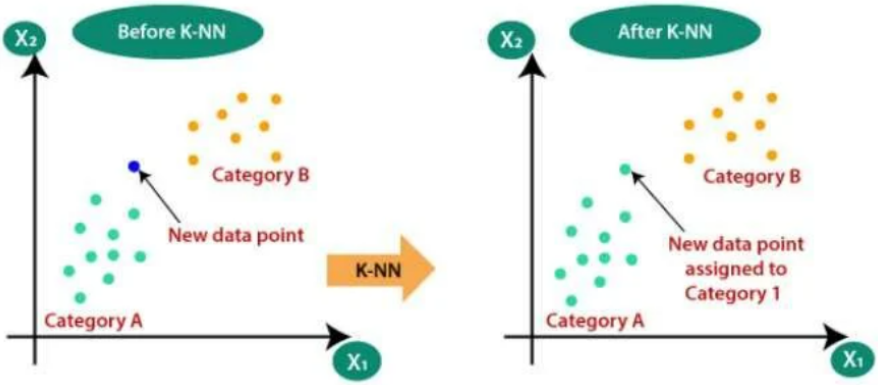


Figure 2.14: KNN model representation [70].

- **XGBoost**: XGBoost is a supervised machine learning algorithm used for regression, classification, and ranking tasks. This algorithm creates a decision-tree ensemble model in which each tree is trained to correct the errors of the previous one [43]. It features parallel processing for faster training on high-dimensional datasets and allows parameter customization to optimize performance [71].

- **K-Means**: K-Means is an unsupervised machine learning algorithm used to group unlabeled data into K clusters of similar data points. This algorithm is commonly used for anomaly detection because it can separate normal baseline data from anomalous states [72].

2.5.2.4 Deep Learning Algorithms

DL architectures use multiple layers of artificial neurons to identify patterns directly from raw sensor data without requiring prior feature engineering. These algorithms can handle complex, non-linear data and identify unknown fault scenarios (anomalous data not detected during model training) with higher accuracy than traditional ML methods. This section presents the most common and relevant models for PdM.

- **CNN**:CNNs are algorithms for various classification tasks. It is composed of three types of layers: convolutional, pooling, and fully connected [40]. Convolutional layers in vibration data analysis are used for feature extraction and for identifying modulation patterns and repetitive defect frequencies [21]. Pooling layers are used to downsample data using operations such as max-pooling (selecting the most dominant value within a chosen window), reducing computational overhead. Fully connected layers are used for classification, using the data learned by the previous layers. Figure 2.15 schematizes this architecture.

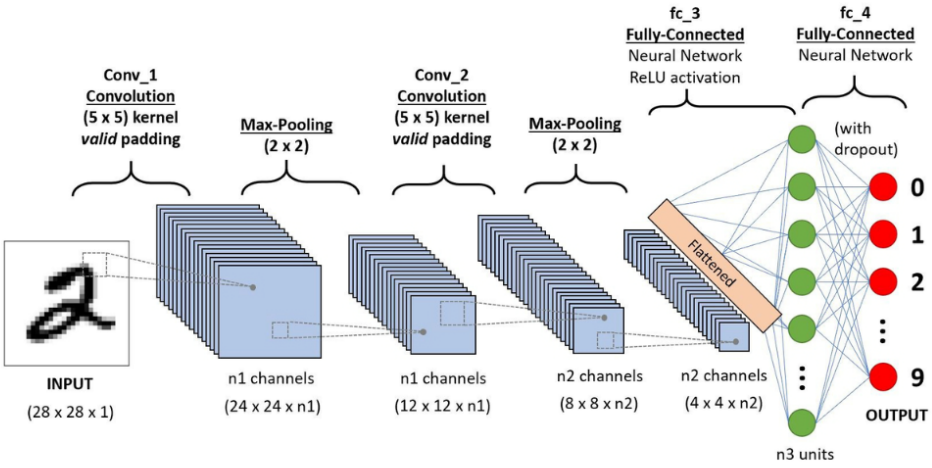


Figure 2.15: Standard topology of a CNN [73].

- **LSTM**: The LSTM model belongs to the RNN category, and is commonly used for processing time-series sequential data [22]. The LSTM model is composed of several key components: the block input, input gate, forget gate, output gate, and block output. For this application, LSTM models outperform RNN architectures due to their gating mechanisms, which control what historical data is preserved or discarded and thus minimize the vanishing gradient problem that affects standard RNNs.

2.5.3 Explainable AI

With the aim of turning the outputs of classification models into actionable maintenance steps, XAI is a contemporary approach for improving model interpretability [33]. Instead of simply getting a percentage of certainty of an error from the classification model, algorithms like SHAP give the user information regarding which health indicator justified the anomaly diagnosis, as well as an indication of the root of the mechanical problem [13]. The incorporation of such models into the PdM framework significantly improves decision-making and increases transparency, enhancing the maintenance team's trust in the model's predictions [18].

2.5.4 Evaluation Metrics

To evaluate and benchmark the architectural performance of different algorithms, evaluation metrics are used to assess the model's effectiveness, using both visual and scalar measures [29]. The most pertinent classifier metrics are stated as follows:

- **Accuracy**: The ratio of correct predictions to total predictions.

$$\text{accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \tag{2.14}$$

  where:
  TP—True Positive (correctly identified positive cases),
  TN—True Negative (correctly identified negative cases),
  FP—False Positive (negative cases incorrectly classified as positive),
  FN—False Negative (actual positive cases incorrectly classified as negative).

- **Precision**: The ratio of correct positive predictions to total predicted positives.

$$\text{precision} = \frac{TP}{TP+FP} \tag{2.15}$$

- **Recall**: The ratio of correct positive cases to total actual positive cases.

$$\text{recall} = \frac{TP}{TP+FN} \tag{2.16}$$

- **F1 score**: The harmonic mean of precision and recall.

$$\text{F1} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}} \tag{2.17}$$

- **Confusion Matrix**: A confusion matrix is a table that displays the number of correct and incorrect classifications for each class, as shown in Figure 2.16.

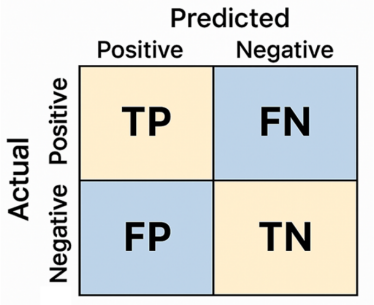


Figure 2.16: Structure of a Confusion Matrix [74].

2.6 Conclusion

In summary, there are several aspects to take into consideration regarding PdM, whether concerning important steps to take into account or alternative contemporary methods that can improve the overall result.

Furthermore, a relevant consideration is the balance between efficiency and accuracy in anomaly detection. Since anomaly detection processing is usually performed on edge devices, such as PLCs or IPCs, computational efficiency is critical to enable the device to process data synchronously with the monitored machinery operation.

Chapter 3

Methodology and Development

This chapter details the architectural design and experimental methodologies developed in this dissertation. Firstly, the hardware and software configurations of the experimental setup are presented, followed by an outline of the delineated experimental trials used to simulate machinery anomalies. Finally, a data-processing pipeline, in the form of a benchmarking series, is described for the anomaly detection tests.

3.1 Setup

With the intention of testing and simulating machinery anomaly scenarios, an experimental infrastructure was developed to determine the proposed system's capabilities. This architecture is split into two main components: the physical Experimental Test Rig (ETR), a workbench designed to replicate the dynamics of complex rotational machinery, and the MMS, a mobile system capable of real-time data acquisition and edge computation.

3.1.1 Experimental Test Rig

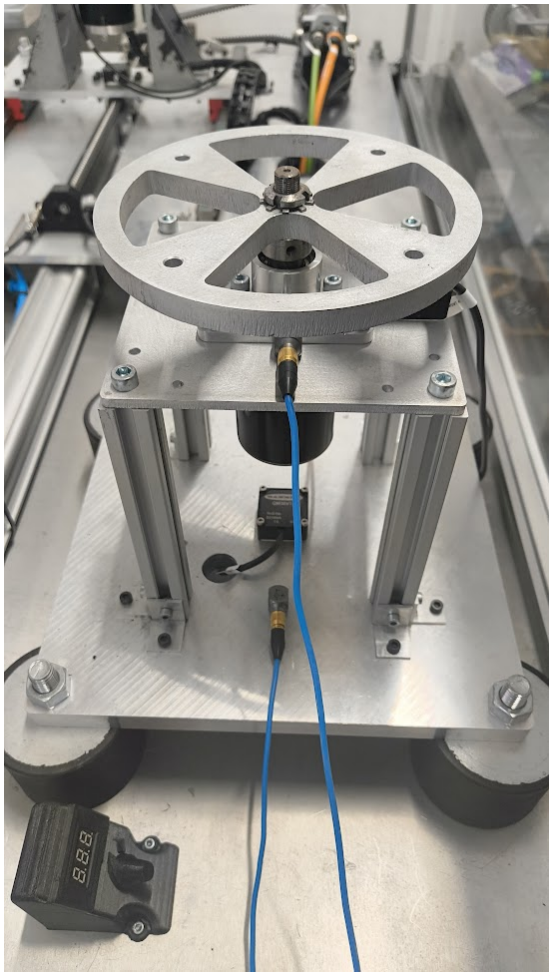
The ETR is a test bench designed to detect imbalances in rotational motion and, as such, is ideal for PdM and anomaly detection simulation. This experimental rig includes two distinct setups:

- ***Rig 1***: Fixed-Axis Inertia Wheel
- ***Rig 2***: Translational-Axis Inertia Wheel

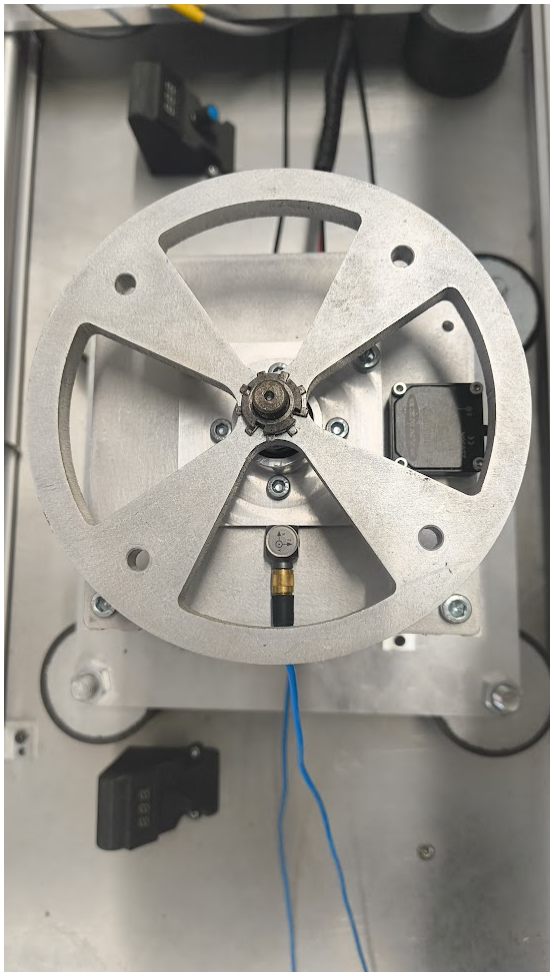
Rig 1 consists of a 24 V DC motor in a vertical position coupled to an inertia wheel, which is mounted on a dedicated bearing housing. This inertia wheel features a round threaded-hole pattern, where counterweights can be mounted at a fixed radial distance, thereby creating a predefined mass imbalance in the wheel's rotational motion. The DC motor has a maximum speed of 3500 rpm and is controlled by a potentiometer positioned

next to the setup, allowing the wheel's rotational speed to be altered. Throughout the experimental trials, the inertia wheel operates at a nominal rotational speed of approximately 1000 rpm, with a secondary dataset acquired at approximately 1200 rpm to evaluate cross-speed generalization. These values were validated using a tachometer. Figure 3.1a shows an image of this setup.

With the intention to monitor the data from the ETR, two Integrated Electronics Piezo-Electric (IEPE) triaxial accelerometers were selected to record vibration data. The selected sensors were Kistler 8763B, which possess a high-frequency response of 10 kHz; low-impedance output for electrical noise isolation; and shear-element technology, ensuring high immunity to base strain. The sensors were mounted using the brand's recommended special adhesive compound. Two triaxial sensors were implemented in this rig, with a total of six independent vibration channels captured simultaneously, ensuring the device is monitored in two distinct positions and thus guaranteeing data reliability. The sensors' placement and orientation, identifiable by their blue cables, are shown in Figures 3.1a and 3.1b.



(a) *Rig 1* configuration



(b) Sensor position.

Figure 3.1: *Rig 1*: Fixed-Axis Inertia Wheel.

Rig 2 has a similar configuration, and the inertia wheel motor and sensors have identical placement and orientation. This setup, however, is not fixed, and combines linear translation profiles with the main rotating element. As shown in Figures 3.2 and 3.3, the system is driven by an industrial Beckhoff AC servomotor coupled to a ball screw assembly via a belt transmission. This ball screw drives a mobile carriage along two parallel linear guide rails, with a length of 1250 mm.

During testing, the carriage executes a continuous linear shuttle motion defined by a target velocity of 300 mm/s, acceleration and deceleration rates of 2000 mm/s^2, and a jerk limit of 20000 mm/s^3. This motion profile was devised to simulate the movement of a CNC axis. The translation is controlled using a hybrid control strategy: at one end, there is a limit switch, while at the other end the motion is dynamically restrained via the servomotor's encoder feedback. *Rig 2* motion control was executed using the code shown in Appendix D and a video displaying its movement is available in Appendix B.

From a PdM perspective, this design enables the simulation of complex operational scenarios, in which the imbalance forces that influence the wheel's rotation are affected not only by the masses' instability but also by the dynamic translational motion of the carriage.

Furthermore, *Rig 2* is equipped with three Beckhoff EtherCAT terminals to control its electromechanical actuation. The EL2008 digital output terminal enables control of the inertia wheel's rotational speed by supplying power to the potentiometers. The EL7211 servomotor terminal serves as the driver for the AC servomotor, regulating the carriage's translational motion. The EL1008 digital input terminal is responsible for acquiring limit switch data.

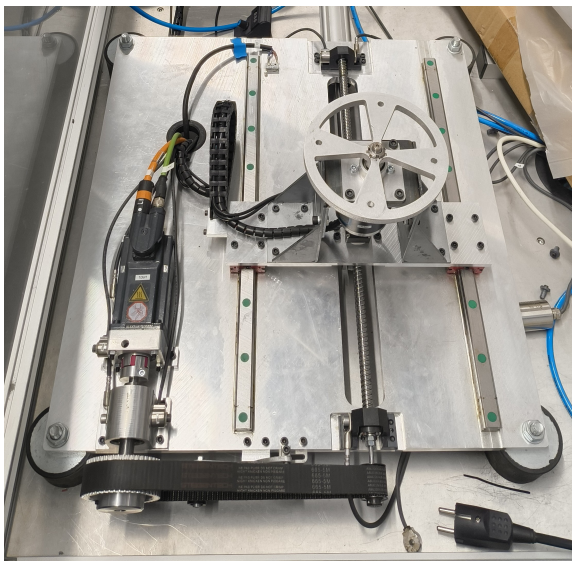


Figure 3.2: *Rig 2*: Translational-Axis Inertia Wheel - top view.

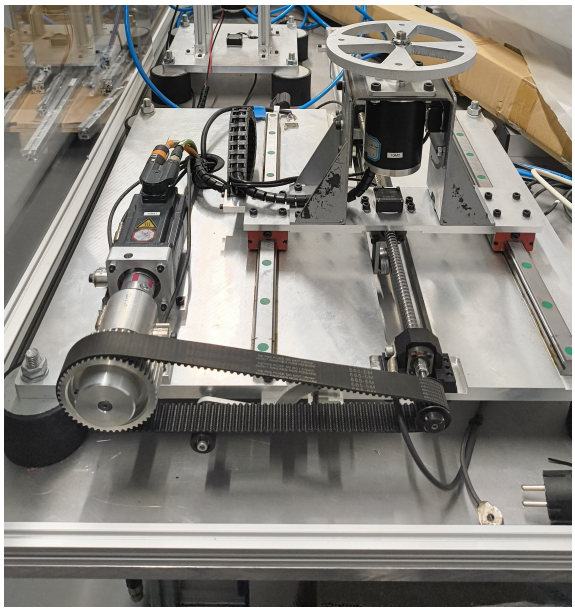


Figure 3.3: *Rig 2*: Translational-Axis Inertia Wheel - side view.

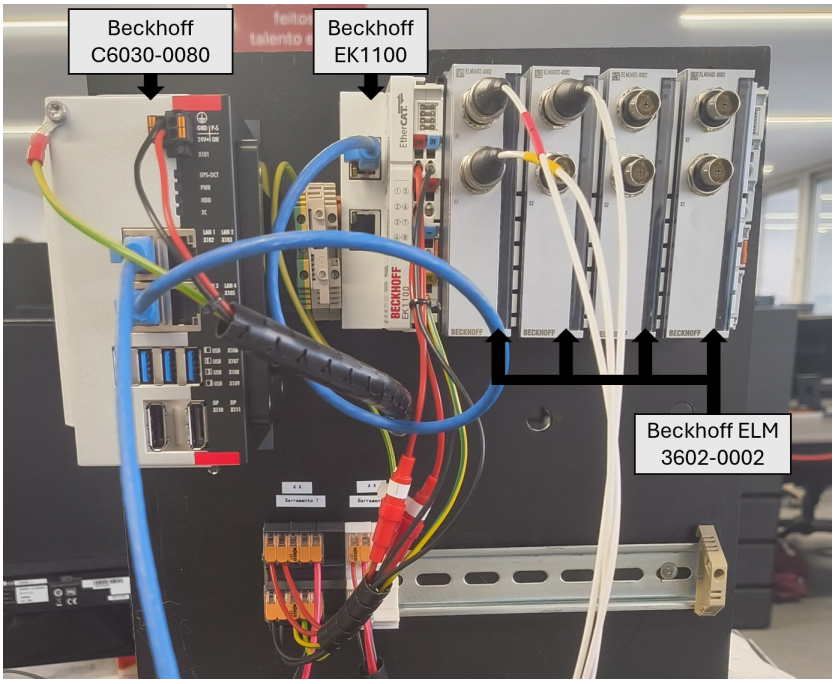
3.1.2 Mobile Monitoring System

The MMS is a powerful mobile edge device designed to acquire sensor data at high sampling rates, process the data, and perform machine health diagnostics. This device incorporates the following components:

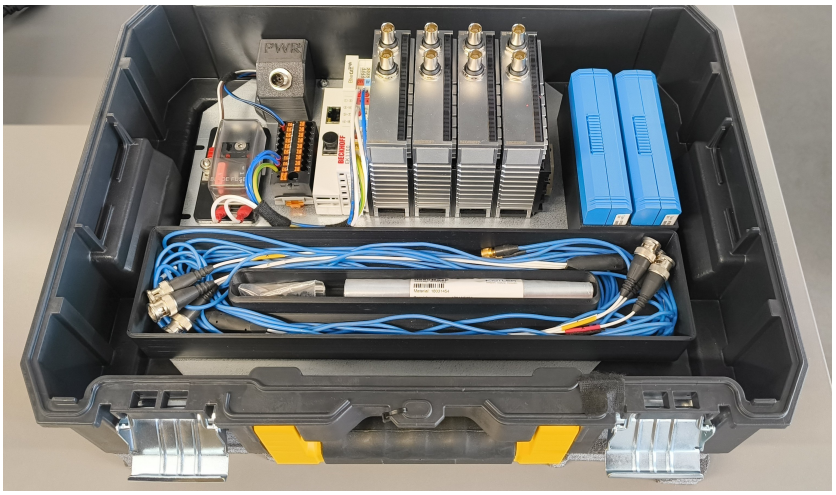
- **Beckhoff C6030-0080**: This IPC acts as the main computational processor, performing data acquisition, database storage, and running the diagnostic models. It is equipped with an Intel i7 processor and runs a real-time Windows/TwinCAT 3 environment. The IPC interfaces with the I/O modules via the EtherCAT protocol, enabling real-time synchronized communication across the entire fieldbus network.
- **Beckhoff EK1100**: This module serves as the primary hardware gateway, converting the EtherCAT data into internal E-bus signal data utilized by the local I/O modules. Simultaneously, this module is responsible for powering all the attached terminals at a constant 24 V DC.
- **Beckhoff ELM-3602-0002**: Beckhoff ELM-3602-0002 are dual-channel high-precision measurement terminals specifically designed for vibration data acquisition. This module offers native IEPE coupling via a shielded BNC connection capable of simultaneously transferring data and supplying power to the Kistler accelerometers. This terminal operates with a 24-bit ADC and is able to oversample data up to 50 kHz per channel. The module also incorporates internal hardware-level digital filters, providing clean time-series arrays for real-time edge processing.
- **Beckhoff EL9011**: Bus end cap that covers and protects the exposed E-bus contacts of the last ELM terminal.

- **Electrical connections**: A 24 V power supply is used to power the IPC and the Beckhoff EK1100. Circuit protection is provided by slow-blow fuses: 6.3 A for the IPC and 1 A for the Beckhoff EK1100.

The physical layout of this system is presented in Figure 3.4a, and its mobile configuration in Figure 3.4b. The electrical schematic of the MMS is provided in Appendix F.



(a) Hardware layout.



(b) Mobile configuration.

Figure 3.4: MMS configuration.

3.2 Experimental Tests

A series of tests of increased complexity was devised in order to simulate industrial scenarios and evaluate the MMS. The tests are stated as follows:

- **Test 1**: Hammer impact detection

Rig 1 - Static State

- **Test 2**: Anomaly detection binary classification
- **Test 3**: Anomaly detection binary classification with untrained anomaly class
- **Test 4**: Anomaly classification
- **Test 5**: Anomaly classification with alternative untrained speed dataset testing

Rig 2 - Static State

- **Test 6**: Anomaly detection binary classification
- **Test 7**: Anomaly detection binary classification with untrained anomaly class
- **Test 8**: Anomaly classification
- **Test 9**: Anomaly classification with alternative untrained speed dataset testing

Rig 2 - Dynamic State

- **Test 10**: Anomaly detection binary classification
- **Test 11**: Anomaly detection binary classification with untrained anomaly class
- **Test 12**: Anomaly classification
- **Test 13**: Anomaly classification with alternative untrained speed dataset testing

Rig 2 - Cross State

- **Test 14**: Anomaly classification with static state training data and untrained dynamic state data

Test 1 was designed to calibrate and test the MMS. The sensors were mounted on a steel beam of a CNC machine structure, as presented in Figure 3.5a, and a Kistler 9724A Impulse hammer, shown in Figure 3.5b, was used to generate high-magnitude anomalies. Since this impulse hammer has an IEPE output, the impact's magnitude and response are known, and, as such, by analyzing the accelerometers' and hammer's output values, the sensors were calibrated to gravitational force equivalent units (g-unit). This test

also allowed an initial approach to anomaly detection. Considering the hammer strike represents a single high-magnitude impact, it creates a clear spike in the accelerometers' output and, therefore, it is an easier anomaly to detect.

(a) CNC beam structure.

(b) Kistler piezoelectric hammer.

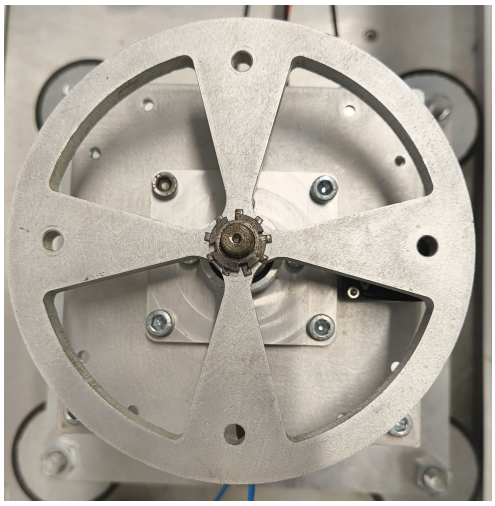
Figure 3.5: *Test 1* setup.

For the remaining tests, *Rig 1* and *Rig 2* were used, and 5 distinct imbalance scenarios were considered through these experiments:

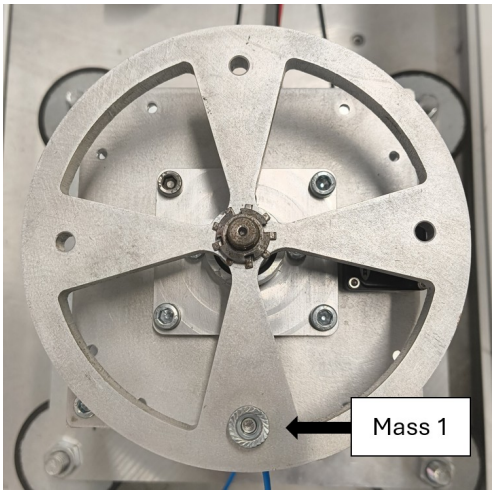
- **Normal:** No imbalance
- **Imbalance 1:** Mass 1 : 7g
- **Imbalance 2:** Mass 1 + Mass 2 : 20g
- **Imbalance 3:** Mass 2 : 13g
- **Imbalance 4:** Mass 2 + Mass 3 : 26g

Note: The imbalance scenarios are numbered according to the physical mass-mounting sequence used during data acquisition, rather than by increasing imbalance magnitude.

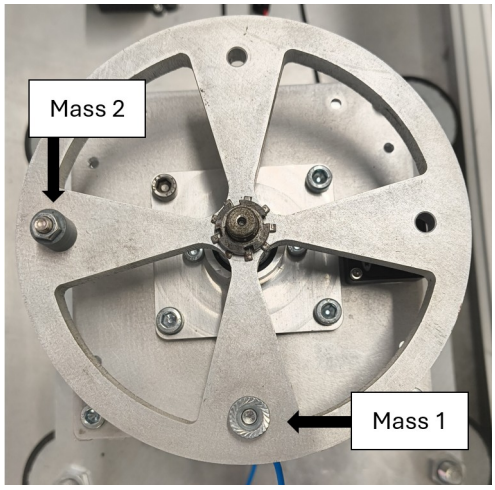
The following images in Figure 3.6 illustrate these 5 scenarios and the masses' positioning.



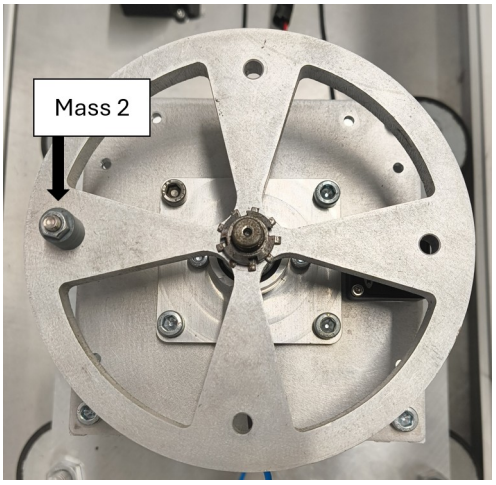
(a) Normal



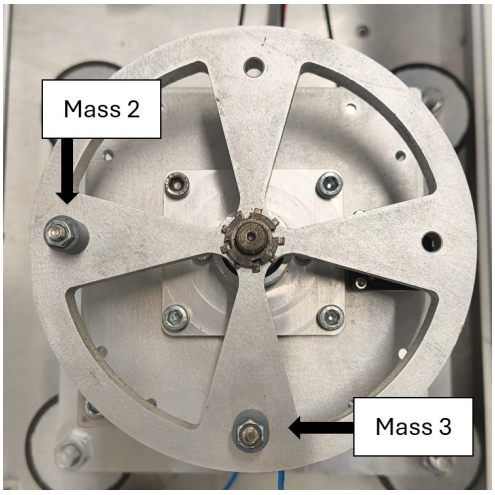
(b) Imbalance 1



(c) Imbalance 2



(d) Imbalance 3



(e) Imbalance 4

Figure 3.6: Inertia Wheel mass positioning.

For Test 2, implemented in *Rig 1*, the first four classes - Normal, Imbalance 1, Imbalance 2, and Imbalance 3 - were recorded at a measured average rotational speed of 1000 rpm. The defined AI models were benchmarked for binary classification between the Normal and Anomaly classes.

Test 3 is an extension of Test 2. In this case, all imbalance scenarios were recorded at the same speed defined in Test 2. In this test, one anomaly was held out during training to determine the system's ability to detect unknown faults. The experimental setup was divided into two distinct alternatives:

- **Test 3.1:** The classification models were trained using the Normal, Imbalance 1, Imbalance 2, and Imbalance 4 datasets. Imbalance 3 was withheld from training and utilized as an unseen test class.
- **Test 3.2:** The models were trained using the Normal, Imbalance 1, Imbalance 2, and Imbalance 3 datasets, with Imbalance 4 tested as the unknown anomaly.

This trial's main objective was to evaluate the different models' adaptability by simulating industrial situations in which unexpected, unseen faults occur.

For Test 4, the first four imbalance scenarios were recorded at a measured average rotational speed of 1000 rpm, identical to Test 2. However, in this trial, the models were evaluated on classification across all four classes to test the algorithm's recognition of each class. This test expanded the AI model's diagnostic capabilities, testing whether the model can not only detect anomalies, but also classify the specific fault scenario.

Test 5 is a continuation of Test 4. It is trained on the same four imbalance scenarios at 1000 rpm; however, a second dataset of these four imbalances is recorded at 1200 rpm, which is used solely as testing data. The objective of this trial is to assess the models' speed invariance capability, that is to say, the capacity to correctly identify the same fault scenarios at an untrained speed, simulating an industrial scenario where there are slight variations in the working conditions of the machine.

An important point to take into account is that most PdM studies encountered in the literature review train and test their models under identical operating conditions, rarely evaluating generalization to unseen faults or, in particular, to unseen speeds. This cross-speed setup addresses that gap, and thus establishes a novel contribution relative to the surveyed literature.

Tests 6 through 9 are performed in *Rig 2* in a static state, which means there is no translational movement actuated. These tests are analogous to Tests 2 through 5, using the same average rotational speed of 1000 rpm for training scenarios and 1200 rpm for the alternative untrained speed in Test 9. Since Tests 6, 7, 8, and 9 were recorded in the same working conditions as Tests 2, 3, 4, and 5, respectively, and the only difference is the physical structure of the motor/inertia wheel setup, the results are expected to be similar. However, these tests are needed to establish a baseline for the tests executed next in the dynamic state.

Tests 10 through 14 are executed in dynamic state in *Rig 2*. Data is recorded continuously and simultaneously with the carriage shuttle motion, including the carriage's acceleration, deceleration, and constant-speed motions, as described in Section 3.1.1. These tests' methodology is also equivalent to Tests 6 through 9, so the motor's rotational speed remains unchanged across both the trained and untrained speed sets. This scenario intends to simulate machinery operating conditions.

For Test 14, the model is trained with data recorded in static state, at 1000 rpm, from classes: Normal, Imbalance 1, Imbalance 2, and Imbalance 3. This model is then tested against an untrained dataset composed of these same classes, but recorded in dynamic state. This test aims to assess the model's cross-domain generalization and its ability to learn fault signatures across varying operational conditions.

This last test further expands this study's contribution to PdM, presenting an even less common evaluation scenario than the cross-speed case, as it assesses generalization across an entire change of operating state rather than a single varying parameter.

Table 3.1 summarizes each test's main parameters.

Table 3.1: Configuration of all experimental tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
		Rig 1 — Fixed-Axis			
2	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
3.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
3.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
4	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
5	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm
		Rig 2 — Static State			
6	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
7.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
7.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
8	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
9	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm
		Rig 2 — Dynamic State			
10	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
11.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
11.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
12	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
13	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm
		Rig 2 — Cross-State			
14	4-class	Normal, Imb. 1–3 (static)	Normal, Imb. 1–3 (dynamic)	1000 rpm	1000 rpm

3.3 Data Acquisition

The data is acquired from the MMS at a cycle time of 1 ms with an oversampling factor of 25, which means every millisecond, 25 data samples per sensor channel are extracted to the IPC. Within the TwinCAT environment, the Condition Monitoring module's function `CMA_Source` gathers raw data into a multiarray buffer to prevent data loss. The TwinCAT code for this function implementation is shown in Appendix C.1.

The data is then transferred to standard arrays, scaled, and converted from the original Integer format to Real, which is the required format for the majority of TwinCAT Condition Monitoring functions. This conversion prevents resolution loss in later processing stages. The code for this process is shown in Appendix C.2.

The data is then aggregated into 100 ms blocks (2500 samples), creating a time-series window appropriate for AI model training. The 100 ms window length was selected as a compromise between frequency resolution and responsiveness. At the nominal speed of 1000 rpm, this window captures approximately 1.67 revolutions. This window length is sufficient to compute the first-order component and its low-order harmonics, while maintaining low latency for real-time edge processing. To accomplish this, a ping-pong buffer is used to prevent read/write conflicts, as shown in the code in Appendix C.3. This means that while one process writes new data into the first buffer, another process reads data from the second, allowing continuous real-time data streaming.

These samples are then processed through a sequential state machine designed to manage the data storage without interrupting the real-time PLC cycle. This program generates a unique CSV filename and opens the file on the local disk, iterating through time-series windows and formatting the values as comma-separated strings. These strings are concatenated and written to the hard drive in bulk, followed by a safe file closure to ensure data integrity. The code to execute this logic is shown in Appendix C.4. This code also includes an operating mode selector that can alternate between saving the raw data to the hard drive, as previously described, and first pre-processing the data using feature extraction and FFT. The pre-processing is further explained in Section 3.4.

A summarized flowchart of the data acquisition pipeline is shown in Figure 3.7 to facilitate the comprehension of the software's architecture.

To facilitate the data acquisition process, a Human-Machine Interface (HMI) was devised in order to manually configure the operating mode, file labeling, and indexing. The HMI is shown in Figure 3.8 and was programmed using the code shown in Appendix C.5.

Using this acquisition pipeline, for each imbalance scenario and operating condition, approximately 300 independent 100 ms windows were recorded, yielding a balanced dataset of roughly 1200 samples per four-class experiment.

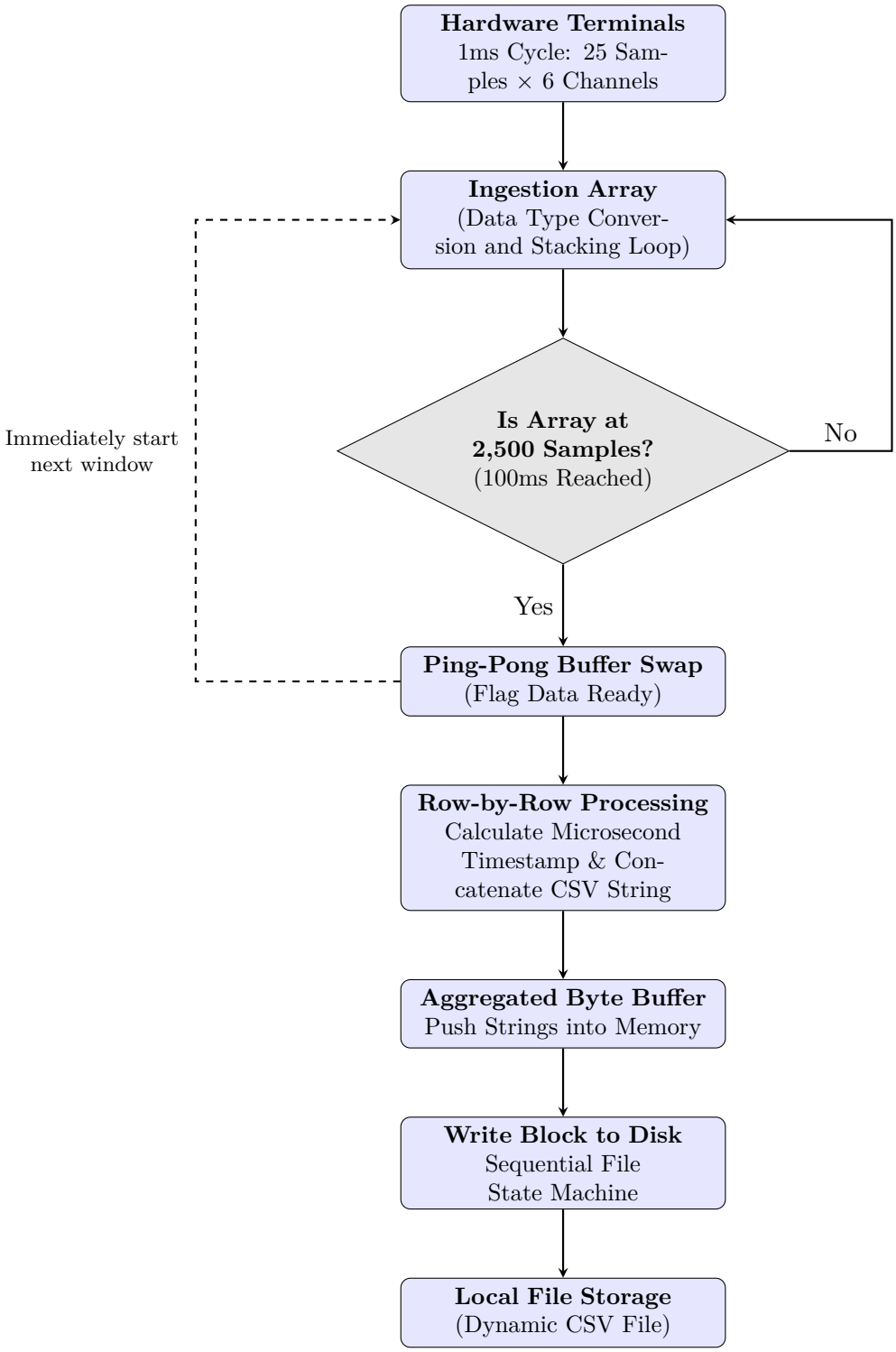


Figure 3.7: Flowchart detailing the raw data acquisition pipeline.

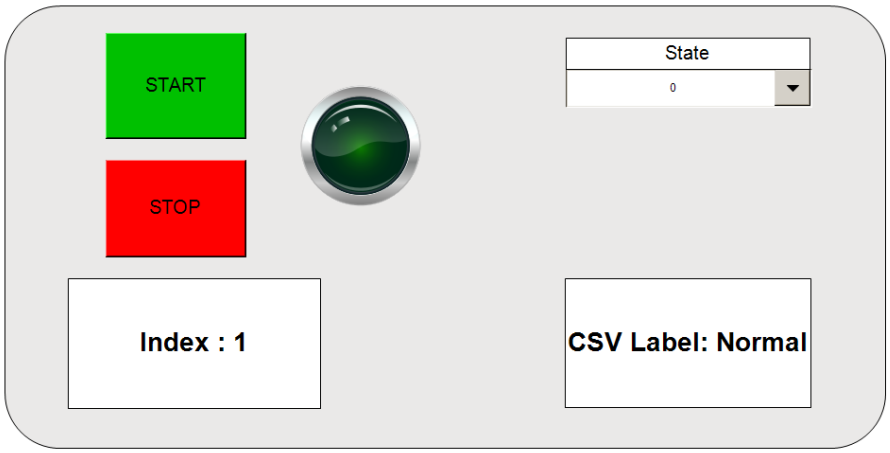


Figure 3.8: Data Acquisition HMI.

3.4 Data Processing and Anomaly Detection

For the series of tests described in Section 3.2, multiple classification algorithms were evaluated in a comparative benchmark analysis.

Firstly, for Test 1, a simpler approach was taken, since its objective was to test and calibrate the setup. No pre-processing was used, and the raw data were fed directly to a 2-layer NN, with 20 and 10 neurons, respectively. The dataset was split into 80% for training and 20% for testing.

For Tests 2 through 9, eight different algorithms were selected from ML and DL alternatives for comparative evaluation. Two pre-processing options were used to format the input features: raw time-series data and a custom frequency-domain pipeline. This pipeline originated from a preliminary benchmark of reasoned pre-processing techniques, among which it showed the most promising results.

Raw time-series data is divided into 100 ms Comma-Separated Values (CSV) files consisting of a 2500 x 7 matrix, where the first column is the timestamp and the remaining columns represent the six-channel accelerometer data. Table 3.2 illustrates the structure of a single 100 ms window.

Table 3.2: Representation of a 100 ms raw sample data file.

Time (μs)	**Ch1_X**	**Ch1_Y**	**Ch1_Z**	**Ch2_X**	**Ch2_Y**	**Ch2_Z**
0	$x_{1,1}$	$y_{1,1}$	$z_{1,1}$	$x_{2,1}$	$y_{2,1}$	$z_{2,1}$
40	$x_{1,2}$	$y_{1,2}$	$z_{1,2}$	$x_{2,2}$	$y_{2,2}$	$z_{2,2}$
80	$x_{1,3}$	$y_{1,3}$	$z_{1,3}$	$x_{2,3}$	$y_{2,3}$	$z_{2,3}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
99960	$x_{1,2500}$	$y_{1,2500}$	$z_{1,2500}$	$x_{2,2500}$	$y_{2,2500}$	$z_{2,2500}$

For the frequency-domain pre-processing methodology, each raw 6-channel accelerometer sample is converted into a speed-invariant log-magnitude spectrum. First, the signal is angularly resampled. The number of samples is scaled by $\mathrm{RPM}_{\mathrm{actual}}/\mathrm{RPM}_{\mathrm{ref}}$ and then truncated back to 2500 points. This process aligns shaft rotational-speed harmonics to fixed frequency bins, regardless of small speed variations within the training set. A Hann window is applied before computing the FFT magnitude, and the result is log-compressed to reduce the dynamic range, yielding a 250×6 matrix covering the 10–2500 Hz spectral band.

Finally, a fixed-alpha speed correction is applied. A broadband ω^2 term is subtracted from all 250 bins to compensate for the centrifugal force increase with speed. Furthermore, an additional resonance term is subtracted strictly from the first 3 bins (representing the $1\times$ shaft-order region, 10–30 Hz), to compensate for the structural resonance that amplifies those specific bins. For the training data, the speed ratio is $\approx$1, so the correction is nearly zero. However, for the 1200 rpm test data, this operation rescales the amplitude back toward the 1000 rpm training distribution, ensuring extracted features remain comparable across both operating speeds. This architecture is illustrated in Figure 3.9. Table 3.3 outlines the structure of a single pre-processed 100 ms window. The programming code for this pre-processing method is presented in Appendix E.1.

Since this pre-processing pipeline was designed to increase the system's adaptability to the defined series of trials, it was used as the primary input across the benchmarked models; the raw time-series input is retained as a comparative baseline and is applied only when beneficial.

Table 3.3: Representation of a 100 ms pre-processed sample data file.

Frequency (Hz)	**Ch1_X**	**Ch1_Y**	**Ch1_Z**	**Ch2_X**	**Ch2_Y**	**Ch2_Z**
10.00	$\lvert X_{1,1}\rvert$	$\lvert Y_{1,1}\rvert$	$\lvert Z_{1,1}\rvert$	$\lvert X_{2,1}\rvert$	$\lvert Y_{2,1}\rvert$	$\lvert Z_{2,1}\rvert$
20.00	$\lvert X_{1,2}\rvert$	$\lvert Y_{1,2}\rvert$	$\lvert Z_{1,2}\rvert$	$\lvert X_{2,2}\rvert$	$\lvert Y_{2,2}\rvert$	$\lvert Z_{2,2}\rvert$
30.00	$\lvert X_{1,3}\rvert$	$\lvert Y_{1,3}\rvert$	$\lvert Z_{1,3}\rvert$	$\lvert X_{2,3}\rvert$	$\lvert Y_{2,3}\rvert$	$\lvert Z_{2,3}\rvert$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
2500.00	$\lvert X_{1,250}\rvert$	$\lvert Y_{1,250}\rvert$	$\lvert Z_{1,250}\rvert$	$\lvert X_{2,250}\rvert$	$\lvert Y_{2,250}\rvert$	$\lvert Z_{2,250}\rvert$

The selected AI algorithms, spanning classical ML (Random Forest, SVM, KNN, XGBoost), clustering (K-Means), and DL (CNN, LSTM, Dual-branch CNN) to cover the spectrum of common PdM approaches, are itemized and described as follows:

- **CNN**: The architecture employs a deep sequence of one-dimensional convolutional and pooling layers to extract spectral features from the dataset, as depicted in Figure 3.10. Compared to a basic single-layer convolutional network, this extended architecture incorporates successive pooling and regularization techniques that allow the model to capture long-term vibration dependencies, reducing the risk of overfitting. This algorithm also includes an early stopping callback to guarantee the best training epoch parameters are used in the final model. The programming code for this algorithm is presented in Appendix E.2.

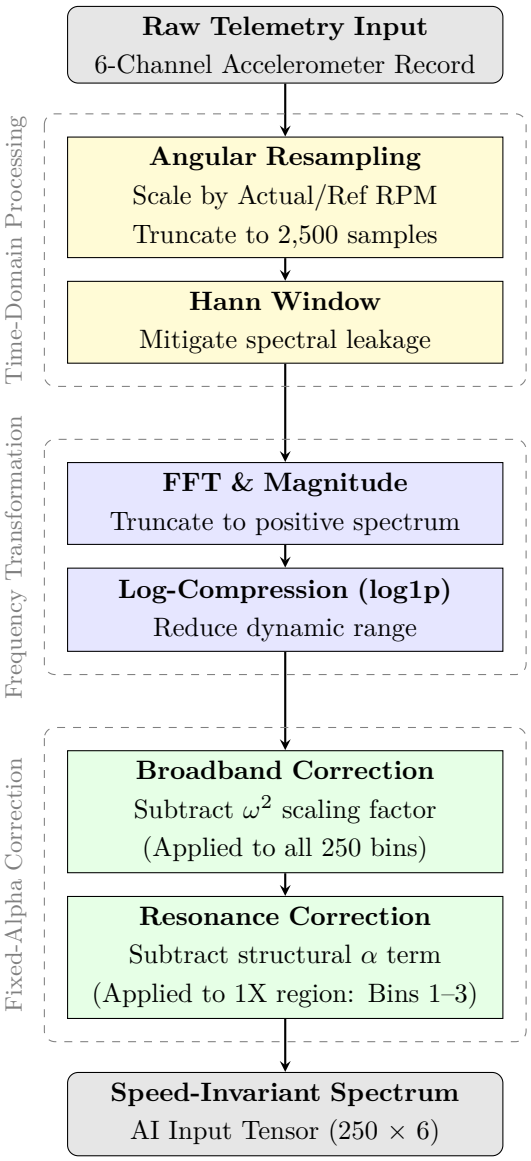


Figure 3.9: Data pre-processing pipeline for the conversion of raw time-series telemetry into a speed-invariant spectral matrix.

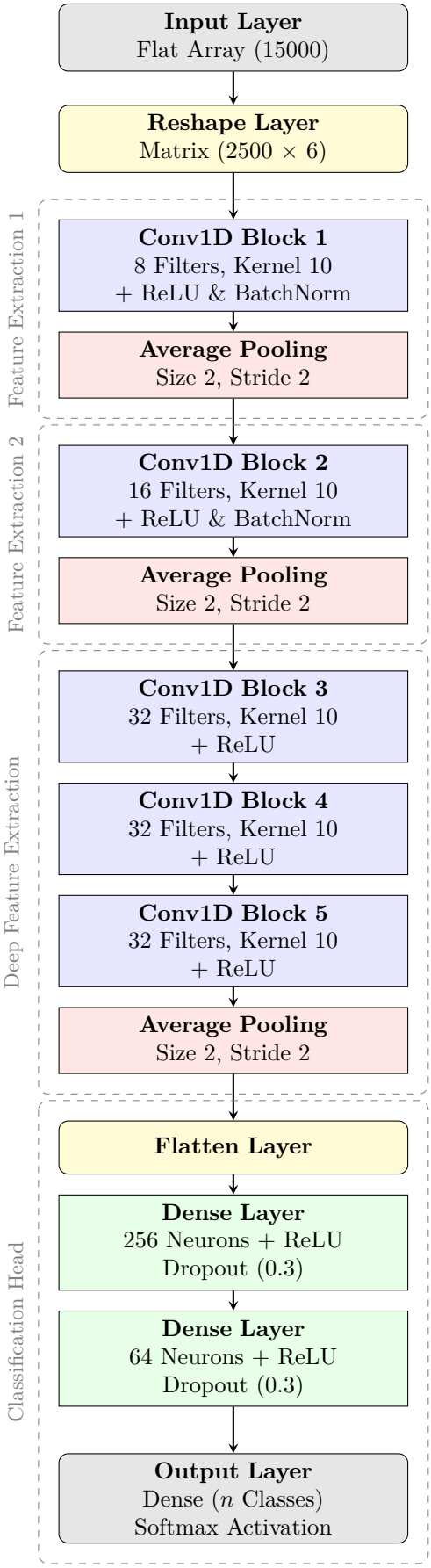


Figure 3.10: Architectural flowchart of the CNN utilized for anomaly classification.

- **Dual-branch CNN**: As depicted in Figure 3.11, this architecture uses a dual-branch design that can simultaneously evaluate distinct spectral regions separately. The left branch analyzes fundamental rotational harmonics by isolating the low-frequency components of the spectrum. The right branch runs a parallel deep convolutional pipeline that extracts high-order fault signatures. This architecture was designed to balance these two distinct spectral zones, preventing the more dominant low-order rotational harmonics from masking the subtler high-order fault signatures. This algorithm also includes an early stopping callback to guarantee the best training epoch parameters are used in the final model The complete programming code for this architecture is detailed in Appendix E.3.
- **LSTM**: As depicted in Figure 3.12, the architecture uses convolutional layers as a dimensionality reduction tool, followed by a sequence of recurrent networks. Compared to a standard recurrent neural network, this bidirectional design allows the model to capture historical vibration trends while bypassing the computational bottleneck of evaluating all 2500 samples directly. This algorithm also includes an early stopping callback to guarantee the best training epoch parameters are used in the final model. The programming code for this algorithm is provided in Appendix E.4.
- **K-Means**: A standard formulation of the K-Means clustering algorithm is adopted, with the hyperparameters and evaluation strategy defined in Table 3.4. The associated implementation is presented in Appendix E.9.

Table 3.4: Hyperparameter configuration and evaluation strategy for the K-Means clustering algorithm.

Parameter	**Value**
`n_clusters`	64
`n_init`	20
`random_state`	42
Pre-Processing	`StandardScaler` (zero mean, unit variance)
Features	1500 (full spectrum)
Cluster labeling	Majority class vote on training labels ($> 50\%$ fault $\rightarrow$ fault)

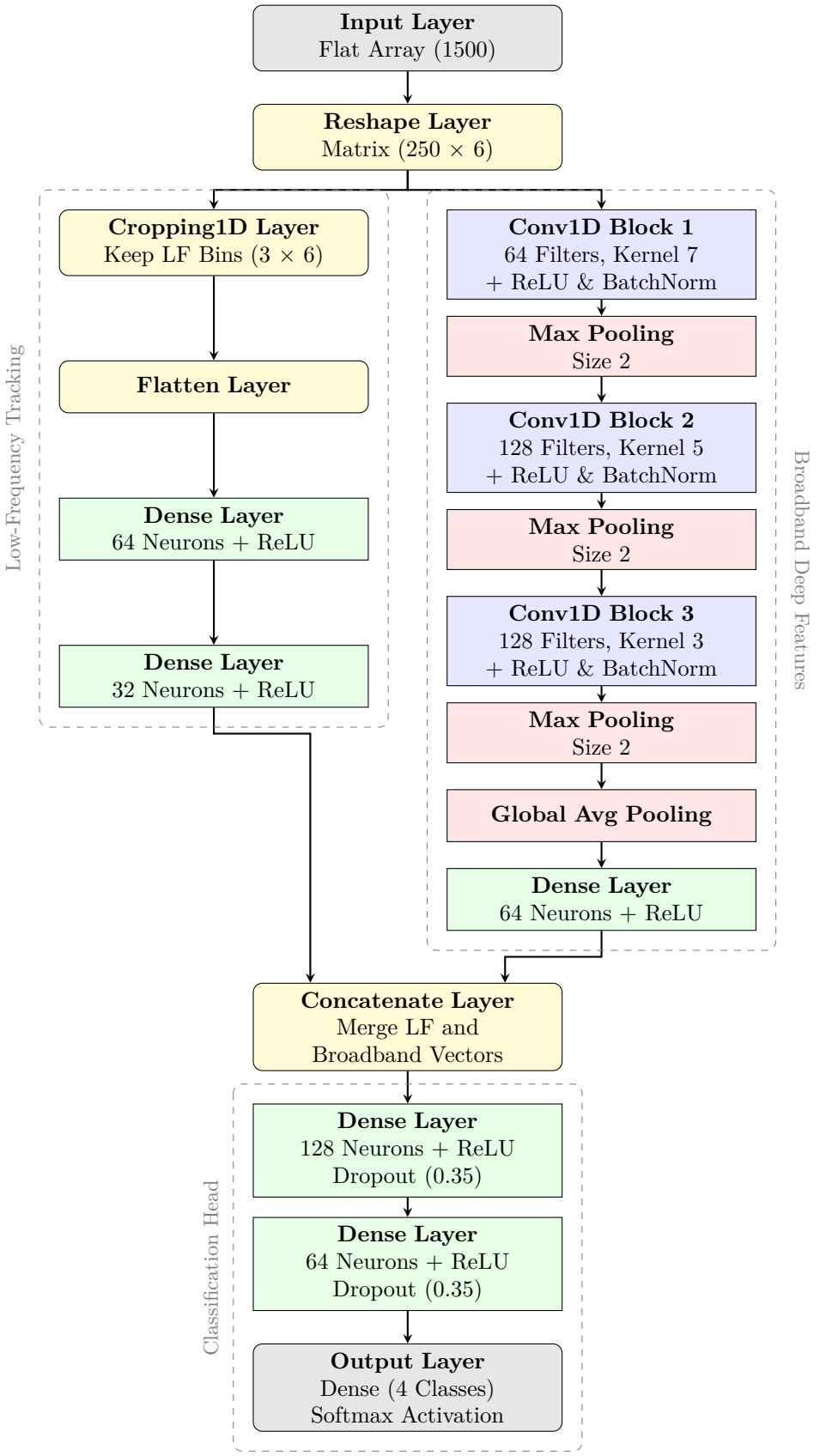


Figure 3.11: Architectural flowchart of the Dual-branch CNN framework.

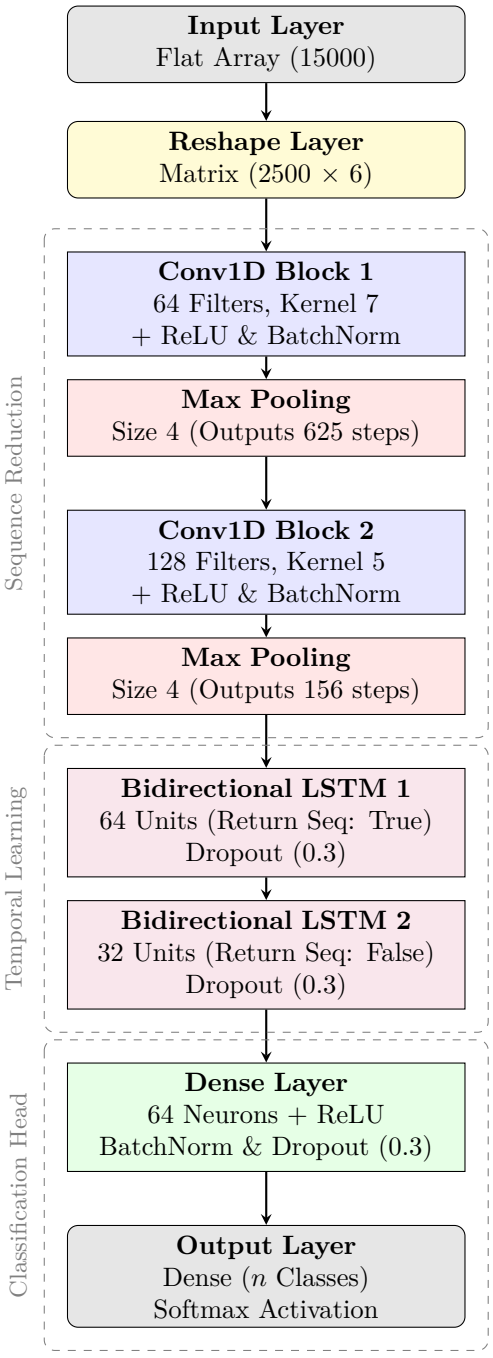


Figure 3.12: Architectural flowchart of the LSTM framework.

- **Random Forest**: This study employs a standard Random Forest classifier, optimized with the hyperparameters presented in Table 3.5. The corresponding implementation is provided in Appendix E.5.

Table 3.5: Parameter configuration for the Random Forest classifier.

Parameter	**Value**
`n_estimators`	500
`max_features`	`sqrt` ($\sqrt{1500} \approx 39$)
`max_depth`	None
`n_jobs`	-1
`random_state`	42

- **Support Vector Machines**: A conventional SVM architecture was adopted as the baseline classifier, parameterized according to the optimized configuration depicted in Table 3.6. The full source code for this classifier can be found in Appendix E.6.

Table 3.6: Parameter configuration for the SVM classifier.

Parameter	**Value**
`kernel`	`rbf`
`C`	10
`gamma`	`scale` $\left(\frac{1}{1500 \times \text{var}}\right)$
Pre-Processing	`StandardScaler` (zero mean, unit variance)
Features	1500 (full spectrum, no PCA)

- **K-Nearest Neighbors**: A standard KNN model was implemented with its optimized parameters represented in Table 3.7. The implementation details are listed in Appendix E.7.

Table 3.7: Hyperparameter configuration for the KNN classifier.

Parameter	**Value**
`n_neighbors`	5
`metric`	`euclidean`
`n_jobs`	-1
Pre-Processing	`StandardScaler` (zero mean, unit variance)
Features	1500 (full spectrum, no dimensionality reduction)
Fit time	$\sim$0 s (lazy learning)

- **eXtreme Gradient Boosting**: A standard XGBoost classifier configuration was employed, with the optimized parameters delineated in Table 3.8. The programming code for this model is available in Appendix E.8.

Table 3.8: Hyperparameter configuration for the XGBoost classifier.

Parameter	**Value**
`n_estimators`	500
`max_depth`	6
`learning_rate`	0.1
`subsample`	0.8
`colsample_bytree`	0.8
`objective`	`multi:softmax`
`num_class`	4
`n_jobs`	-1
`random_state`	42

To enable real-time edge inference within the MMS, the trained Dual-branch CNN model was exported to the Open Neural Network Exchange (ONNX) format and deployed directly on the IPC. The pre-processing pipeline described above was implemented in TwinCAT using the TwinCAT Analytics `TF3600` library, which provides the `FB_CMA_MagnitudeSpectrum` function block for real-time FFT computation. The angular resampling, Hann windowing, log-compression, and fixed-alpha speed correction steps are executed within a single PLC task cycle upon completion of each 2500-sample window, creating the 1500-feature input vector. Model inference is then performed using the `TF3800` library's `FB_TF3800_ONNX` function block, which loads the ONNX model at startup and executes inference on each new feature vector. The predicted class and softmax confidence scores are written to global variables, making the diagnostic result available for real-time monitoring. The TwinCAT code for this implementation is provided in Appendices C.6 and C.7.

3.5 Conclusion

In summary, a methodology was developed to evaluate the system for PdM applications. An experimental workbench was devised to simulate diverse imbalance scenarios, which were used to benchmark eight distinct AI models. The test sequence was composed of fourteen trials of increasing complexity to evaluate simultaneously the system's ability to identify specific fault conditions and to explore the system's limits by testing unseen faults, cross-domain, and cross-speed scenarios, therefore evaluating operational conditions that remain under-researched in the current literature.

Chapter 4

Results and Discussion

This chapter presents the results obtained through the methods described in the previous chapter. Furthermore, a thorough examination and discussion of the presented findings are included.

4.1 Results

In this section, the results for the pre-established tests in Section 3.2 are presented. For each test, all AI models are compared using two evaluation metrics: overall accuracy and macro F1-score. The inclusion of F1-score validates the accuracy result by taking into account potential class imbalance, as it weights each class equally regardless of sample count. Additionally, to assess the viability of real-time deployment of the proposed models, the per-sample inference time for each benchmarked method is also provided. For the best-performing model in each test, a confusion matrix is presented to illustrate the misclassification of individual faults and their separability. For the cross-speed and domain transfer tests, where generalization is the main challenge, the confusion matrix of the worst-performing model is also included to highlight the nature of the failure patterns.

4.1.1 Test 1 Results: Hammer impact detection

Test 1 was devised as an initial calibration and validation trial for the MMS, in which a binary classification approach was adopted for a small dataset. Table 4.1 reports the accuracy, macro F1-score, and per-sample inference time of the selected model. Since this test achieved 100% accuracy, the confusion matrix is omitted, as it provides no additional diagnostic information beyond the reported metrics.

Table 4.1: Test 1 results: Hammer impact detection.

Model	**Accuracy (%)**	**Macro F1**	**Inference Time (ms)**
Neural Network	100.0	1.00	1.00

4.1.2 Test 2 Results: Anomaly detection binary classification

A summary of the tests performed in *Rig 1* is presented in Table 4.2, with Test 2 highlighted in bold. Table 4.3 reports the accuracy, macro F1 score, and per-sample inference time of each model. The confusion matrix of the best-performing model is presented in Figure 4.1.

Table 4.2: Configuration of the Rig 1 tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
2	**Binary**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1000 rpm**
3.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
3.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
4	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
5	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.3: Test 2 results: Anomaly detection binary classification.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
CNN	100.00	1.0000	0.85
LSTM	100.00	1.0000	2.90
Dual-branch CNN	99.17	0.9890	0.76
SVM	99.17	0.9889	0.37
XGBoost	99.17	0.9889	0.43
Random Forest	97.93	0.9718	39.21
KNN	95.04	0.9342	16.44
K-Means	90.50	0.8819	3.80

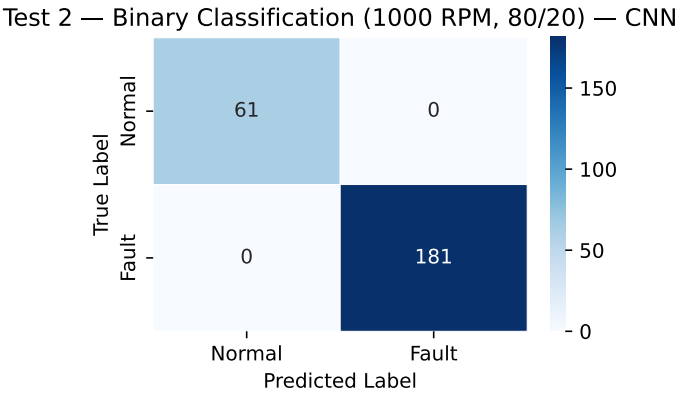


Figure 4.1: Confusion matrix for the best-performing model (CNN) in Test 2.

4.1.3 Test 3 Results: Anomaly detection binary classification with untrained anomaly class

4.1.3.1 Test 3.1: Anomaly detection binary classification with Imbalance 3 as untrained anomaly class

A summary of the tests performed in *Rig 1* is presented in Table 4.4, with Test 3.1 highlighted in bold. The F1-score is omitted in this trial, as this metric is meaningless in a single-class test. The accuracy and per-sample inference time of each model are reported in Table 4.5. The confusion matrix of the best-performing model is presented in Figure 4.2.

Table 4.4: Configuration of the Rig 1 tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
2	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
3.1	**Binary**	**Normal, Imb. 1, 2, 4**	**Imb. 3**	**1000 rpm**	**1000 rpm**
3.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
4	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
5	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.5: Test 3.1 results: anomaly detection binary classification with Imbalance 3 as untrained anomaly class.

Model	Accuracy (%)	Inference (ms/sample)
Dual-branch CNN	100.00	1.02
CNN	100.00	1.05
LSTM	100.00	1.84
Random Forest	100.00	40.11
SVM	100.00	0.39
XGBoost	100.00	0.41
KNN	97.35	17.73
K-Means	95.03	3.77

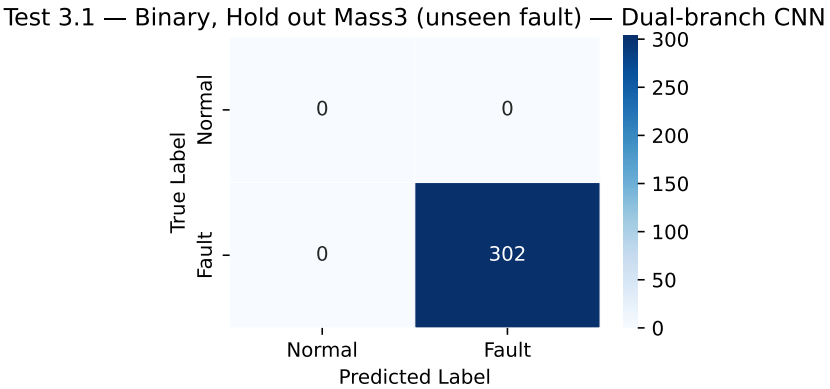


Figure 4.2: Confusion matrix for the best-performing model (Dual-CNN) in Test 3.1.

4.1.3.2 Test 3.2 Results: Anomaly detection binary classification with Imbalance 4 as untrained anomaly class

A summary of the tests performed in *Rig 1* is presented in Table 4.6, with Test 3.2 highlighted in bold. The F1-score is omitted in this trial, as this metric is meaningless in a single-class test. The accuracy and per-sample inference time of each model are reported in Table 4.7. The confusion matrix of the best-performing model is presented in Figure 4.3.

Table 4.6: Configuration of the Rig 1 tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
2	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
3.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
3.2	**Binary**	**Normal, Imb. 1–3**	**Imb. 4**	**1000 rpm**	**1000 rpm**
4	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
5	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.7: Test 3.2 results: Anomaly detection binary classification with Imbalance 4 as untrained anomaly class.

Model	Accuracy (%)	Inference (ms/sample)
Dual-branch CNN	100.00	0.61
Random Forest	100.00	39.78
XGBoost	99.68	0.43
CNN	98.71	0.70
SVM	97.74	0.38
KNN	80.97	17.05
LSTM	80.32	2.46
K-Means	53.87	3.83

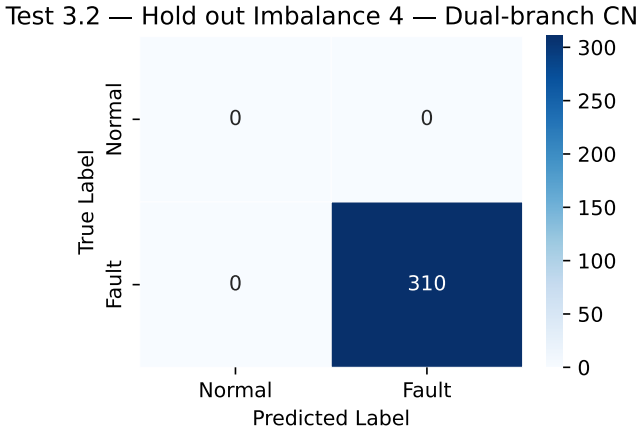


Figure 4.3: Confusion matrix for the best-performing model (Dual-CNN) in Test 3.2.

4.1.4 Test 4 Results: Anomaly classification

A summary of the tests performed in *Rig 1* is presented in Table 4.8, with Test 4 highlighted in bold. The accuracy, macro F1 score, and per-sample inference time of each model are reported in Table 4.9. The confusion matrix of the best-performing model is presented in Figure 4.4.

Table 4.8: Configuration of the Rig 1 tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
2	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
3.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
3.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
4	**4-class**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1000 rpm**
5	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.9: Test 4 results: Anomaly classification.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
LSTM	100.00	1.0000	2.86
XGBoost	100.00	1.0000	0.57
CNN	99.59	0.9959	0.83
Random Forest	99.17	0.9917	40.10
SVM	99.17	0.9917	0.53
Dual-branch CNN	98.76	0.9876	0.75
KNN	94.21	0.9428	17.00
K-Means	83.47	0.8341	3.78

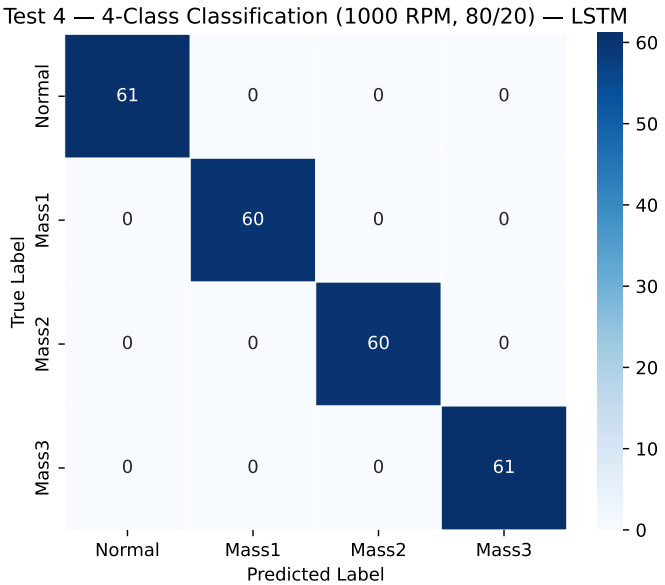


Figure 4.4: Confusion matrix for the best-performing model (LSTM) in Test 4.

4.1.5 Test 5 Results: Anomaly classification with alternative speed dataset testing

A summary of the tests performed in *Rig 1* is presented in Table 4.10, with Test 5 highlighted in bold. Results are reported for both the four-class classification task (Table 4.11) and a binary normal-versus-fault formulation (Table 4.12). The binary results are included to show that, even where a model fails to identify the specific fault under the speed shift, it may still reliably discriminate the normal condition from a faulty one. The confusion matrices of the best- and worst-performing models in the four-class task are presented in Figures 4.5 and 4.6, respectively.

Table 4.10: Configuration of the Rig 1 tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
2	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
3.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
3.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
4	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
5	**4-class**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1200 rpm**

Table 4.11: Test 5 Results: Anomaly classification with alternative speed dataset testing.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
Dual-branch CNN	70.17	0.6293	0.64
LSTM	55.17	0.5353	1.49
K-Means	41.10	0.3573	3.79
KNN	38.14	0.3093	19.74
SVM	29.41	0.1790	1.25
CNN	27.46	0.2733	0.35
Random Forest	27.20	0.1681	40.07
XGBoost	22.37	0.1798	0.59

Table 4.12: Test 5 results: binary normal-versus-fault classification under the alternative speed dataset.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
Dual-branch CNN	99.41	0.9920	0.64
CNN	84.41	0.7208	0.35
Random Forest	83.98	0.8162	37.97
LSTM	80.51	0.6453	1.49
SVM	79.41	0.7403	0.56
XGBoost	78.31	0.7234	0.40
K-Means	73.73	0.7116	2.54
KNN	58.81	0.5823	24.29

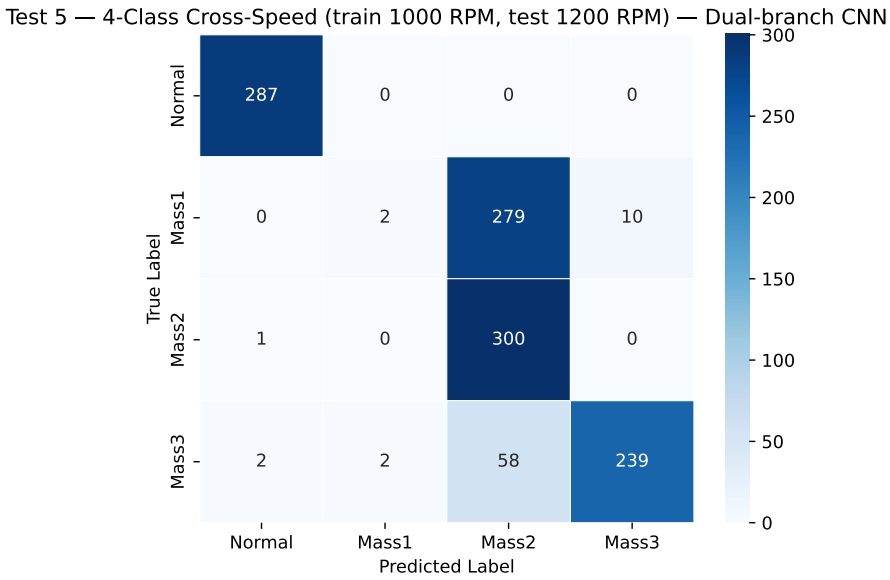


Figure 4.5: Confusion matrix for the best-performing model (Dual-CNN) in Test 5.

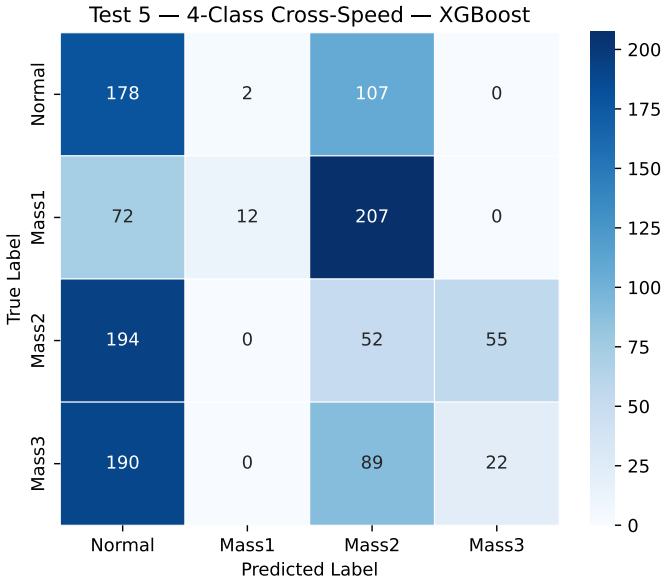


Figure 4.6: Confusion matrix for the worst-performing model (XGBoost) in Test 5.

4.1.6 Test 6 Results: Anomaly detection in static state

A summary of the tests performed in *Rig 2*, in static state, is presented in Table 4.13, with Test 6 highlighted in bold. The accuracy, macro F1 score, and per-sample inference time of each model are reported in Table 4.14. The confusion matrix of the best-performing model is presented in Figure 4.7.

Table 4.13: Configuration of the Rig 2 static-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
6	**Binary**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1000 rpm**
7.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
7.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
8	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
9	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.14: Test 6 results: Anomaly detection in static state.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
LSTM	100.00	1.0000	3.04
Dual-branch CNN	99.59	0.9944	0.78
SVM	99.59	0.9944	0.48
CNN	99.17	0.9888	0.85
Random Forest	99.17	0.9888	39.78
XGBoost	99.17	0.9888	0.42
KNN	97.52	0.9660	19.51
K-Means	94.21	0.9224	3.78

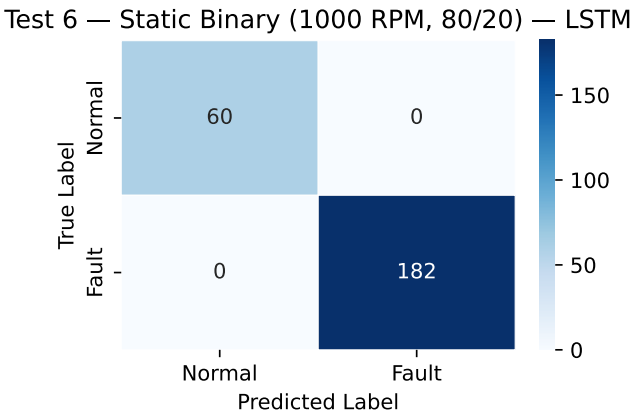


Figure 4.7: Confusion matrix for the best-performing model (LSTM) in Test 6.

4.1.7 Test 7 Results: Anomaly detection binary classification with untrained anomaly class

4.1.7.1 Test 7.1 results: Anomaly detection in static state with untrained anomaly class

A summary of the tests performed in *Rig 2*, in static state, is presented in Table 4.15, with Test 7.1 highlighted in bold. The F1-score is omitted in this trial, as this metric is meaningless in a single-class test. The accuracy and per-sample inference time of each model are reported in Table 4.16. The confusion matrix of the best-performing model is presented in Figure 4.8.

Table 4.15: Configuration of the Rig 2 static-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
6	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
7.1	**Binary**	**Normal, Imb. 1, 2, 4**	**Imb. 3**	**1000 rpm**	**1000 rpm**
7.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
8	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
9	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.16: Test 7.1 Results: Anomaly detection in static state with Imbalance 3 as untrained anomaly class.

Model	Accuracy (%)	Inference (ms/sample)
Dual-branch CNN	100.00	0.65
CNN	100.00	0.74
LSTM	100.00	2.54
Random Forest	100.00	39.68
XGBoost	100.00	0.43
KNN	91.00	19.09
K-Means	86.00	3.79
SVM	81.67	0.42

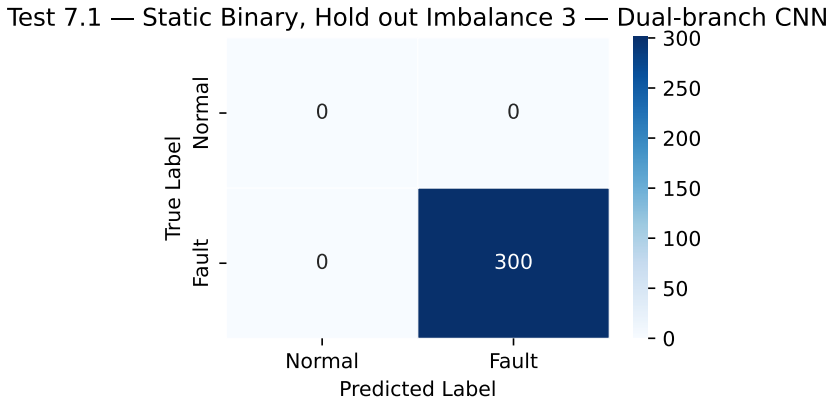


Figure 4.8: Confusion matrix for the best-performing model (Dual-CNN) in Test 7.1.

4.1.7.2 Test 7.2 results: Anomaly detection in static state with untrained anomaly class

A summary of the tests performed in *Rig 2*, in static state, is presented in Table 4.17, with Test 7.2 highlighted in bold. The F1-score is omitted in this trial, as this metric is meaningless in a single-class test. The accuracy and per-sample inference time of each model are reported in Table 4.18. The confusion matrix of the best-performing model is presented in Figure 4.9.

Table 4.17: Configuration of the Rig 2 static-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
6	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
7.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
7.2	**Binary**	**Normal, Imb. 1–3**	**Imb. 4**	**1000 rpm**	**1000 rpm**
8	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
9	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.18: Test 7.2 Results: Anomaly detection in static state with Imbalance 4 as untrained anomaly class.

Model	Accuracy (%)	Inference (ms/sample)
Dual-branch CNN	100.00	0.72
CNN	100.00	0.75
LSTM	100.00	2.85
Random Forest	100.00	50.80
SVM	100.00	0.63
KNN	100.00	21.32
XGBoost	100.00	0.45
K-Means	98.68	3.97

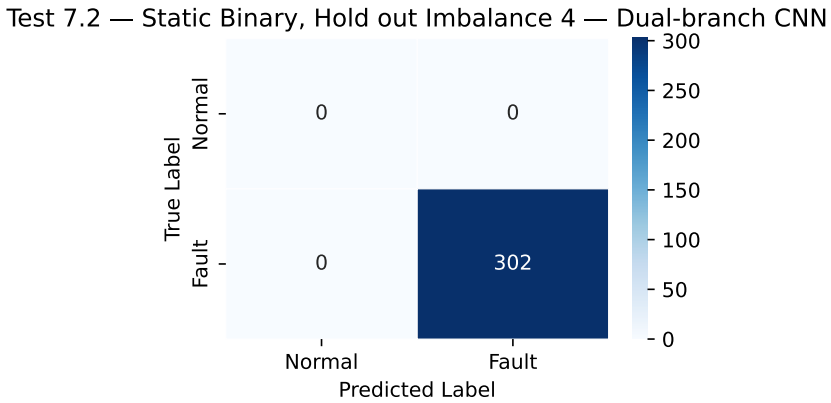


Figure 4.9: Confusion matrix for the best-performing model (Dual-CNN) in Test 7.2.

4.1.8 Test 8 Results: Anomaly classification in static state

A summary of the tests performed in *Rig 2*, in static state, is presented in Table 4.19, with Test 8 highlighted in bold. The accuracy, macro F1 score, and per-sample inference time of each model are reported in Table 4.20. The confusion matrix of the best-performing model is presented in Figure 4.10.

Table 4.19: Configuration of the Rig 2 static-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
6	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
7.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
7.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
8	**4-class**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1000 rpm**
9	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.20: Test 8 Results: Anomaly classification in static state.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
XGBoost	100.00	1.0000	0.017
Dual-branch CNN	97.93	0.9793	0.762
Random Forest	97.52	0.9752	0.289
SVM	96.69	0.9670	0.715
LSTM	95.04	0.9506	2.464
CNN	88.02	0.8794	0.691
KNN	87.19	0.8712	1.153
K-Means	74.79	0.7447	0.014

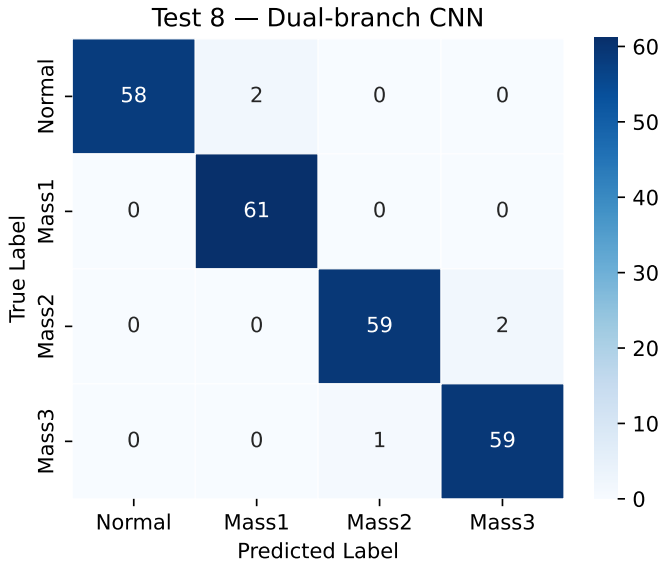


Figure 4.10: Confusion matrix for the best-performing model (Dual-CNN) in Test 8.

4.1.9 Test 9 Results: Anomaly classification in static state with alternative speed dataset testing

A summary of the tests performed in *Rig 2*, in static state, is presented in Table 4.21, with Test 9 highlighted in bold. Results are reported for both the four-class classification task (Table 4.22) and a binary normal-versus-fault formulation (Table 4.23). The binary results are included to show that, even where a model fails to identify the specific fault under the speed shift, it may still reliably discriminate the normal condition from a faulty one. The confusion matrices of the best- and worst-performing models in the four-class task are presented in Figures 4.11 and 4.12, respectively.

Table 4.21: Configuration of the Rig 2 static-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
6	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
7.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
7.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
8	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
9	**4-class**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1200 rpm**

Table 4.22: Test 9 results: Anomaly classification in static state with alternative speed dataset testing (four-class).

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
Dual-branch CNN	91.90	0.9189	0.227
LSTM	77.85	0.7259	1.694
CNN	65.45	0.6339	0.691
Random Forest	63.31	0.6298	0.059
XGBoost	57.77	0.4692	0.007
KNN	36.69	0.3846	0.079
K-Means	32.31	0.2121	0.569
SVM	31.49	0.3237	0.004

Table 4.23: Test 9 results: binary normal-versus-fault classification under the alternative speed dataset.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
Dual-branch CNN	100.00	1.0000	0.227
Random Forest	100.00	1.0000	0.060
SVM	99.83	0.9978	0.569
XGBoost	99.83	0.9978	0.006
LSTM	99.09	0.9880	2.310
CNN	95.29	0.9408	0.691
KNN	87.19	0.7888	0.271
K-Means	81.40	0.6496	0.005

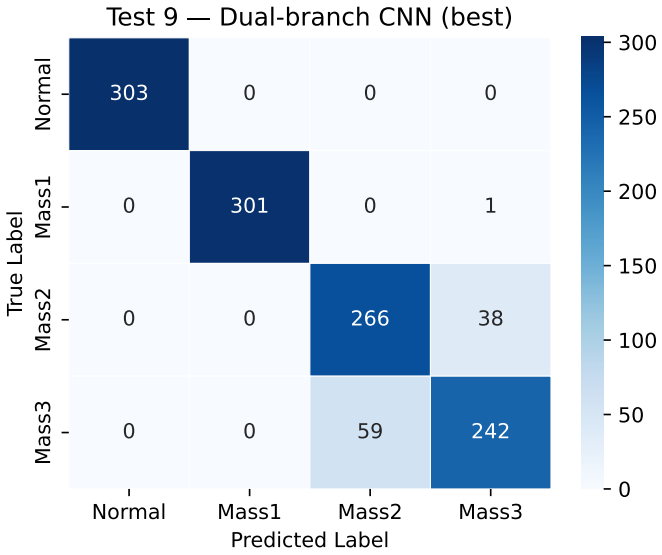


Figure 4.11: Confusion matrix for the best-performing model (Dual-CNN) in Test 9.

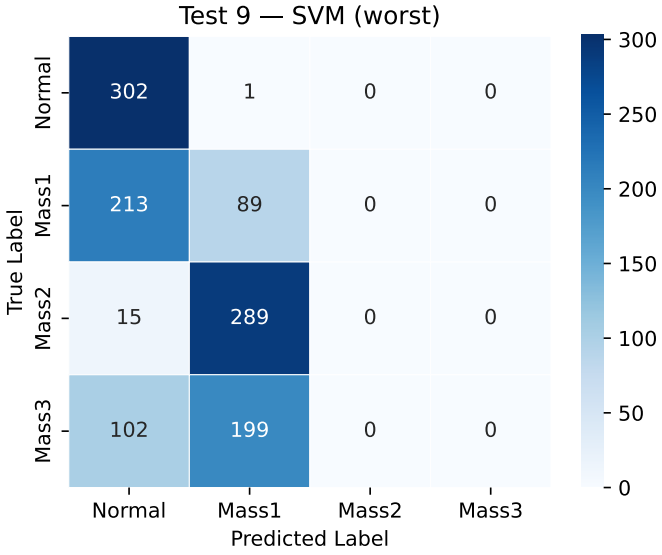


Figure 4.12: Confusion matrix for the worst-performing model (SVM) in Test 9.

4.1.10 Test 10 Results: Anomaly detection in dynamic state

A summary of the tests performed in *Rig 2*, in dynamic state, is presented in Table 4.24, with Test 10 highlighted in bold. The accuracy, macro F1 score, and per-sample inference time of each model are reported in Table 4.25. The confusion matrix of the best-performing model is presented in Figure 4.13.

Table 4.24: Configuration of the Rig 2 dynamic-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
10	**Binary**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1000 rpm**
11.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
11.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
12	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
13	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.25: Test 10 results: Anomaly detection in dynamic state.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
XGBoost	99.18	0.9891	0.43
Dual-branch CNN	98.35	0.9786	0.77
Random Forest	96.71	0.9547	39.81
LSTM	85.19	0.8030	2.86
SVM	82.72	0.7136	0.76
CNN	74.49	0.4269	0.87
KNN	71.19	0.5989	19.94
K-Means	69.96	0.4979	3.73

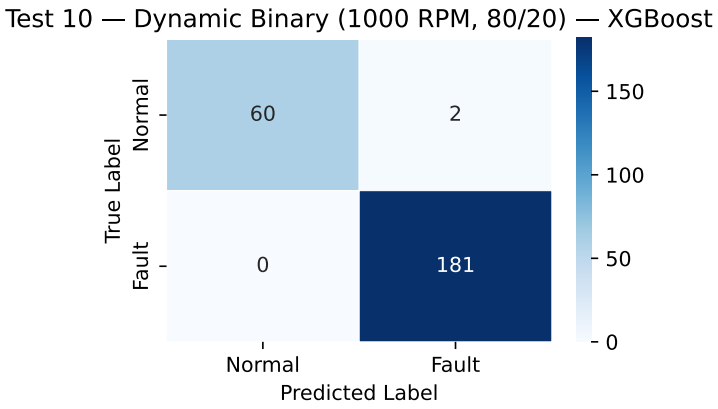


Figure 4.13: Confusion matrix for the best-performing model (XGBoost) in Test 10.

4.1.11 Test 11 Results: Anomaly detection binary classification with untrained anomaly class

4.1.11.1 Test 11.1 Results: Anomaly detection in dynamic state with untrained anomaly class

A summary of the tests performed in *Rig 2*, in dynamic state, is presented in Table 4.26, with Test 11.1 highlighted in bold. The F1-score is omitted in this Test, as this metric is meaningless in a single-class test. The accuracy and per-sample inference time of each model are reported in Table 4.27. The confusion matrix of the best-performing model is presented in Figure 4.14.

Table 4.26: Configuration of the Rig 2 dynamic-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
10	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
11.1	**Binary**	**Normal, Imb. 1, 2, 4**	**Imb. 3**	**1000 rpm**	**1000 rpm**
11.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
12	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
13	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.27: Test 11.1 results: Anomaly detection in dynamic state with Imbalance 3 as untrained anomaly class.

Model	Accuracy (%)	Inference (ms/sample)
Random Forest	100.00	39.71
XGBoost	100.00	0.46
Dual-branch CNN	98.34	0.66
SVM	98.01	1.17
LSTM	96.01	2.58
K-Means	95.35	3.80
CNN	95.02	0.74
KNN	79.40	20.29

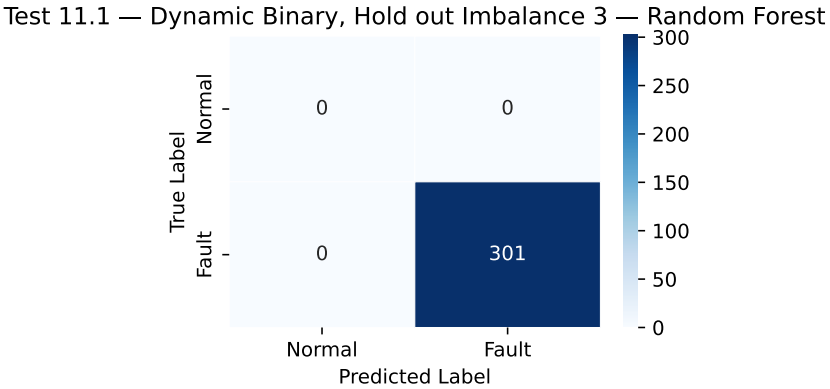


Figure 4.14: Confusion matrix for the best-performing model (RF) in Test 11.1.

4.1.11.2 Test 11.2 Results: Anomaly detection in dynamic state with untrained anomaly class

A summary of the tests performed in *Rig 2*, in dynamic state, is presented in Table 4.28, with Test 11.2 highlighted in bold. The F1-score is omitted in this Test, as this metric is meaningless in a single-class test. The accuracy and per-sample inference time of each model are reported in Table 4.29. The confusion matrix of the best-performing model is presented in Figure 4.15.

Table 4.28: Configuration of the Rig 2 dynamic-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
10	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
11.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
11.2	**Binary**	**Normal, Imb. 1–3**	**Imb. 4**	**1000 rpm**	**1000 rpm**
12	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
13	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.29: Test 11.2 results: Anomaly detection in dynamic state with Imbalance 4 as untrained anomaly class.

Model	Accuracy (%)	Inference (ms/sample)
LSTM	100.00	2.48
Random Forest	100.00	39.39
XGBoost	99.67	0.43
Dual-branch CNN	99.34	0.67
SVM	98.68	0.73
CNN	98.01	0.73
K-Means	95.03	3.74
KNN	86.09	23.80

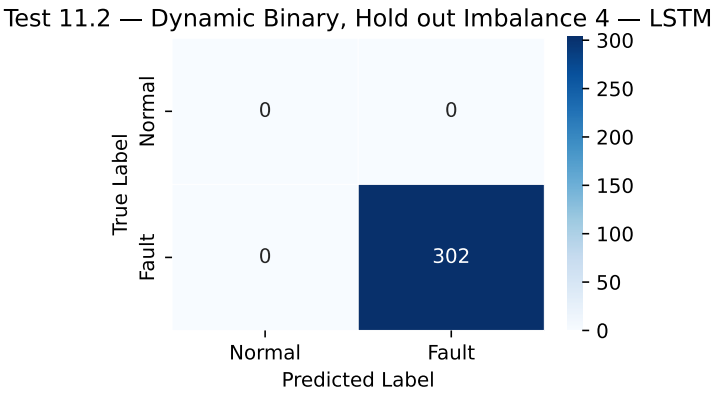


Figure 4.15: Confusion matrix for the best-performing model (LSTM) in Test 11.2.

4.1.12 Test 12 Results: Anomaly classification in dynamic state

A summary of the tests performed in *Rig 2*, in dynamic state, is presented in Table 4.30, with Test 12 highlighted in bold. For the CNN model, both frequency-domain (CNN original) and raw time-domain (CNN raw) inputs are stated to assess the pre-processing pipeline's effect under dynamic operation. The accuracy, macro F1 score, and per-sample inference time of each model are reported in Table 4.31. The confusion matrix of the best-performing model is presented in Figure 4.16.

Table 4.30: Configuration of the Rig 2 dynamic-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
10	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
11.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
11.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
12	**4-class**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1000 rpm**
13	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1200 rpm

Table 4.31: Test 12 results: Anomaly classification in dynamic state.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
CNN raw	99.59	0.9959	1.609
XGBoost	81.89	0.8181	0.040
LSTM	79.42	0.7948	5.207
Dual-branch CNN	79.01	0.7911	0.762
Random Forest	76.54	0.7732	0.561
SVM	69.96	0.7082	1.378
CNN original	61.73	0.6162	1.807
KNN	42.80	0.4144	1.740
K-Means	31.69	0.3160	0.017

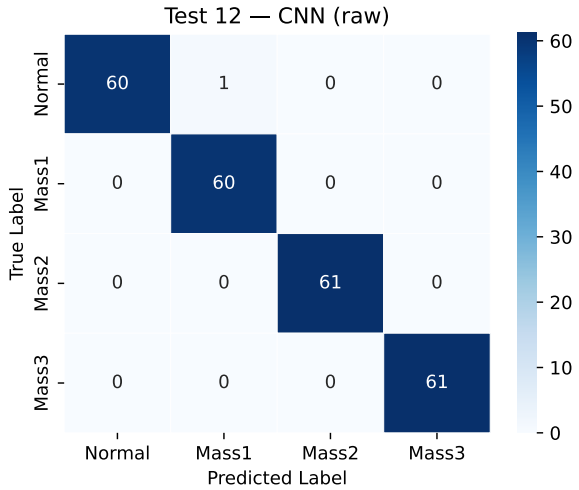


Figure 4.16: Confusion matrix for the best-performing model (CNN raw) in Test 12.

4.1.13 Test 13 Results: Anomaly classification in dynamic state with alternative speed dataset testing

A summary of the tests performed in *Rig 2*, in dynamic state, is presented in Table 4.32, with Test 13 highlighted in bold. Results are reported for both the four-class classification task (Table 4.33) and a binary normal-versus-fault formulation (Table 4.34). The binary results are included to show that, even where a model fails to identify the specific fault under the speed shift, it may still reliably distinguish the normal condition from a faulty one. The confusion matrices of the best- and worst-performing models in the four-class task are presented in Figures 4.17 and 4.18, respectively.

Table 4.32: Configuration of the Rig 2 dynamic-state tests.

Test	Classification	Training classes	Testing classes	Train speed	Test speed
10	Binary	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
11.1	Binary	Normal, Imb. 1, 2, 4	Imb. 3	1000 rpm	1000 rpm
11.2	Binary	Normal, Imb. 1–3	Imb. 4	1000 rpm	1000 rpm
12	4-class	Normal, Imb. 1–3	Normal, Imb. 1–3	1000 rpm	1000 rpm
13	**4-class**	**Normal, Imb. 1–3**	**Normal, Imb. 1–3**	**1000 rpm**	**1200 rpm**

Table 4.33: Test 13 results: Anomaly classification in dynamic state with alternative speed dataset testing (four-class).

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
XGBoost	62.45	0.5545	0.025
Dual-branch CNN	60.23	0.5611	0.86
LSTM	57.70	0.5051	2.535
Random Forest	56.95	0.5316	0.069
CNN original	52.46	0.5218	0.667
SVM	40.97	0.3823	1.859
KNN	32.39	0.3102	0.506
K-Means	23.93	0.2009	0.009

Table 4.34: Test 13 results: binary normal-versus-fault classification under the alternative speed dataset.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
Random Forest	100.00	1.0000	0.069
XGBoost	100.00	1.0000	0.025
Dual-branch CNN	100.00	1.0000	0.86
LSTM	99.58	0.9944	2.535
CNN original	95.34	0.9356	0.667
SVM	91.01	0.8594	1.859
KNN	77.60	0.6328	0.506
K-Means	72.94	0.4746	0.009

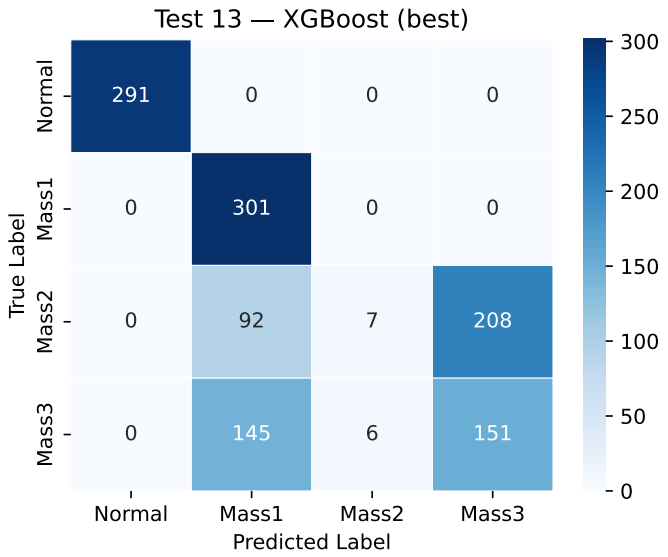


Figure 4.17: Confusion matrix for the best-performing model (XGBoost) in Test 13.

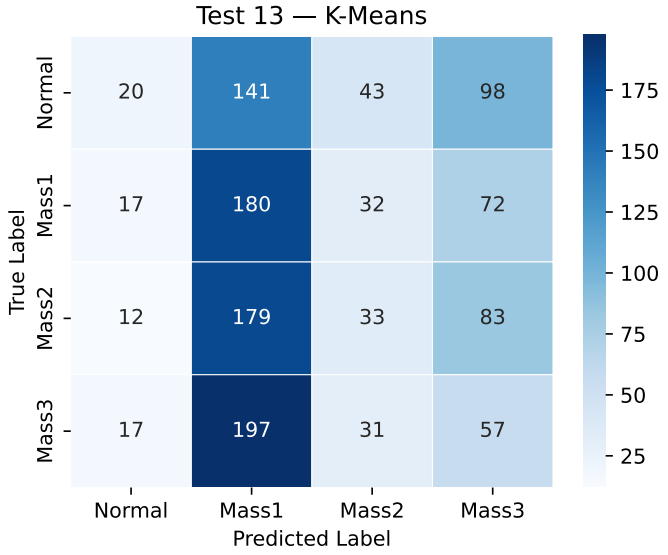


Figure 4.18: Confusion matrix for the worst-performing model (K-Means) in Test 13.

4.1.14 Test 14 Results: Anomaly classification with static-state training and dynamic-state testing

Test 14 evaluates cross-domain generalization on Rig 2 at a fixed speed of 1000 rpm. A summary of this test is presented in Table 4.35. Results are reported for both the four-class classification task (Table 4.36) and a binary normal-versus-fault formulation (Table 4.37). The binary results are included to show that even when a model fails to identify the specific fault across this domain shift, it may still reliably distinguish the normal condition from a faulty one. The confusion matrices of the best- and worst-performing models are presented in Figures 4.19 and 4.20, respectively.

Table 4.35: Configuration of the Rig 2 cross-state test (Test 14).

Test	Classification	Classes	Training state	Testing state
14	**4-class**	**Normal, Imb. 1–3**	**Static**	**Dynamic**

Table 4.36: Test 14 results: Anomaly classification with static-state training and dynamic-state testing.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
XGBoost	53.42	0.5088	0.018
Random Forest	42.46	0.3872	0.108
Dual-branch CNN	34.95	0.2718	0.233
LSTM	33.39	0.2672	3.098
K-Means	29.43	0.2098	0.004
KNN	27.78	0.2168	0.392
CNN original	25.64	0.1314	0.513
SVM	25.56	0.1307	1.971

Table 4.37: Test 14 results: binary normal-versus-fault classification with static-state training and dynamic-state testing.

Model	Accuracy (%)	Macro F1	Inference (ms/sample)
XGBoost	95.38	0.9342	0.018
Random Forest	89.53	0.8343	0.108
LSTM	80.38	0.6171	3.098
Dual-branch CNN	77.25	0.5135	0.233
K-Means	76.26	0.6071	0.004
CNN original	75.43	0.4428	0.513
KNN	70.07	0.5802	0.392
SVM	28.36	0.2554	1.971

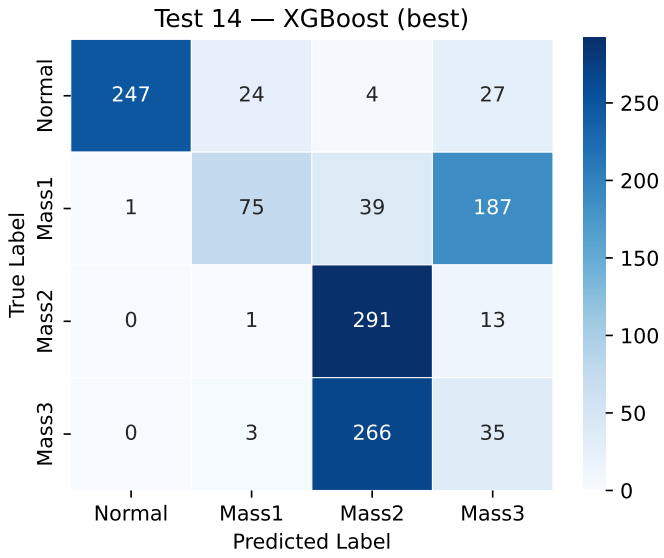


Figure 4.19: Confusion matrix for the best-performing model (XGBoost) in Test 14.

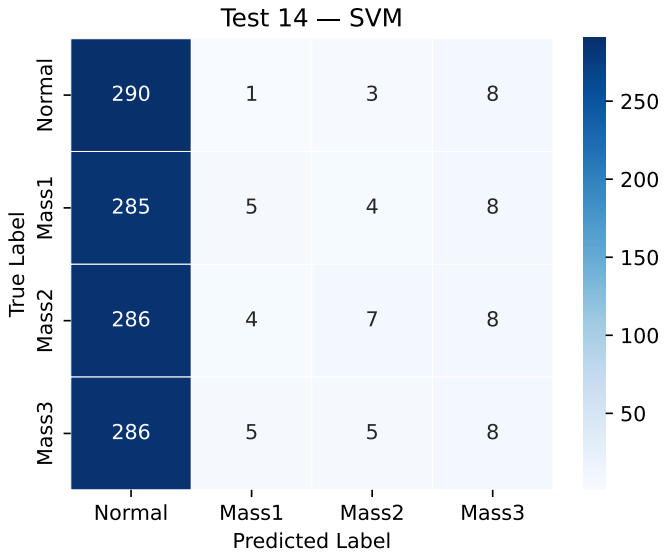


Figure 4.20: Confusion matrix for the worst-performing model (SVM) in Test 14.

4.1.15 Summary of Results

An overview of the models' outcomes for the entire test series is provided in Table 4.38, which compares the performance of the three most consistent models: Dual-branch CNN, XGBoost and LSTM. These algorithms were chosen because they achieved the highest reliability across static, dynamic, and cross-speed scenarios.

Table 4.38: Accuracy (%) of the three best-performing models across all tests.

Test	Description	Dual-branch CNN	XGBoost	LSTM
Rig 1 — Static State				
2	Binary, static	99.2	99.2	100.0
3.1	Binary, hold out Imb. 3	98.4	100.0	100.0
3.2	Binary, hold out Imb. 4	100.0	99.7	80.3
4	4-class, static	98.8	100.0	100.0
5	4-class, cross-speed	70.2	22.4	55.2
	Binary, cross-speed	99.4	78.3	80.5
Rig 2 — Static State				
6	Binary, static	99.6	99.2	100.0
7.1	Binary, hold out Imb. 3	100.0	100.0	100.0
7.2	Binary, hold out Imb. 4	100.0	100.0	100.0
8	4-class, static	97.9	100.0	95.0
9	4-class, cross-speed	91.9	57.8	77.9
	Binary, cross-speed	100.0	99.8	99.1
Rig 2 — Dynamic State				
10	Binary, dynamic	98.4	99.2	85.2
11.1	Binary, hold out Imb. 3	98.3	100.0	96.0
11.2	Binary, hold out Imb. 4	99.3	99.7	100.0
12	4-class, dynamic	79.0	81.9	79.4
13	4-class, cross-speed	60.2	62.5	57.7
	Binary, cross-speed	100.0	100.0	99.6
Rig 2 — Cross-State				
14	4-class, static→dyn.	35.0	53.4	33.4
	Binary, static→dyn.	77.3	95.4	80.4

4.2 Discussion

All experimental tests defined in Section 3.2 were executed successfully, with every model trained and evaluated under the conditions intended for each scenario, and no procedural failures were encountered. The results, however, varied in their degree of success depending on the complexity of the testing conditions. The following discussion analyses these outcomes in order of increasing difficulty, examining what each test reveals about the diagnostic capabilities and limitations of the proposed system.

Test 1 was devised as an introductory test and, as such, generated clear results. The high-energy impact in this test is considered a point anomaly that substantially deviates from the baseline and is therefore easier to identify. This fact is substantiated by the results, which show that a low-complexity NN correctly classifies the entire dataset, achieving 100% accuracy. This test is a reference point for this experimental sequence, as it deviates from the others: Test 1 evaluates transient, high-magnitude faults, while the remaining tests assess subtle, periodic imbalance data distributed across the vibration spectrum, posing a considerably more demanding classification task.

To assess the models' anomaly-detection capabilities, Tests 2, 6, and 10 perform binary classification between a single anomalous class and the normal condition. Regardless of the operational scenario, both in *Rig 1* and *2* static state and *Rig 2* dynamic state, the models' performance is overall very positive, with the majority of models achieving an accuracy score over 95%. Test 10 expands this experiment by imposing a more complex working scenario. Under the carriage's dynamic motion, the weaker models (notably CNN, KNN, and K-Means) degrade considerably, while the strongest models (XGBoost, the Dual-branch CNN, and Random Forest) remain largely unaffected. The Random Forest (RF) model, while accurate, presents a significantly higher inference time than the other top performers ($\approx$ 40 ms/sample versus sub-millisecond for XGBoost and the Dual-branch CNN), making it less attractive for edge deployment despite remaining within the 100 ms processing budget.

Overall, these results indicate that fault detection can be achieved consistently across different operating conditions, and thus establish a strong baseline for the next more demanding tests. Tests 3, 7, and 11 evaluate the models' performance on correctly classifying unknown faults as anomalous. Two different scenarios were considered. On the one hand, Tests 3.1, 7.1, and 11.1 evaluate the model by interpolating between known fault magnitudes. On the other hand, Tests 3.2, 7.2, and 11.2 present a greater challenge, in which the models must extrapolate beyond the trained imbalance classes to detect a more severe, unseen fault. Both scenarios achieved very favorable results, with nearly all models succeeding across the six tests. Notably, the Dual-branch CNN and XGBoost, the strongest performers in the detection tests, again rank among the best, classifying unseen faults with near-perfect accuracy in every case, confirming their robustness across both interpolation and extrapolation. The exceptions are KNN, which struggled in Tests 3.2,

11.1, and 11.2, and K-Means, which plummeted in Test 3.2. The strong performance even under extrapolation is particularly relevant for deployment, where fault severities are not known in advance and may exceed those represented in the training data.

Following the favorable outcome of the binary classification tests, Tests 4, 8, and 12 address fault classification and the system's ability to identify the specific fault scenario. The models' performance in Tests 4 and 8 is very positive, with most models surpassing 95% accuracy, while K-Means remains the worst-performing model. However, Test 12 reveals that in *Rig 2*, under the influence of lateral movement, a more pronounced effect is felt, reducing the accuracy to $\approx 82\%$. This is a solid outcome, though a clear reduction relative to the near-perfect results of the static tests. The aggregate of these three results demonstrated that the system is capable of achieving solid results for fault classification.

In the dynamic state, since data is recorded continuously during the carriage's shuttle motion, the inertia wheel is no longer the only source of vibration. In this scenario, the translational motion creates broadband transients that are not synchronized with the shaft's rotation, causing the FFT pre-processing to blur the signal across the frequency spectrum. This occurs because frequency-domain pre-processing assumes that the signal's spectral content remains constant within each 100 ms analysis window. Consequently, and given that all benchmarked models use frequency-domain pre-processing, the results peak at $\approx 82\%$. To evaluate whether FFT data adversely affects the results, an additional CNN model was tested on raw time-domain data. This model achieves an accuracy of $\approx 100\%$, showing a considerable improvement over models trained on preprocessed data, and thus validating this argument.

Tests 5, 9, and 13 assess the system's capability to classify faults under a speed shift, training on data recorded at 1000 rpm and testing exclusively on an unseen dataset recorded at 1200 rpm. This represents one of the most demanding scenarios in this work, as cross-speed generalization is where the frequency-domain pre-processing is most heavily relied upon to compensate for changes in operating conditions. As noted in the methodology, this cross-speed evaluation is rarely addressed in PdM literature, making these tests a distinctive and challenging aspect of this work. The three tests span the fixed-axis rig (*Rig 1*), the translational rig in static state (*Rig 2*), and the translational rig in dynamic state, yielding noticeably different outcomes, which are discussed individually below.

In *Rig 1*, Test 5 reveals a moderate outcome. Of the benchmarked models, only the Dual-branch CNN achieved reasonable results, reaching 70% accuracy for fault classification and $\approx 100\%$ accuracy for binary classification. On the other hand, all the classical ML models completely collapsed on fault-classification tasks.

This breakdown can be explained by the architecture and characteristics of classical models. These models create decision boundaries based on vibration data captured at 1000 rpm. When presented with 1200 rpm test data, this new dataset falls outside the learned boundaries, and the models are categorized randomly. The Dual-branch CNN, however, was designed to compensate for this variation through its architecture; even so,

despite a considerable increase in accuracy, the model still achieves a moderate accuracy of $\approx 70\%$. This outcome reveals that the rig's structural dynamics are unfavorable, and that the model struggles to generalize to cross-speed data, even for the best-performing models.

In *Rig 2*, under static conditions, Test 9 produces mixed results. Traditional ML models maintain poor performance in fault classification; however, several of these models perform satisfactorily in binary classification, achieving accuracies over 99%. The best performance in this test belongs to the Dual-branch CNN, which achieves $\approx 92\%$ in fault classification and 100% in binary classification, indicating a substantial performance boost relative to Test 5.

This improvement can be explained by the structural dynamics of *Rig 2*. Whereas the fault signatures on *Rig 1* scale weakly with rotational speed, on *Rig 2* they scale consistently and predictably across the spectrum. Consequently, 1200 rpm data is, in this case, easier to generalize to the trained reference, allowing the speed-normalized features calculated in pre-processing to remain comparable across speeds and thus allowing the Dual-branch CNN model to preserve the fault distinctions that collapse on *Rig 1*.

In *Rig 2*, under dynamic conditions, Test 13 achieves a modest outcome. XGBoost featured the highest accuracy of $\approx 62\%$, which still falls short for reliable fault classification. Binary classification, however, generated superior results, with several models reaching $\approx 100\%$ accuracy, proving that faults are still reliably detected in complex operational conditions.

Test 13 introduces the worst anomaly classification results among the three cross-speed tests. The binary detection results, however, are comparable to those of Test 9 and even surpass those of Test 5. This outcome occurs because Test 13 combines three separate challenges that particularly affect the fault classification results. In addition to the speed variation common to all three tests, the translational movement introduces non-stationary motion transients that blur the spectral fault signatures and push the imbalance classes closer together, negatively impacting the separation between them. Moreover, the reverberation in the physical support structure increases progressively with rotational speed, introducing additional noise into the sensor data. Binary detection, however, only requires separating the normal condition from any fault, a much easier task to accomplish. Consequently, under these test conditions, fault classification achieves the lowest accuracy among the three tests, whereas fault detection yields solid results across all cross-speed cases.

Test 14 represents the most demanding scenario evaluated, requiring the models to transfer knowledge learned exclusively from static-state data to an unseen dynamic-state domain. This cross-state scenario was not found among the works surveyed in the literature review, positioning it as a largely open problem within PdM frameworks. As expected, it produces the weakest results of the entire benchmark for fault classification: the Dual-branch CNN falls to $\approx 35\%$, while XGBoost delivers the best, though still modest, accuracy

of $\approx 53\%$.

Unlike the cross-speed tests, this difficulty cannot be attributed to a variation in speed, since training and testing are performed at the same rotational speed. Instead, the difficulty reflects a true domain shift: the static training data and the dynamic test data are recorded from significantly different working conditions, causing the stationary fault signatures the models learned to no longer match the non-stationary signatures present under dynamic operation. This also explains why XGBoost outperforms the Dual-branch CNN here: as a generic classifier that makes no assumptions about the signal's spectral structure, it adapts more easily to unfamiliar dynamic data.

As in the previous tests, binary detection is more successful, with XGBoost reaching $\approx 95\%$ accuracy, distinguishing a faulty from a healthy state despite the domain shift, whereas it cannot identify the specific fault.

Another decisive factor for edge deployment that was analyzed in this series of tests is inference latency. Since each diagnostic window spans 100 ms of acquired data and acquisition proceeds continuously alongside inference, the classification of a single window must complete within this 100 ms period to sustain real-time operation. All benchmarked models satisfy this constraint comfortably, and, as such, every candidate model is viable for real-time edge inference, leaving classification accuracy, rather than computational cost, as the deciding criterion for model selection.

4.3 Conclusion

In summary, the devised tests were all correctly executed, with no procedural errors encountered. For each test, eight different AI models were benchmarked, with the Dual-branch CNN and XGBoost attaining the best and most consistent performance. The trials showed excellent results in binary classification, unseen fault detection, and fault classification in controlled scenarios. For the most complex, under-researched conditions, the system showed inconsistent results, especially in the cross-state scenario for fault classification, although anomaly detection remained very strong in these scenarios. The tests also showed that the inference time for the benchmarked models is highly suitable for real-time edge inference.

Chapter 5

Conclusions

5.1 Final Considerations

With the consolidation of AI across industries, the implementation of PdM is surging as a prominent maintenance solution for global enterprises. Its integration into industrial maintenance policies reduces downtime and repair costs, while extending machinery's RUL. The main current limitation in PdM deployment, which remains largely unresolved, is domain adaptability. AI models are trained under specific normal operational conditions, whereas during its lifespan, there can be slight fluctuations in the working conditions and unseen, untrained faults can occur. This dissertation's goal was to evaluate the system's capability for real-time edge-monitoring implementation in diverse operational scenarios.

Preliminary research was conducted to determine the basic concepts of data acquisition and anomaly detection. The main algorithmic implementations for vibration data analysis, namely ML and DL, were analyzed and compared. On the one hand, classical ML models offer higher computational efficiency for edge inference, perform well on modest datasets, and can be more robust in cross-domain generalization. On the other hand, DL algorithms can capture more complex patterns without requiring manual feature extraction.

Based on the conducted review, a benchmark of eight different AI models was implemented to identify the most accurate model for our system. Simultaneously, a frequency-domain pre-processing pipeline was developed to reduce the impact of domain-shift scenarios on the model's performance. A series of tests was devised to assess the implementation of the developed system in changing operational conditions.

The results indicate that both the Dual-branch CNN and XGBoost algorithms achieve robust outcomes in fault detection across the vast majority of tests. This confirms that the system can differentiate faults from normal operation in real time across various domain conditions, representing a valuable achievement, taking into consideration that these complex, under-researched scenarios still lack established solutions. Additionally, the tests' outcomes reflect strong results in fault classification under controlled conditions.

In contrast, the system struggles to correctly distinguish between individual faults in certain cross-speed scenarios. Further analysis indicates that the pre-processing method helps moderately in this scenario, but is negatively affected by the translational movement of the carriage in *Rig 2*. Nonetheless, given that cross-speed classification remains a largely unsolved problem, the strong $\approx 92\%$ accuracy achieved in Test 9 signals a step forward in this field, providing a foundation on which future refinements can build.

Furthermore, an innovative industrial strategy can be implemented based on the obtained results. In highly complex situations where the system cannot reliably identify the specific fault, it can still detect the presence of an anomaly. Consequently, when a faulty state is detected in an industrial setting, the machine can be stopped and flagged for maintenance, where a controlled, standardized movement, similar to the scenarios evaluated in Tests 4 and 8, can then be used to accurately identify the particular fault.

Overall, this dissertation's goal can be considered largely successful. This system can be reliably implemented as a fault-detection monitoring system across various complex, variable working conditions. In addition, although specific fault identification remains limited under domain-shifted conditions, in real-world scenarios, this can be addressed by flagging detected anomalies for inspection under controlled conditions, where reliable classification is possible. In conclusion, this system constitutes an industrially valuable option for PdM.

5.2 Future Works

For future research, the main focus should be on improving the system's domain adaptability and cross-speed proficiency. For that, multiple approaches need to be combined and tested. In the pre-processing phase, alternative time-frequency representations, such as wavelets or spectrograms, could improve cross-speed results. Furthermore, different normalization methods can yield varying performance on more complex tests by affecting the separation between classes. In the processing phase, hybrid NNs architectures with both time-series and frequency-domain inputs could also be a valid hypothesis for tests under dynamic state.

For Test 14, the inclusion of additional sensors to infer the carriage's status (moving or stopped) could improve results by labeling these statuses and creating separate classes for each.

Finally, implementing this system in an industrial setting would be the natural follow-up project. Machinery's continuous monitoring would yield a much larger database to train the model, where the system would be further tested in new dynamic scenarios.

References

[1] Ruaa W. Abdalah, Osamah Fadhil Abdulateef, and Ali Hussein Hamad. A Predictive Maintenance System Based on Industrial Internet of Things for Multimachine Multiclass Using Deep Neural Network. *Journal Europeen des Systemes Automatises*, 58 (2):373–381, 2025. doi: 10.18280/jesa.580218.

[2] Xanthi Bampoula, George Siaterlis, Nikolaos Nikolakis, and Kosmas Alexopoulos. A deep learning model for predictive maintenance in cyber-physical production systems using LSTM autoencoders. *Sensors (Switzerland)*, 21(3):1–14, 2021. doi: 10.3390/s21030972.

[3] Eyup Cinar, Sena Kalay, and İnci Sarıçiçek. A Predictive Maintenance System Design and Implementation for Intelligent Manufacturing. *Machines*, 10(11), 2022. doi: 10.3390/machines10111006.

[4] Michał Cioch, Monika Kulisz, and Arkadiusz Gola. Comparison of machine learning methods in predictive maintenance of machines. *Advances in Science and Technology Research Journal*, 19(11):33–44, 2025. doi: 10.12913/22998624/208284.

[5] Kassem Danash, Hassan Harb, H. O. Issa, and Louai Saker. An explainable data-driven optimization framework for industrial predictive maintenance scheduling. *Results in Engineering*, 29, 2026. doi: 10.1016/j.rineng.2026.109022.

[6] Tulsi Pawan Fowdur and Lekhraj Ajageer. A hybrid online learning based predictive maintenance system for industry 4.0. *Computing*, 107(6), 2025. doi: 10.1007/s00607-025-01494-z.

[7] Ali R. Hosseinzadeh, Ffrank Frank Chen, Mohammad Shahin, and Hamed Bouzary. A predictive maintenance approach in manufacturing systems via AI-based early failure detection. *Manufacturing Letters*, 35:1179–1186, 2023. doi: 10.1016/j.mfglet.2023.08.125.

[8] Zokirov Javohir, G. Khurazov, M. Temirova, Khudaybergan Khudayberganov, Inomjon B. Matkarimov, Malika Mirzaeva, C. van Truong, and K. Rajabova. Deep Learning Enhanced Predictive Maintenance Framework Using Industrial Internet of Things Sensors for Smart Manufacturing Systems. *International Journal of Industrial Engineering and Management*, 16(4):389–410, 2025. doi: 10.24867/IJIEM-395.

[9] Anusak Luekiangkhamla, Natee Panagant, Sujin Bureerat, and Nantiwat Pholdee. Application of various machine learning models for fault detection in the refrigeration system of a brewing company. *Engineering and Applied Science Research*, 50(2):149–154, 2023. doi: 10.14456/easr.2023.16.

[10] Ons Masmoudi, Mehdi Jaoua, Amel Jaoua, and Soumaya Yacout. Data Preparation in Machine Learning for Condition-based Maintenance. *Journal of Computer Science*, 17(6):525–538, 2021. doi: 10.3844/JCSSP.2021.525.538.

[11] Lukas Meitz, Julia Marie Senge, Tim Wagenhals, Thorsten Schöler, Jörg Hähner, Janick Edinger, and Christian Krupitzer. A Literature Review Framework and Open Research Challenges for Predictive Maintenance in industry 4.0. *Computers and Industrial Engineering*, 206, 2025. doi: 10.1016/j.cie.2025.111193.

[12] Noor Azad Mohammed, Osamah Fadhil Abdulateef, and Ali Hussein Hamad. An IoT and Machine Learning-Based Predictive Maintenance System for Electrical Motors. *Journal Europeen des Systemes Automatises*, 56(4):651–656, 2023. doi: 10.18280/jesa.560414.

[13] Prabu Subramani, R. Senthilraja, Ahmed Mudassar Ali, S. Jayapoorani, and M. Arun. AI-Driven Predictive Maintenance for Smart Manufacturing Systems Using Digital Twin Technology. *International Journal of Computational and Experimental Science and Engineering*, 11(1):1350–1355, 2025. doi: 10.22399/ijcesen.1099.

[14] V. C. Todkari, Avinash P. Kaldate, Shrikrishna U. Kolhar, and Nilesh P. Sable. Development of an AI-Based Predictive Maintenance Application for CNC Machines. *Journal Europeen des Systemes Automatises*, 58(11):2417–2425, 2025. doi: 10.18280/jesa.581118.

[15] David Velásquez, Enrique Perez, Xabier Oregui, Arkaitz Artetxe, Jorge Manteca, Jordi Escayola Mansilla, Mauricio Toro, Mikel Maiza, and Basilio Sierra. A Hybrid Machine-Learning Ensemble for Anomaly Detection in Real-Time Industry 4.0 Systems. *IEEE Access*, 10:72024–72036, 2022. doi: 10.1109/ACCESS.2022.3188102.

[16] Saleh Othman Alhuqay, Abdulaziz Turki Alenazi, Hamad Abdulaziz Alabduljabbar, and Mohd Anul Haq. Improving Predictive Maintenance in Industrial Environments via IIoT and Machine Learning. *International Journal of Advanced Computer Science and Applications*, 15(4):627–636, 2024. doi: 10.14569/IJACSA.2024.0150464.

[17] Reem Atassi and Fuad Alhosban. Predictive Maintenance in IoT: Early Fault Detection and Failure Prediction in Industrial Equipment. *Journal of Intelligent Systems and Internet of Things*, 9(2):231–238, 2023. doi: 10.54216/JISIoT.090217.

[18] D. Dhinakaran, Edwin Raja S, Rajendran Velselvi, and N. Purushotham. Intelligent IoT-Driven Advanced Predictive Maintenance System for Industrial Applications. *SN Computer Science*, 6(2), 2025. doi: 10.1007/s42979-025-03695-x.

[19] Marta Fernandes, Juan M. Corchado Rodríguez, and Goreti Marreiros. Machine learning techniques applied to mechanical fault diagnosis and fault prognosis in the context of real industrial manufacturing use-cases: A systematic literature review. *Applied Intelligence*, 2022. doi: 10.1007/s10489-022-03344-3.

[20] Ida Hector and Rukmani Panjanathan. Predictive maintenance in Industry 4.0: A survey of planning models and machine learning techniques. *PeerJ Computer Science*, 10:1–50, 2024. doi: 10.7717/peerj-cs.2016.

[21] Urszula Jachymczyk, Pawel Knap, and Krzysztof Lalik. Improved Intelligent Condition Monitoring with Diagnostic Indicator Selection. *Sensors*, 25(1), 2025. doi: 10.3390/s25010137.

[22] Ouiam Khattach, Omar Moussaoui, and Mohammed Haissaine. Enhancing machine failure prediction with a hybrid model approach. *IAES International Journal of Artificial Intelligence*, 13(3):2946–2955, 2024. doi: 10.11591/ijai.v13.i3.pp2946-2955.

[23] Rindi Kusumawardani, Nani Kurniati, and Fauzi Irfandi Yusuf. Machine Learning Applications for Condition-Based Maintenance of Critical Machinery Components. *International Journal of Engineering Trends and Technology*, 73(8):190–201, 2025. doi: 10.14445/22315381/IJETT-V73I8P116.

[24] Minh Huong Le-Nguyen, Fabien Turgis, Pierre Emmanuel Ifeolohoum Fayemi, and Albert Bifet. Exploring the potentials of online machine learning for predictive maintenance: A case study in the railway industry. *Applied Intelligence*, 53(24):29758–29780, 2023. doi: 10.1007/s10489-023-05092-4.

[25] Narayan Nayak, Ambarish Gajendra Mohapatra, Ashish Khanna, Jaideep Talukdar, Satyapriya Satapathy, Dipak Ranjan Nayak, and Nilam N. Ghuge. Enhancing fault detection and predictive maintenance of rotating machinery with Fiber Bragg Grating sensor and machine learning techniques. *International Journal of Information Technology (Singapore)*, 17(2):1225–1234, 2025. doi: 10.1007/s41870-024-02256-4.

[26] Eduardo Quiles-Cucarella, Alejandro García-Bádenas, Ignacio Agustí-Mercader, and Guillermo Escrivá-Escrivá. Optimizing Bearing Fault Diagnosis in Rotating Electrical Machines Using Deep Learning and Frequency Domain Features. *Applied Sciences (Switzerland)*, 15(6), 2025. doi: 10.3390/app15063132.

[27] Rajnish M. Rakholia, Andrés L. Suárez-Cetrulo, Manokamna Singh, and Ricardo Simón-Carbajo. Integrating AI and IoT for Predictive Maintenance in Industry 4.0 Manufacturing Environments: A Practical Approach. *Information (Switzerland)*, 16 (9), 2025. doi: 10.3390/info16090737.

[28] Sheeja S. Rani, Raafat Omar Aburukba, and K. El-Fakih. Optimizing Predictive Maintenance in Industrial IoT Cloud Using Dragonfly Algorithm. *IEEE Internet of Things Journal*, 12(17):36001–36018, 2025. doi: 10.1109/JIOT.2025.3582671.

[29] Gulsim Rysbayeva, A. D. Umurzakova, and Mohammed Hussein Ahmed Alanesi. Implementation of Advanced Vibration Analysis Techniques for Predictive Maintenance on Rotating Machinery. *Eastern-European Journal of Enterprise Technologies*, 1(9(133)):69–79, 2025. doi: 10.15587/1729-4061.2025.323894.

[30] Abdul Wahid, John G. Breslin, and Muhammad Intizar Ali. Prediction of Machine Failure in Industry 4.0: A Hybrid CNN-LSTM Framework. *Applied Sciences (Switzerland)*, 12(9), 2022. doi: 10.3390/app12094221.

[31] Ibrahim Abdullahi, Stefano Longo, and Mohammad Samie. Towards a Distributed Digital Twin Framework for Predictive Maintenance in Industrial Internet of Things (IIoT). *Sensors*, 24(8), 2024. doi: 10.3390/s24082663.

[32] Daniel Campos-Olivares, Alejandro Carrasco-Muñoz, Mirko Mazzoleni, Antonio Ferramosca, and Amalia Luque-Sendra. Screening of Machine Learning Techniques on Predictive Maintenance: A Scoping Review. *Dyna (Spain)*, Dyna-Acelerado, 2023. doi: 10.6036/10950.

[33] Auday Shaker Hadi and Luttfi A. Al-Haddad. Towards Fault Diagnosis Interpretability: Gradient Boosting Framework for Vibration-Based Detection of Experimental Gear Failures. *Journal of Dynamics, Monitoring and Diagnostics*, 4(3):160–169, 2025. doi: 10.37965/jdmd.2025.771.

[34] Youssef Jabari, Imad El Adraoui, and Hassan Gziri. Towards a Comparative and Analytical Study on Intelligent Models Implemented in Predictive Maintenance. *International Journal on Technical and Physical Problems of Engineering*, 17(4):134–143, 2025.

[35] Rochdi Kerkeni, Anis M'halla, and Kais Bouzrara. Unsupervised Learning and Digital Twin Applied to Predictive Maintenance for Industry 4.0. *Journal of Electrical and Computer Engineering*, 2025(1), 2025. doi: 10.1155/jece/3295799.

[36] Thomas Küfner, Frank Döpper, Daniel Müller, and AndréGerhard Trenz. Predictive Maintenance: Using Recurrent Neural Networks for Wear Prognosis in Current Signatures of Production Plants. *International Journal of Mechanical Engineering and Robotics Research*, 10(11):583–591, 2021. doi: 10.18178/ijmerr.10.11.583-591.

[37] Chirag Mongia, Shankar Sehgal, and Deepam Goyal. Vibration-based Condition Monitoring for Intelligent Anomaly Detection in Rotary Machinery Under Dynamic Environments. *Journal of Vibration Engineering and Technologies*, 14(2), 2026. doi: 10.1007/s42417-025-02283-w.

[38] Aziz Kubilay Ovacıklı, Mert Yağcioğlu, Sevgi Demircioğlu, Tugberk Kocatekin, and Sibel Birtane Akar. Supervised Learning-Based Fault Classification in Industrial Rotating Equipment Using Multi-Sensor Data. *Applied Sciences (Switzerland)*, 15 (13), 2025. doi: 10.3390/app15137580.

[39] Mert Sehri, Igor Mattos Dos Santos Varejão, Zehui Hua, Vitor Berger Bonella, Adriano Santos, Francisco de Assis Boldt, Patrick Dumond, and Fĺavio Miguel Varejão. Towards a Universal Vibration Analysis Dataset: A Framework for Transfer Learning in Predictive Maintenance and Structural Health Monitoring. *International Journal of Prognostics and Health Management*, 16(3), 2025. doi: 10.36001/IJPHM.2025.v16i3.4239.

[40] Mohammad Shahin, Ffrank Frank Chen, Ali R. Hosseinzadeh, and Neda Zand. Using machine learning and deep learning algorithms for downtime minimization in manufacturing systems: An early failure detection diagnostic service. *International Journal of Advanced Manufacturing Technology*, 128(9-10):3857–3883, 2023. doi: 10.1007/s00170-023-12020-w.

[41] Jeetesh Sharma, Murari Lal Lal Mittal, and Gunjan Soni. Predictive Maintenance of Industrial Assets: A Fault Detection and Diagnosis Approach with Explainable AI. *International Journal of Reliability, Quality and Safety Engineering*, 32(2), 2025. doi: 10.1142/S0218539324400072.

[42] Nur Haninie Abd Wahab, Khairunnisa Binti Hasikin, Khin Wee Lai, Kaijian Xia, Lulu Bei, Kai Huang, and Xiang Wu. Systematic review of predictive maintenance and digital twin technologies challenges, opportunities, and best practices. *PeerJ Computer Science*, 10, 2024. doi: 10.7717/PEERJ-CS.1943.

[43] Yaqiao Yang and Hongjun Wang. Random Forest-Based Machine Failure Prediction: A Performance Comparison. *Applied Sciences (Switzerland)*, 15(16), 2025. doi: 10.3390/app15168841.

[44] Ozlem Ece Yürek, Derya Birant, and Alp Kut. T-PdM: A tripartite predictive maintenance framework using machine learning algorithms. *International Journal of Computational Science and Engineering*, 25(3):325–338, 2022. doi: 10.1504/IJCSE.2022.123122.

[45] Xuanbai Yu and Olivier Caspary. TKEO-Enhanced Machine Learning for Classification of Bearing Faults in Predictive Maintenance. *Applied Sciences (Switzerland)*, 15 (7), 2025. doi: 10.3390/app15073774.

[46] Mikell P. Groover. *Automation, Production Systems, and Computer-Integrated Manufacturing*. Pearson Prentice Hall, 3 edition, 2008.

[47] Frank Lamb. *Industrial Automation: Hands-On*. McGraw-Hill Education, 2013.

[48] IEC. IEC 62264-1:2013 – enterprise-control system integration – part 1: Models and terminology, 2013.

[49] EMQ Technologies. Exploring isa-95 standards in manufacturing, 12 2024. URL https://www.iotforall.com/exploring-isa95-standards-in-manufacturing. Accessed: 2026-03-06.

[50] European Commission. Industry 5.0: Towards a sustainable, human-centric and resilient european industry. Technical report, Publications Office of the European Union, 2021.

[51] Marco Antonio Díaz Martínez, Reina Verónica Román Salinas, Rocío del Carmen Vargas Castilleja, Mario Alberto Morales Rodriguez, Gabriela Cervantes Zubirias, Yadira Aracely Fuentes Rubio, Jose Roberto Grande Ramirez, and Vania Irais González Rubín. The impact of the internet of things on corporate sustainability in the industry 4.0 era: A systematic literature review. *Results in Engineering*, 26, 2026. doi: 10.1016/j.rineng.2025.1003452. URL https://www.sciencedirect.com/science/article/pii/S2590123026003452.

[52] Aditya Pratap Singh and Pradeep Tomar. Ai and iot capabilities: Standards, procedures, applications, and protocols. In Gurjit Kaur, Pradeep Tomar, and Marcus Tanque, editors, *Artificial Intelligence to Solve Pervasive Internet of Things Issues*, pages 67–83. Academic Press, 2021. ISBN 978-0-12-818576-6. doi: 10.1016/B978-0-12-818576-6.00004-6.

[53] Polimak. The differences between IoT and IIoT (IoT vs IIoT), 2026. URL https://polimak.com/en/the-differences-between-iot-and-iiot-iot-vs-iiot/. Accessed: 2026-03-11.

[54] GFOS MBH. Digital twin | definition & function, 2026. URL https://www.gfos.com/en/glossary/digital-twin/. Accessed: 2026-03-11.

[55] Angreine Stevia Singal, Sweetstory Josefard Regar, Lorena Rima Aneru, Antonius P. G. Manginsela, and Marson James Budiman. Motor and bearing temperature monitoring of milling machine based on PLC-SCADA. *International Journal of Science and Research (IJSR)*, 5(6), May 2026. ISSN 2827-9832. URL http://ijsr.internationaljournallabs.com/index.php/ijsr.

[56] Yaguo Lei, Huan Liu, Naipeng Li, Junyi Cao, Yuting Qiao, and Hongbo Wang. Condition monitoring and fault diagnosis of industrial robots: A review. *Science China Technological Sciences*, 68(1):1–25, 2025. doi: 10.1007/s11431-024-2810-2.

[57] Juan Carlos A. Jauregui Correa and Alejandro A. Lozano Guzman. Chapter eight - condition monitoring. In *Mechanical Vibrations and Condition Monitoring*, pages 147–168. Academic Press, 2020. doi: 10.1016/B978-0-12-819796-7.00008-1.

[58] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), jul 2009. ISSN 0360-0300. doi: 10.1145/1541880.1541882. URL https://doi.org/10.1145/1541880.1541882.

[59] 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*, 104:799–834, 2018. doi: 10.1016/j.ymssp.2017.11.016.

[60] Rob Toulson and Tim Wilmshurst. Analog input. In *Fast and Effective Embedded Systems Design: Applying the ARM mbed*, chapter 5, pages 77–90. Newnes, 2012.

[61] Beckhoff Automation. Xfc technology oversampling. Application Note DK9222-0909-0005, Beckhoff Automation GmbH & Co. KG, 2024. URL https://download.beckhoff.com/download/document/application_notes/DK9222-0909-0005.pdf.

[62] IQS Directory. Data acquisition systems: Types, principles and applications, 2024. URL https://www.iqsdirectory.com/articles/data-acquisition-system.html. Digital Data Recorder [Image].

[63] Auday Al-Dulaimy, Matthijs Jansen, Bjarne Johansson, Animesh Trivedi, Alexandru Iosup, Mohammad Ashjaei, Antonino Galletta, Dragi Kimovski, Radu Prodan, Konstantinos Tserpes, George Kousiouris, Chris Giannakos, Ivona Brandic, Nawfal Ali, André B. Bondi, and Alessandro V. Papadopoulos. The computing continuum: From IoT to the cloud. *Internet of Things*, 27:101272, 2024. ISSN 2542-6605. doi: 10.1016/j.iot.2024.101272.

[64] Luca Deri, Simone Mainardi, and Francesco Fusco. tsdb: A compressed database for time series. In Antonio Pescapè, Luca Salgarelli, and Xenofontas Dimitropoulos, editors, *Traffic Monitoring and Analysis*, volume 7189 of *Lecture Notes in Computer Science*, pages 143–156, Berlin, Heidelberg, 2012. Springer. ISBN 978-3-642-28534-9. doi: 10.1007/978-3-642-28534-9_16.

[65] Solitary Road. Music production by vibration of taut strings, 2024. URL http://solitaryroad.com/c1031.html. Harmonics and Standing Waves [Images].

[66] Electronics Coach. What is vestigial sideband modulation (vsb)?, 2024. URL https://electronicscoach.com/vestigial-sideband-modulation.html. Spectrum of VSB Signal [Image].

[67] Suhas S. Aralikatti, K. N. Ravikumar, and Hemantha Kumar. Fault diagnosis of single-point cutting tool using vibration signal by rotation forest algorithm. *SN Applied Sciences*, 1(9):1017, 2019. doi: 10.1007/s42452-019-1061-0. Figure 6: Decision tree (RT) based on statistical features of vibration data.

[68] InteractiveChaos. Random forest, 2026. URL https://interactivechaos.com/en/wiki/random-forest.

[69] GeeksforGeeks. Support vector machine (SVM) algorithm, 2026. URL https://www.geeksforgeeks.org/machine-learning/support-vector-machine-algorithm/. SVM Decision Plane Diagram [Image].

[70] JavaTpoint. K-nearest neighbor (KNN) algorithm for machine learning, 2026. URL https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. Classification of KNN algorithm [Image].

[71] Distributed Machine Learning Community. Introduction to boosted trees, 2026. URL https://xgboost.readthedocs.io/en/release_3.2.0/tutorials/model.html. XGBoost Tutorials Documentation.

[72] Abiodun M. Ikotun, Absalom E. Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210, 2023. doi: 10.1016/j.ins.2022.11.139.

[73] Paperspace. Convolutional neural network (CNN), 2026. URL https://machine-learning.paperspace.com/wiki/convolutional-neural-network-cnn. CNN Architecture Diagram [Image].

[74] Jim Frost. Confusion matrix, 2026. URL https://statisticsbyjim.com/glossary/confusion-matrix/. Confusion Matrix Grid Layout [Image].

Appendix A

Code and Data Repository

All software and data developed throughout this work are openly available in a public GitHub repository:

`github.com/Geada8/Tese_GoncaloPacheco`

The repository is organized into the following top-level directories:

- `Data_rig1` — the complete vibration dataset recorded on the fixed-axis inertia wheel (*Rig 1*);
- `Data_rig2` — the complete vibration dataset recorded on the translational-axis inertia wheel (*Rig 2*), including the static and dynamic acquisitions;
- `TwinCAT_MMS` — the TwinCAT 3 project implementations of: data acquisition, preprocessing, CSV storage, and edge inference;
- `TwinCAT_ShuttleMovement` — the TwinCAT 3 project controlling the shuttle motion of *Rig 2*;
- `ML` — the Python implementations of the preprocessing pipeline and all benchmarked AI models.

A `README.md` at the root of the repository describes the directory structure and the dataset format. The following appendices describe the most relevant source files, while each section provides a direct link to the corresponding file in the repository.

Appendix B

Rig 2 Movement

A video demonstrating the lateral shuttle movement in *Rig 2* during dynamic-state data acquisition is available in the repository:

`rig2movement.mp4`

Appendix C

Data Acquisition Programming

This appendix presents the TwinCAT 3 programs that implement the data acquisition, preprocessing, and edge-inference pipeline of the MMS. The complete project is available in the `TwinCAT_MMS` directory of the repository (Appendix A).

C.1 PRG_DAQ

View `PRG_DAQ.TcPOU` in the repository.

C.2 PRG_DataPrep

View `PRG_DataPrep.TcPOU` in the repository.

C.3 PRG_Slicer

View `PRG_Slicer.TcPOU` in the repository.

C.4 PRG_CSVWriter

View `PRG_CSVWriter.TcPOU` in the repository.

C.5 MAIN

View `MAIN.TcPOU` in the repository.

C.6 PRG_TF3600_Math

View `PRG_TF3600_Math.TcPOU` in the repository.

C.7 PRG_TF3800_Inference

View `PRG_TF3800_Inference.TcPOU` in the repository.

Appendix D

Rig 2 Motion Control Programming

This appendix presents the TwinCAT 3 program controlling the shuttle motion of the translational axis. The complete project is available in the `TwinCAT_ShuttleMovement` directory of the repository (Appendix A).

D.1 PRG_ShuttleMotion

View `PRG_ShuttleMotion.TcPOU` in the repository.

Appendix E

AI models programming

This appendix presents the Python implementations of the signal-processing pipeline and the AI models benchmarked in this dissertation. The complete code is available in the `ML` directory of the repository (Appendix A).

E.1 Preprocessing

View `preprocessing.py` in the repository.

E.2 CNN

View `CNN.py` in the repository.

E.3 Dual-branch CNN

View `CNN_dualbranch.py` in the repository.

E.4 LSTM

View `LSTM.py` in the repository.

E.5 Random Forest

View `rf_benchmark.py` in the repository.

E.6 SVM

View `svm_benchmark.py` in the repository.

E.7 KNN

View `knn_benchmark.py` in the repository.

E.8 XGBoost

View `xgb_benchmark.py` in the repository.

E.9 K-Means

View `kmeans_benchmark.py` in the repository.

Appendix F

MMS Electrical Schematic

The electrical schematic of the MMS, detailing the wiring between the IPC, the EtherCAT coupler, the measurement terminals, and the IEPE accelerometers, is presented in Figure F.1.

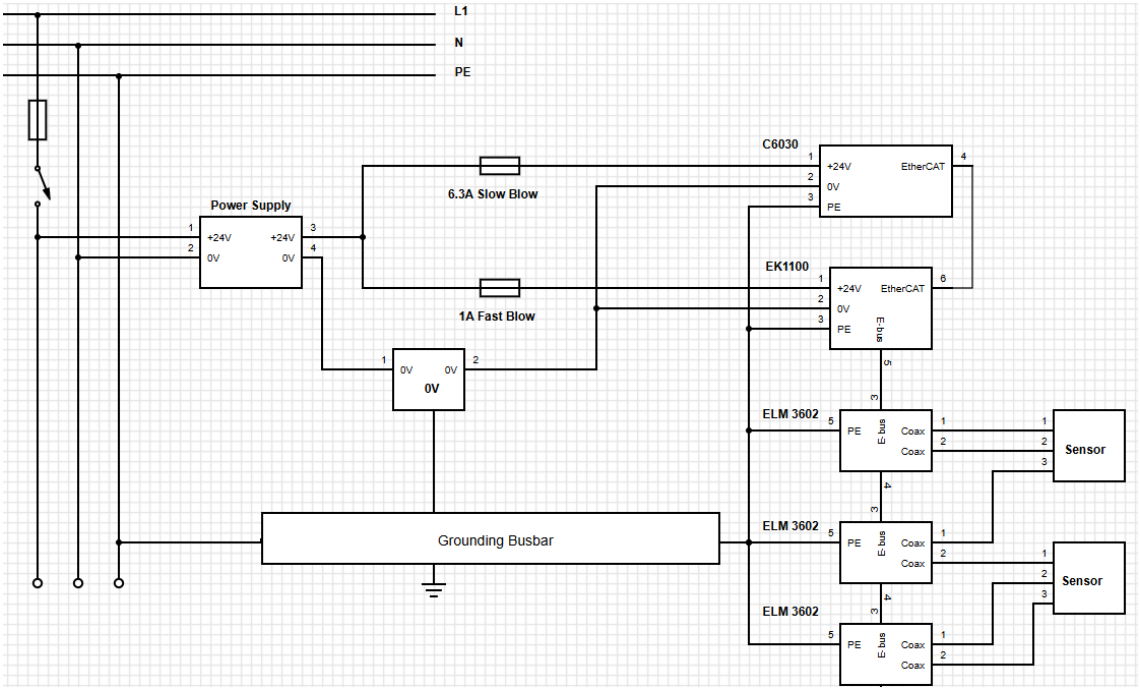


Figure F.1: Electrical schematic of the MMS.