Data-Driven and Model-Based Prognostics and Health Management for Embedded Systems

Juliano Cesar Pimentel

Thesis submitted for the degree of
Doctor of Philosophy
University of Derby

College of Science and Engineering
University of Derby
April 2025

Declaration

As required by the University of Derby, I hereby declare that this thesis is the result of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are appropriately referenced.

Juliano Cesar Pimentel
University of Derby
25 April 2025

To the Lord Jesus, worthy of all glory,

To my mother *(in memorian)*, Diva, for all her love,

To my systers, Sani, Consuelo *(in memorian)* and Celina

To my wife, Kelly,

To my son, Estevao.

Abstract

Electronics are fundamental to most engineering systems. Despite increasing demand driven by rapid digitalization, the Internet of Things, and autonomous vehicles, Prognostics and Health Management (PHM) for electronic systems, particularly those with safety-critical functions, remains limited. While interest in applying machine learning to PHM is growing, its adoption in electronic systems is still in its early stages. This thesis presents a data-driven real-time PHM framework for condition monitoring, fault diagnosis, and multi-step ahead forecasting, all integrated into an unified data pipeline and deployed directly on the edge device. An optimal model is first developed by reducing system measurement noise through exponentially weighted moving average (EWMA) and exponentially weighted moving standard deviation (EWMS). This is followed by feature selection using SHAP (SHapley Additive Explanations), an additive feature attribution method. Finally, hyperparameter optimization is performed using a well-defined search space, and the model's performance is validated through cross-validation. The optimized model, an adapted multivariate bidirectional LSTM (Bidi-LSTM), is then deployed on the target embedded system for real-time inference and multi-step ahead forecasting. The proposed PHM methodology was evaluated in two real-world case studies: (1) an electronic control unit (ECU) used in industrial applications and (2) battery state-of-charge (SOC) estimation for lithium-ion batteries. The data-driven PHM framework outperformed state-of-the-art model-based methods, including Kalman filter-based estimators. It achieved an overall classification accuracy exceeding 99.98% in the ECU experiment and demonstrated superior performance in SOC estimation, with a lower mean absolute error (MAE) and improved forecasting accuracy. The optimal models were deployed on two hardware testbeds to evaluate execution time and resource consumption, yielding positive results for both case studies. The results confirm the generalization capability of the proposed PHM framework, demonstrating its adaptability to different embedded systems with

minimal adjustments. By integrating real-time inference and multi-step-ahead forecasting, this approach provides actionable insights for predictive maintenance, enhancing system reliability and operational efficiency.

List of Publications

J. Pimentel and T. Vladimirova, "Towards mpsoc enabled subsea embedded systems for fault tolerant applications," in 2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), 2019, pp. 1–8.

J. Pimentel, A. A. McEwan, and H. Q. Yu, "Towards a real-time smart prognostics and health management (phm) of safety critical embedded systems," in 2022 25th Euromicro Conference on Digital System Design (DSD), 2022, pp. 696–703.

J. Pimentel, A. A. McEwan, and H. Q. Yu, "A novel real-time framework for embedded systems health monitoring," in 2023 26th Euromicro Conference on Digital System Design (DSD), 2023, pp. 309–316. [**Selected as a candidate for the "Best Paper Award" in DSD 2023**]

J. Pimentel, A. A. McEwan, and H. Q. Yu, "A comprehensive machine learning methodology for embedded systems phm," in 2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), 2023, pp. 1954–1959.

J. Pimentel, A. A. McEwan, and H. Q. Yu, "Multi-step ahead battery SOC estimation using data-driven prognostics and health management," in 13th International Conference on Software and Information Engineering (ICSIE 2024). [**Selected as the "Best Presented Paper" in ICSIE 2024**]

Acknowledgements

First and foremost I thank God and His Son, Jesus Christ for giving me life and health to pursue this dream.

I would like to express my gratitude to my supervisor, Professor Alistair McEwan for his support and insightful feedback. I will always remember his friendship during the pandemic, which helped me navigate that difficult period.

I am equally grateful to my second supervisor, Professor Harry Yu, for his active involvement and guidance throughout this thesis.

I extend my heartfelt appreciation to the University of Derby for providing the resources and support that enabled me to reach this milestone.

I also acknowledge Aker Solutions Ltd. for their support throughout these studies, and more recently, I thank OneSubsea for honoring their commitments and allowing me to complete this thesis.

Last but certainly not least, I extend my deepest thanks to my wife Kelly and son Estevao for their patience and understanding throughout this entire journey.

Juliano Cesar Pimentel
Derby, 2025

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Parameter
η	Learning rate
$\mathcal{J}$	Learner (inducer)
k	Timestep in a discrete-time system
μ_k	Standard deviation
e_k	Scale-dependent error
ω_k	Weights, applicable for neural networks
h_k	Hidden states, applicable for neural networks
b_k	Bias, applicable for neural networks
λ	Vector of hyperparameters
θ	Model parameters set
$\hat{F}_\theta$	Optimised model predictor
$\sigma(k)$	Sigmoidal, activation function
$\mathrm{H}\infty$	H-infinite
∇E	Gradient descent of the quadratic error
$\nabla^d x$	Recursive differencing of x
$F(k)$	LSTM forget gate
$I(k)$	LSTM input gate
$O(k)$	LSTM output gate
$ReLU(k)$	Rectifier linear unit, activation function
$\arg\min(\cdot)$	Arguments of the minima
$\tanh(\cdot)$	Hyperbolic tangent function
$C(I_n)$	Continuous functions on I_n
$\mathcal{L}_S(\theta)$	Training loss (L2 loss) over a set S
$\sum(\cdot)$	Summation

– continued on next page

– continued from previous page

Symbol	Parameter
SS_{res}	Residual sum of squared error
R^2	R-squared
$var(\cdot)$	Variance
$bias(\cdot)$	Bias
$corr(\cdot)$	Correlation
u_k	Dynamic system input
x_k	Dynamic system internal states
$\hat{x}_k$	Estimated state value
$\bar{x}_k$	Average state value
$\hat{x}_k^-$	*A priori* state value estimate
$\hat{x}_k^+$	*A posteriori* state value estimate
y_k	Dynamic System output
$\hat{y}_k$	Estimated target value
$\bar{y}_k$	Average state value
s	Arbritary span
ϕ_k	Feature attribution
$\pi(M)$	Ordered set of permutations of M
$\mathbb{R}$	Real number
$\mathbb{Z}$	Integer number
$\mathbb{Z}_+$	Positive integer number
$\mathbb{E}(\cdot)$	Expected value
$\mathcal{H}(\cdot)$	Nonlinear activation function
$\mathbf{W}_{ij}$	Weight matrix (input, hidden and output layers)
w_k	Model uncertainties
v_k	Measurement noise
P_0	State error covariance
Q_k	Model uncertainties covariance
R_k	Measurement noise covariance
K_k	Kalman correction gain
$\partial f(\cdot)/\partial x$	Partial derivatives
$\mathcal{X}(\cdot)$	Sigma points
$Cov(\cdot)$	Covariance matrix

– continued on next page

– continued from previous page

Symbol	Parameter
Ah	Ampère-hour
C_{nom}	Nominal capacity
$Li\text{-}ion$	Lithium ion
Q_n	Rated battery capacity
RC	Resistor-capacitor
τ_n	Resistor-capacitor time-constant
V_{dc}	DC Voltage
V_t	Terminal voltage of the battery (V)
R_0	Ohmic resistances (Ω)
R_1, R_2	Polarization resistances (Ω)
C_1, C_2	Polarization capacitances (F)
kWh	Kilowatt-hour
Hz	Hertz
$cu.ft$	Cubic foot
^oC	Degrees Celsius
Ω	Ohms (Resistance value)

List of Abbreviations

A	Ampère
AI	Artificial Intelligence
ANN	Artificial Neural Network
AR	Autoregressive
ARMA	Autoregressive Moving Average
ARIMA	Autoregressive Integrated Moving Average
ASIC	Application-Specific Integrated Circuit
BMS	Battery Management System
BO	Bayesian Optimization
BPTT	Back-propagation Through Time
CbM	Condition-Based Monitoring
CC	Coulomb-Counting
CM	Condition Monitoring
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Cross-Validation
DNN	Deep Neural Network
DSO	Distribution System Operator
DT	Decision Tree
ECM	Equivalent Circuit Model
ECU	Electronic control Unit
EIM	Electrochemical Impedance Model
EIS	Electrochemical Impedance Spectroscopy
EKF	Extended Kalman Filter
ELF	Energy Load Forecasting

– continued on next page

– continued from previous page

EM	Electrochemical Model
EV	Electric Vehicle
EWMA	Exponentially Weighted Moving Average
EWMS	Exponentially Weighted Moving Standard Deviation
FMEA	Failure Mode and Effects Analysis
FNN	Feedforward Neural Network
FN	False Negative
FP	False Positive
FPGA	Field Programmable Gate Arrays
FUKF	Fractional Unscented Kalman Filter
GP	Gaussian Process
GPU	Graphical Processing Unit
GRU	Gated Recurrent Units
HWFET	EPA Highway Fuel Economy Test Cycle
HPO	Hyperparameter Optimization
HPPC	Hybrid Power Pulse Characterisation
ICE	Individual Conditional Expectation
IoT	Internet of Things
KF	Kalman Filter
KPI	Key Performance Indicator
LA92	California Unified Driving Schedule
LIME	Local Surrogate
LR	Logistic Regression
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
MaxAE	Max Absolute Error
MCS	Mixed-Criticality System
MIMO	Multi-Input Multi-Output
ML	Machine Learning
MLP	Multiple Layer Perceptron
MSA	Multi-Step Ahead

– continued on next page

– continued from previous page

MCS	Mixed-Criticality System
MPSoC	Multi-Processor System-on-Chip
MSE	Mean Squared Error
NAS	Neural Architecture Search
NARX	Nonlinear Autoregressive Network
NCA	Nickel Cobalt Aluminum (oxide)
NRMSE	Normalised Root Mean Squared Error
OCV	Open Circuit Voltage (V)
OSA	One-Step Ahead
PCBA	Printed Circuit Board Assembly
PDP	Partial Dependence Plot
PHM	Prognostics and Health Management
RBM	Restricted Boltzmann Machine
RCA	Root-Cause Analysis
ReLU	Rectifier Linear Unit
RF	Radio Frequency
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RMSSE	Root Mean Squared Scaled Error
RNN	Recursive Neural Network
RQ	Research Question
RUL	Remaining Useful Life
SHAP	SHapley Additive exPlanation
SLR	Systematic Literature Review
SIMD	Single Instruction, Multiple Data
sMAE	Scaled Mean Absolute Error
sMAPE	Symmetric Mean Absolute Percentage Error
SME	Subject Matter Expert
SGD	Stochastic Gradient Descent
SOC	State of Charge
SoC	System-on-Chip
SoS	System of Systems
SPKF	Sigma Point Kalman Filter

– continued on next page

– continued from previous page

SSE	Sum squared error
SVM	Support Vector Machine
SVR	Support Vector Regression
TN	True Negative
TP	True Positive
UDDS	Urban Dynamometer Driving Schedule
UKF	Unscented Kalman Filter
US06	Supplemental Federal Test Procedure driving schedule
V	Voltage
WCET	Worst-Case Execution Time
xAI	Explainable Artificial Intelligence

Chapter 1

Introduction

This thesis evaluates the practical feasibility of using data-driven methods, based on input-output measurements, as an alternative to model-based approaches for Prognostics and Health Management (PHM) in real-time embedded systems. It explores and implements various machine learning algorithms on real datasets, proposing an optimal method for the analysed systems. Additionally, the thesis emphasizes generalisation and real-time performance, ensuring the proposed methods are applicable across different embedded system scenarios.
The focus of this research is on system-level electronics, including subsystems and system-of-systems levels, rather than individual devices, components, or lower-level implementations. A comprehensive comparison with model-based methods, thoroughly tested and validated, is also presented.

This chapter begins by introducing the research background, followed by the problem statement and research motivation. It then defines the main objectives and research questions. Additionally, the chapter outlines the key contributions of the thesis and provides an overview of the subsequent chapters.

1.1 Background Information

Electronics are crucial building blocks of most engineering systems. The demand for electronics is at an all-time high due to rapid digitalization, the Internet of Things and autonomous vehicles. Miniaturization of electronics, reduction of time to market and new functionalities, especially in the context of autonomous driving, electrification, and connectivity, are bringing about new reliability challenges [1]. Malfunctioning, failure, or miscommunication will likely cause catastrophic accidents, and therefore must be avoided through effective monitoring systems capable of predicting failures and providing timely warnings. Prognostics and Health Monitoring (PHM) for electronics primarily involves data acquisition, diagnosis, prognosis, and maintenance. Functional safety is a key motivation for the development of PHM, and it has been widely implemented in sectors such as avionics and large mechanical systems. During the diagnostic stage, the root cause of faults or the nature of failures is identified. This enables the estimation of the Remaining Useful Lifetime (RUL) of the component in focus, which may need to be supplemented with additional experimental data [2]. Prognosis of RUL is performed based on knowledge of the processes causing degradation and leading to system failure [3]. The ultimate goal of the PHM framework is to predict the system's lifespan and issue a warning before any catastrophic consequences occur. To achieve this, monitored precursors should be used efficiently and quickly for prediction. However, most existing PHM methodologies involve a series of data processing algorithms, which can be resource-intensive, consume high power, and be computationally demanding [1]. The ability to monitor the current state of the systems and predict their behaviour becomes essential. Therefore, techniques for establishing fault tolerance and fault prediction are necessary [4].

A PHM Functional Model for electronics is defined in [5] as the reference or framework for performing engineering design analysis at the device, component, assembly, system, or system-of-systems level. This classification is based on failure modes and precursors, events that signify impending failure [2]. Embedded systems are examples of dependable systems, which are required to be dependable from failures, mishaps and attacks [6]. The study of various types of embedded systems shows that they share four common requirements: dependability, real-time constraints, resource consumption and a long operational life-cycle. Real-time embedded systems are used

in several safety-critical domains, including aerospace, automotive and industrial applications. In these domains, ensuring high dependability is essential [7].

Three approaches to PHM are: data-driven, model-based, and fusion (also called hybrid), which combines the first two approaches [1], [2], [8], [9]. However, the fusion approach is less commonly discussed in the literature, as developing an effective hybrid model remains a challenge [2], [10]. A model-based approach assumes that the system internal states are accessible whilst the data-driven assumes the state-space inaccessibility [1], [11]. The challenge of modelling a process lies in finding a mathematical model that can describe its dynamic input-output behavior, given input-output measurements and prior knowledge about the process [12]. Data-driven approaches are often considered "black box" methods for PHM because they do not require system models or specific system knowledge to begin prognostics [3]. Furthermore, data-driven approaches can be applied to complex and nonlinear systems, as they can model correlations between parameters, interactions between subsystems, and the effects of environmental factors using in situ data from the system. These methods can model degradation characteristics based on historical sensor data, revealing underlying correlations and causalities, and enabling the estimation of system information such as RUL [13].

1.2 Problem Statement and Motivation

A number of publications emphasize that Prognostics and Health Monitoring (PHM) is not yet well developed for electronic systems, especially those with safety-critical functionality [1], [3], [10], [14], [15], [16]. Literature reviews on techniques for implementing PHM in safety-critical embedded electronics suggest that machine learning techniques hold promise for achieving the required levels of prognostics and performance, offering solutions to complex problems that traditional approaches could not solve, or improving the performance of existing systems [17], [18]. The concept of PHM is not new, but its application to electronic systems is relatively recent. PHM for electronics is still an emerging field, and much work remains to be done. While PHM has been applied to other fields for some time, it has only recently been considered for electronics. Extensive research has been conducted on PHM for mechanical systems, but much less attention has been given to PHM in

the context of electronic systems [2], [3], [19]. According to [1], there are no review papers that provide a simple, intuitive overview of the application of intelligent algorithms in PHM for electronics. The authors also found that the number of machine learning applications at the system level is still limited, which reflects the difficulty of achieving comprehensive system monitoring. Most of the research on failure diagnostics and prognostics in the literature focuses on component-level health assessments. However, complex engineering systems are composed of multiple interacting components, and the failure of any component can significantly impact overall system performance. Based on the current knowledge about PHM available in the literature, it is identified that achieving high-level prognostics accuracy in an integrated and automated system is still an issue [15], where development of PHM methodologies for system-level monitoring is just as important as component-level PHM [16], [20].

A limitation of data-driven approaches is their reliance on training data. These methods depend on historical system data to determine correlations, establish patterns, and evaluate trends that lead to failure [13]. Data-driven approaches use statistical pattern recognition and machine learning to detect changes in parameter data, enabling diagnostic and prognostic calculations. While data-driven techniques in PHM were initially based on statistical methods, advances in sensor technology and signal processing have made artificial intelligence (AI) techniques increasingly popular [8]. Today, AI and machine learning (ML) algorithms are integrated into PHM approaches, allowing for more complex fault diagnoses [1]. Additionally, the use of machine learning for PHM of safety-critical embedded systems has gained significant interest [21]. While machine learning has been widely applied in many fields, its use for embedded systems, particularly those with safety-critical functions, is still emerging. Machine learning is expected to help solve complex PHM and RUL challenges that are difficult to address with traditional methods [4].

From a generalisation perspective, it would be interesting to explore whether, in the context of Big Data and deep learning, a single algorithm could be employed to address all levels of prognostics, depending on input size and data quality. Nevertheless, the ongoing advancements in AI will play a key role in the development of future PHM systems across various fields [1], [18]. The use of a fusion (or hybrid) approach remains a challenge from an implementation perspective. The hybrid approach aims to combine the strengths of both model-based and data-

driven methods while minimizing their weaknesses. The fusion of data-driven and model-based approaches is critical for the performance of PHM systems [2]. However, a significant challenge remains in developing a suitable mathematical model that can accurately describe the system dynamics and its processes [12]. Combining physical models with advanced data-driven techniques like deep learning remains particularly difficult [13]. There are also many other challenges to be addressed in the development of system-level PHM methodologies. These challenges include model structure and parameter uncertainties, nonlinearity in system models, environmental effects, measurement noise, and interactions between component degradation processes [16], [20].

Another important aspect for the PHM of embedded systems is the ability to anticipate a potential failure and prevent catastrophic failures, especially for safety-critical systems. In this context, multi-step ahead (MSA) forecasting provides significant benefits by allowing for longer-term predictions, better system management and predictive maintenance. Several publications related to multi-step ahead forecasting for real-world applications have been identified [22]: these models include those for predicting critical levels of abnormality in physiological signals, flood forecasting using RNNs, nitrogen oxide emissions forecasting, electric power load forecasting, and even earthquake and seismic response prediction. However, to the best of our knowledge, there are relatively few studies on MSA forecasting for embedded electronics, highlighting its scarcity as an area worthy of further research. In addition, multi-step forecasting is very challenging and there are a lack of studies available that consist of machine learning algorithms and methodologies for multi-step ahead forecasting [23]. Similarly, there is limited research on MSA prediction for the State of Charge (SOC) of Lithium-ion batteries. In [24], a statistical model is proposed for predicting SOC under high charging/discharging rates (C-rates). Machine learning-based MSA estimation for the long-term prognosis of Lithium-ion battery conditions is explored in [25], [26], and [27].

1.3 Research Objectives

This thesis addresses real-time condition monitoring and prognostics for microcontrolled embedded systems, enabling real-time inference and forecasting at system-level using data-driven methods.

The objectives of this thesis are to:

- Evaluate the use of machine learning-based data-driven modelling for the prognostics and health management of embedded systems to prevent potential catastrophic failures. The results obtained through these approaches shall meet well established performance criteria to demonstrate the feasibility of using data-driven models instead of other approaches, such as model-driven modelling.
- Selected appropriate machine learning models, particularly deep learning techniques such as long short-term memory (LSTM) with higher model significance, able to cope with long datasets where past data is as important as recent ones for fault classification.
- Test the selected models on a real-world dataset to verify the model's accuracy and significance, validate the proposed framework and measure the overall system classification.
- Provide, with the selected model, a real-time inference for the system's health condition monitoring as well as multi-step ahead forecasting for long-terms predictions.
- Propose a comprehensive framework encompassing a thorough offline schema to determine an optimized model significance and accuracy.
- Test the proposed framework on a different system to verify its potential for generalisation.
- Test the proposed framework against a model-based state-of-the-art implementation of the new system.
- Test the proposed framework for inference and multi-step ahead real-time performance on selected embedded platforms.

1.4 Scope and Research Questions

The primary Research Questions (RQ) that guide this work and provide a framework for the research are as follows [4]:

I. **RQ1:** Can data-driven machine learning algorithms, particularly those based on recurrent neural networks, be effectively utilised and trusted for the PHM of electronic systems to predict catastrophic failures?

II. **RQ2:** Are the results obtained from data-driven machine learning algorithms sufficiently explainable to be trusted and understood, particularly for reliability assessments and root cause analysis?

III. **RQ3:** What are the most suitable machine learning techniques for the Prognostics and Health Management (PHM) of real-time embedded systems.

These research questions are explored in detail in the subsequent chapters.

1.5 Thesis Contributions

In this thesis, we propose a novel data-driven, system-level PHM (Prognostics and Health Management) methodology for real-time embedded systems. The proposed methodology emphasizes generalisation [28] and real-time performance [29]. Additionally, the novel method is compared with state-of-the-art model-based implementations to evaluate their accuracy and suitability for the proposed datasets.

The key contributions of this thesis can be summarised as follows:

1. **A novel real-time PHM methodology for a system-level approach, based on recurrent neural networks, used by embedded systems.** This efficient methodology includes three stages: an offline processing schema that identifies cause-and-effect relationships between outputs and relevant inputs; provides noise immunity to handle various system signals; optimises the mapping of complex nonlinearities and time-variability through an automated process; and establishes the estimator with the lowest risk across all candidate models. The online processing is divided into two stages: real-time data processing for online inference and multi-step ahead forecasting for providing an operational advisory window.

2. **A large real-world dataset with over 2,000 hours of continuous data.** The dataset was collected from an electronic control unit located at an remote access industrial environment and data was retrieved every minute performing over 130,000 datapoints. This valuable dataset was used to test and validate the novel real-time PHM methodology, with results that exceeded a model precision score $> 99.97\%$ for the real-time inference.

3. **A multi-step ahead forecasting estimator for embedded systems to provide an operational advisory window to prevent catastrophic failures.** The multi-step ahead (MSA) forecasting estimator is a hybrid approach combining statistical regression methods (ARIMA) to forecast explanatory variables in real-time with optimised machine learning models based on an adapted multivariate bidirectional long short-term memory (LSTM) algorithm. This estimator forecasts future values of interest according to the specified horizon (H). For a horizon of 30 future timesteps, the model achieved a precision score of $> 80\%$ for the real-time implementation.

4. **A thorough comparison of the proposed novel data-driven real-time PHM method with a state-of-the-art model-based approach**. A state-space model for a nonlinear dynamic system is proposed, and online state estimators based on Kalman filters are implemented. Two variants of Kalman filters were used: the extended Kalman filter (EKF) and the unscented Kalman filter (UKF). These models were applied to the battery dataset in Chapter 6, and the results were compared with those of the proposed novel data-driven PHM method.

5. **A complete framework for battery state of charge (SOC) determination using the novel data-driven PHM method, including multi-step ahead forecasting.** The data-driven PHM method, built upon extensive offline training and feature selection, allowed for more accurate SOC determination compared to model-based methods. The PHM-based SOC determination outperformed state-of-the-art methods in the literature in terms of mean absolute error (MAE) for all drive cycles tested. The results were consistently kept below the 1% threshold, which is considerably better than the 2% found in the literature. Additionally, the maximum absolute error (MaxAE) was lower than those computed for both EKF and UKF. The MSA forecasting estimator was also tested on two different hardware platforms, achieving results within two seconds on one platform and four seconds on the other, demonstrating its feasibility for deployment in real-time environments

1.6 Thesis Organisation

The contents of the remainder of the thesis are structured as follows:

Chapter 2 reviews related work, providing a comprehensive investigation of the state-of-the-art in Prognostics and Health Management (PHM) for electronic systems, establishing the foundation for the remainder of this thesis. The chapter presents a detailed review of machine learning techniques and their application in PHM, offering insights into current advancements and emerging trends. Finally, it explores multi-step ahead (MSA) forecasting in PHM, emphasizing its benefits, key application areas, and the growing role of data-driven methodologies in predictive maintenance.

Chapter 3 discusses both model-based and data-driven approaches for nonlinear discrete systems. To enable a comparison between the proposed data-driven method and state-of-the-art model-based algorithms, two implementations of Kalman filters are presented: the Iterated Extended Kalman Filter and the Unscented Kalman Filter. The mathematics and underlying assumptions of these approaches are discussed in detail. Additionally, a deep learning method of interest is introduced: Recurrent Neural Networks (RNNs) and their use as universal approximators. The chapter further examines Long Short-Term Memory (LSTM) networks, a specialised type of RNN, for their inherent advantages, particularly their use of memory blocks. A bidirectional LSTM architecture is proposed as the candidate for the novel PHM methodology.

Chapter 4 details the novel data-driven PHM methodology, which comprises three main stages: offline processing, real-time inference, and multi-step ahead forecasting. A method overview is provided, along with a data flow diagram outlining the stages. In the offline processing section, a method for handling noisy measurements using Exponentially Weighted Moving Average (EWMA) and Exponentially Weighted Moving Standard Deviation (EWMS) is proposed. Feature extraction is discussed as a means of identifying important system information indicative of health state transitions in system degradation. SHAP (SHapley Additive exPlanations) is introduced for model explainability, focusing on feature attributions. The chapter also covers automated hyperparameter optimization and model selection. It concludes with a detailed explanation of real-time data

processing and multi-step ahead forecasting strategies.

Chapter 5 focuses on the first experiment of the thesis, which involves a real-world dataset from an electronic control unit, operating remotely. The real-time data, gathered from multiple sensors and actuators, is pushed to a cloud server where it is processed for condition monitoring and historical data analysis. The proposed PHM methodology is applied to assess a faulty state that led to system downtime. Nine different machine learning models are implemented and compared based on their deviation from the true state, particularly during faulty conditions. Both real-time state estimators and multi-step ahead forecasting are evaluated, and the results are discussed in detail.

Chapter 6 presents the second experiment, which involves the real-time estimation of the State of Charge (SOC) of a Lithium-ion battery system. The novel PHM method is compared with a model-based approach that uses a Second-Order RC Thevenin Battery Model. The system equations and state-space modelling are described. A rich, publicly available dataset is used for comparison. The model-based SOC estimation is performed using the Kalman filter techniques discussed in Chapter 3, while the data-driven PHM method relies solely on input-output (I/O) data and their intrinsic correlation. The results are compared and discussed for both real-time inference and multi-step ahead forecasting.

Finally, **Chapter 7** discusses the potential for generalisation achieved in this thesis, summarizes its conclusions and key achievements, and outlines directions for future research.

Chapter 2

Related Works

This chapter reviews related work on Prognostics and Health Management (PHM) for electronic systems, defining the appropriate taxonomy, capabilities, and current approaches. The discussion focuses on embedded systems, providing a detailed analysis of dependability in the context of PHM. After establishing the terminology and scope, the chapter presents a comprehensive review of machine learning applications in PHM, offering insights into the current state of research and emerging trends. Finally, the chapter explores multi-step-ahead (MSA) forecasting in PHM, highlighting its advantages, key applications, and the growing role of data-driven methodologies in predictive maintenance.

2.1 Prognostics and Health Management

Prognostics and Health Management (PHM) and Condition-Based Maintenance (CBM) have emerged in recent years as significant technologies that are having a substantial impact on both military and commercial maintenance practices [30]. There are several definitions for PHM of electronic systems in the literature, including those provided in [1], [3], [5], [31] and many others.
The definitions presented in [32], [33] and [34] offer a comprehensive coverage:
Prognostics and Health Management is an approach to system life-cycle support that aims to reduce or eliminate inspections and time-based maintenance. This is

achieved through accurate monitoring, incipient fault detection, and the prediction of impending faults. Prognostics (P) involves using appropriate models to assess the degree of performance degradation and predict the Remaining Useful Life (RUL) of the system.
Health Management (HM) integrates outputs from monitoring, diagnosis, and prognosis to make optimal maintenance and logistical decisions, considering factors such as economic costs and available resources. Overall, PHM significantly enhances operational safety, system reliability and maintainability, while simultaneously reducing lifecycle costs [33]. And yet, [35] defines PHM in the IEEE standard as a maintenance and asset management approach that utilizes signals, measurements, models, and algorithms to detect, assess, and track system degradation, as well as predict failure progression. The goal is to protect the integrity of the equipment and avoid unanticipated operational issues that could lead to performance deficiencies, degradation, or adverse effects on mission safety [5].
Some of the key functional capabilities of PHM systems, as outlined in [30] and [36] are listed in the table below:

Table 2.1: PHM Capabilities

PHM Functions
Fault Detection Fault Isolation Condition Monitoring
Performance Degradation Trending Tracking
Fault Prediction Predictive Prognostics Remaining Useful Life
Information Management Decision Making Support

2.1.1 Taxonomy of Related Terms

There have been many attempts to correlate PHM with other approaches and techniques such as anomaly detection, condition based maintenance, health management

and others [33], [37], [38], [39]. After a careful review process, the taxonomy of the related terms can be accurately defined:

- **Condition Monitoring (CM):** This refers to the application of the appropriate sensors (data), analysis (knowledge), and reasoning (context) to estimate the health and track the degradation of equipment. Usually assumes human-in-the-loop processing [30], [40].
- **Condition Based Monitoring (CbM):** This approach uses of machinery run-time data to determine the machinery condition and hence its current fault/failure condition, which can be used to schedule required repair and maintenance prior to breakdown [30], [40].
- **Remaining Useful Life (RUL):** This refers to the remaining time before system health falls below a defined failure threshold [40], [41]. RUL is the remaining time-cycles before requiring maintenance [42], the left-over operational time of an asset before failure [43]; or the prediction of future failure time or the remaining time to maintain regular operation [44].

While CM focuses on the present status of the system, PHM extends this to the future [19]. PHM not only monitors the current condition but also predicts future damage progression and estimates the Remaining Useful Life (RUL). Figure 2.1 shows the relationship of CM, PHM and RUL.

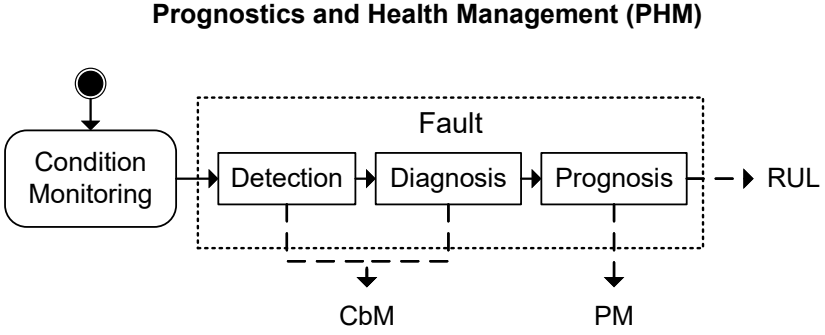

Figure 2.1: Relationship of CM, PHM and RUL

2.1.2 PHM Approaches

Three approaches to PHM are presented in the literature [2], [3], [8], [9], [19], [40]: data-driven, model-driven and fusion (or hybrid) approaches.
The model-based approach to PHM uses mathematical representations to incorporate a physical understanding of the system, including both system modelling and physics-of-failure (PoF) modelling. Prognosis of RUL is performed based on knowledge of the processes that cause degradation and lead to system failure [3].
Data-driven techniques leverage data to provide intelligent decision-making information. Anomalies, trends, and patterns are detected in data collected by in-situ monitoring to assess the health status of the system. These trends are subsequently used to estimate the time to failure [3]. As explained in [16] and [20], the data-driven approach can predict RUL through statistical and probabilistic methods. This approach identifies a suitable damage propagation model by analyzing collected data on degradation paths and relevant operating conditions. Once the model is developed, future system states can be predicted using mathematical models, weighted parameters, and other derived factors based on training data under various usage conditions. The data-driven approach focuses on identifying system characteristics and future behaviors based on trends in the data and indicative parameters of the system's health.
Statistical and machine learning techniques are commonly used to detect changes in component performance and degradation. Supervised machine learning, in particular, addresses the issue of extracting a model from labeled training data, which can then be used to make predictions on new data from the same underlying distribution, while minimizing error.
The data fusion (hybrid) approach combines the strengths of both data-driven and model-based methods to provide an optimised, accurate prognostics solution. This approach first detects failure mechanisms, critical components, and monitoring parameters using the data-driven method. Then, the model-based approach is employed to assess product damage. The RUL of the product is estimated by combining the results from the model with anomaly trends identified by the data-driven method.
A further classification of PHM approaches and the technologies they use is provided in [45], as shown in Fig. 2.2.

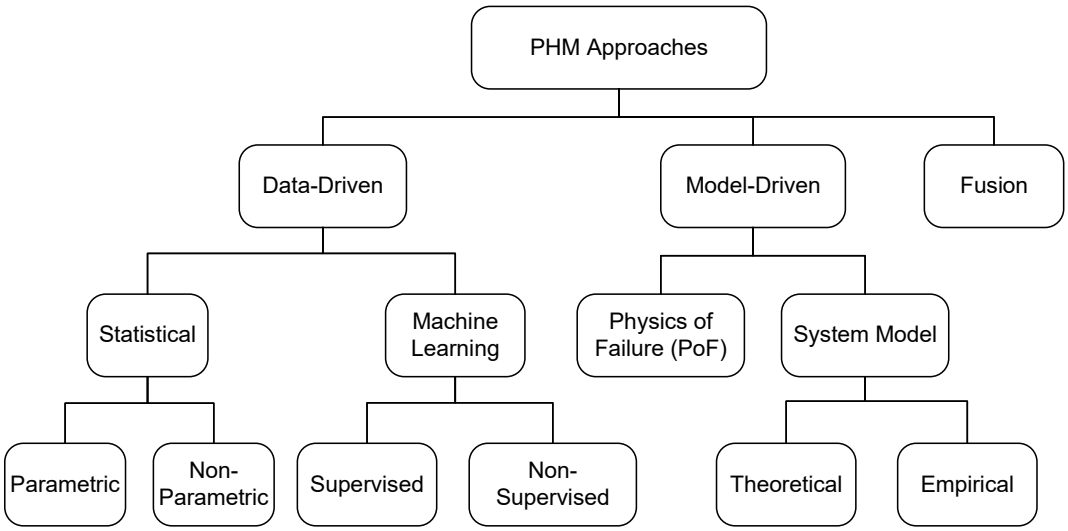

Figure 2.2: Classification of PHM approaches

2.1.3 PHM for Electronic Systems

Embedded systems are classified as dependable systems, meaning they must be resilient to failures, mishaps, and attacks [6]. A comprehensive review of dependable systems is provided in Section 2.2.
Although prognostics and health management (PHM) have been widely used in other areas, their application to electronic systems has only emerged recently [1], [2], [3] and [4]. According to [1], no review papers provide a simple and intuitive overview of the application of intelligent algorithms in PHM for electronics.The challenges are diverse, including scale and complexity, varying load cases, failure modes, computational costs, and efficient data acquisition.

Therefore, defining prognostics levels for electronics becomes very relevant. Six prognostics levels for electronics have been defined in the literature [1], [5] and [46]. A "PHM Functional Model" is outlined in [5] as a reference framework for performing engineering design analysis at the device, component, assembly, subsystem, system, or system-of-systems level. This model aims to characterize existing foundational layers, design integration strategies, and identify necessary, and potentially missing, functions required to implement PHM. Further details about these levels are provided in Table 2.2.

Table 2.2: Electronic Levels Description

Level	Prognostic	Level Description
0	Device	Basic level of any electronic system, i.e. chip and on-chip areas. Circuits and metallization failures come under this level.
1	Component	Components that are not expected to function individually if they are disassembled. Passive components such as resistors, capacitors, IGBTs, MOSFETs, wirebond, lead-frames belong to this level.
2	Assembly	Interconnections such as solder balls, leads along with circuit board. Key sites on the circuit boards like vias, traces, through hole, and pads are also included in this level.
3	Subsystem	This level includes an individual component like video cards, hard drives and power supply systems. This also covers connections for PCBs, enclosure elements, and supportive chassis.
4	System	An entire, modular electronic product, for example, electronic control units, computers, high-level controllers and others belong to this category
5	System-of-systems	It is a system of electronic systems. It consists of several level 4 products connected by physical or digital communication system. Connections between systems are also labelled as level 5 in this study.

2.1.4 System-Level Approach

As industrial systems become more complex, consisting of multiple interconnected components, system-level prognostics is gaining increasing attention from both industry and academia [47]. Machines equipped with sensors and actuators are essential for executing physical processes [48]. The distinction between component-level and system-level prognostics is summarised in [47]. Examples of system-level prognostics include systems such as aircraft engines, robotic arms, and rotating machinery. These systems require multiple sensors for operation and typically present limited failure data from the field. Since model-based methods are often unavailable, data-driven methods are preferred. These approaches use data collected from installed sensors to build mathematical models for estimating RUL [47].

There is a limited number of machine learning applications for system-level PHM of electronic systems, according to [1]. This highlights the challenges of applying PHM at the system level for comprehensive monitoring. Most regression-based algorithms in use are related to level "2" and level "1" prognostics. Filter-based methods are also quite popular in both of these levels.
Generally, much of the research in failure diagnostics and prognostics of assets focuses on component-level health assessment. However, complex engineering systems consist of multiple interacting components, and the failure of one component can significantly impact the system's overall performance. Therefore, developing PHM methodologies for system-level monitoring is just as important as for component-level systems.
Although some papers address system-level PHM, many issues and challenges remain in developing adequate methodologies. Challenges for system-level PHM include model structure and parameter uncertainties, nonlinearity of system models, environmental effects, measurement noise, and interactions between component degradations [16], [20].
Based on current knowledge in the literature [8], [47], [49], it is noted that achieving high-level prognostic accuracy in integrated and automated systems remains a challenge.
Despite significant progress in PHM over recent years, many challenges persist, especially in automotive electronics. For instance, computational techniques need to be improved to handle large volumes of data more efficiently and accurately. Another challenge is dealing with unexpected new faults in automotive electronic components, which complicates predicting the RUL. Future research is expected to focus on developing comprehensive PHM systems that ensure accuracy, rather than merely detecting faults [3], [47].
PHM implementation levels in electronics are defined in [5]. Proper classification of an electronic system is essential to help manufacturers or users select the appropriate methodology for implementing PHM capabilities.

According to [5], subsystem-level PHM for electronics can be defined based on the functionality of a group of assemblies, such as printed circuit board assemblies (PCBAs), which are grouped together in a specific geographical location. Examples of subsystems include electronic control units (ECUs), transmitters, receivers, and power supplies. Yet, system of systems (SoS) refers to a set of independent and useful systems that, when integrated, deliver unique capabilities. Examples of systems

of systems include aircraft, trains, automobiles, seagoing vessels, and integrated communication, navigation, and identification systems. A review of system-level prognostics approaches is presented in [47].

At the component level, a single or a set of sensors — such as vibration, acoustic emission, and temperature sensors — can be used to monitor degradation. Since components are easier to test, a large amount of failure data can be collected from a testbed for algorithm development. Furthermore, dedicated algorithms can be developed for feature extraction specific to the target component.

On the contrary, system-level prognostics involve multiple sensors from various components, making it more complex. Dedicated algorithms may not function effectively across the entire system. Additionally, due to the complexity of the system, models are rarely available, meaning that data-driven methods may be the only viable option. Real operational failure data may be scarce or nonexistent, which further complicates system-level prognostics.

2.2 Dependable Systems

Dependability is defined as the ability of a system to deliver a service that can be justifiably trusted, or more specifically, as the ability of a system to avoid failures that are more severe or frequent than acceptable to users. It can also be understood as a system property that prevents failure in unexpected or catastrophic ways [50]. Embedded systems are examples of dependable systems, which shall be dependable from failures, mishaps and attacks [6]. The study of various types of embedded systems reveals that they share four common requirements: dependability, real-time constraints, resource consumption and long operational life-cycle.
Safety-critical systems are another category of dependable systems [51]. Safety-critical systems are those whose failure could result in loss of life, significant property damage, or harm to the environment [52], [53].
According to [54] dependability is an integrating concept that encompasses the following attributes:

Table 2.3: Dependability Attributes

Attributes	Definition
Availability	Readiness for correct service
Reliability	Continuity of correct service
Maintainability	Ability to undergo modifications and repairs
Safety	Absence of catastrophic consequences on the user(s) and the environment
Integrity	Absence of improper system alterations

For various types of systems, dependability, according to [54], has the five attributes defined above, threats (that may affect a system during its entire life) and means (to attain the envisaged dependability). Threats to dependability are classified as failures, errors or faults. A service failure, or simply "failure", is an event that occurs when the delivered service deviates from correct service. Since a service is a sequence of the system's external states, a service failure means that at least one (or more) external states of the system deviates from the correct service state.
The deviation is called an error. The adjudged or hypothesised cause of an error is called a fault. Faults can be internal or external of a system. Means have been developed to attain the various attributes of dependability. They are grouped in

four major categories, namely fault prevention, fault tolerance, fault removal and fault prediction (forecasting). Means are further described below:

- **Fault Prevention:** means to prevent the occurrence or introduction of faults. It is considered to be part of general engineering practices. Fault prevention schemes are usually implemented during specification, design, implementation and tests.
- **Fault Tolerance:** means to avoid service failures in the presence of faults. Fault tolerance which is aimed at failure avoidance, is carried out via error detection and system recovery.
- **Fault Removal:** means to reduce the number and severity of faults. Fault removal is considered in two phases, during system development and during system use. Fault removal during the development phase of a system lifecycle consists of three steps: verification, diagnosis, and correction. These three steps are related to V&V (verification & validation) of the system, prior to its use. Fault removal during use of a system is corrective or preventive maintenance. Corrective maintenance aims to remove faults that have produced one or more errors and have been reported, while preventive maintenance is aimed at uncovering and removing faults before they might cause errors during normal operation.
- **Fault Prediction (or Forecasting):** means to estimate the present number, the future incidence, and the likely consequences of faults. Fault forecasting is conducted by performing an evaluation of the system behaviour with respect to fault occurrence or activation.

Fig. 2.3 shows the dependability tree with its attributes, threats and means.

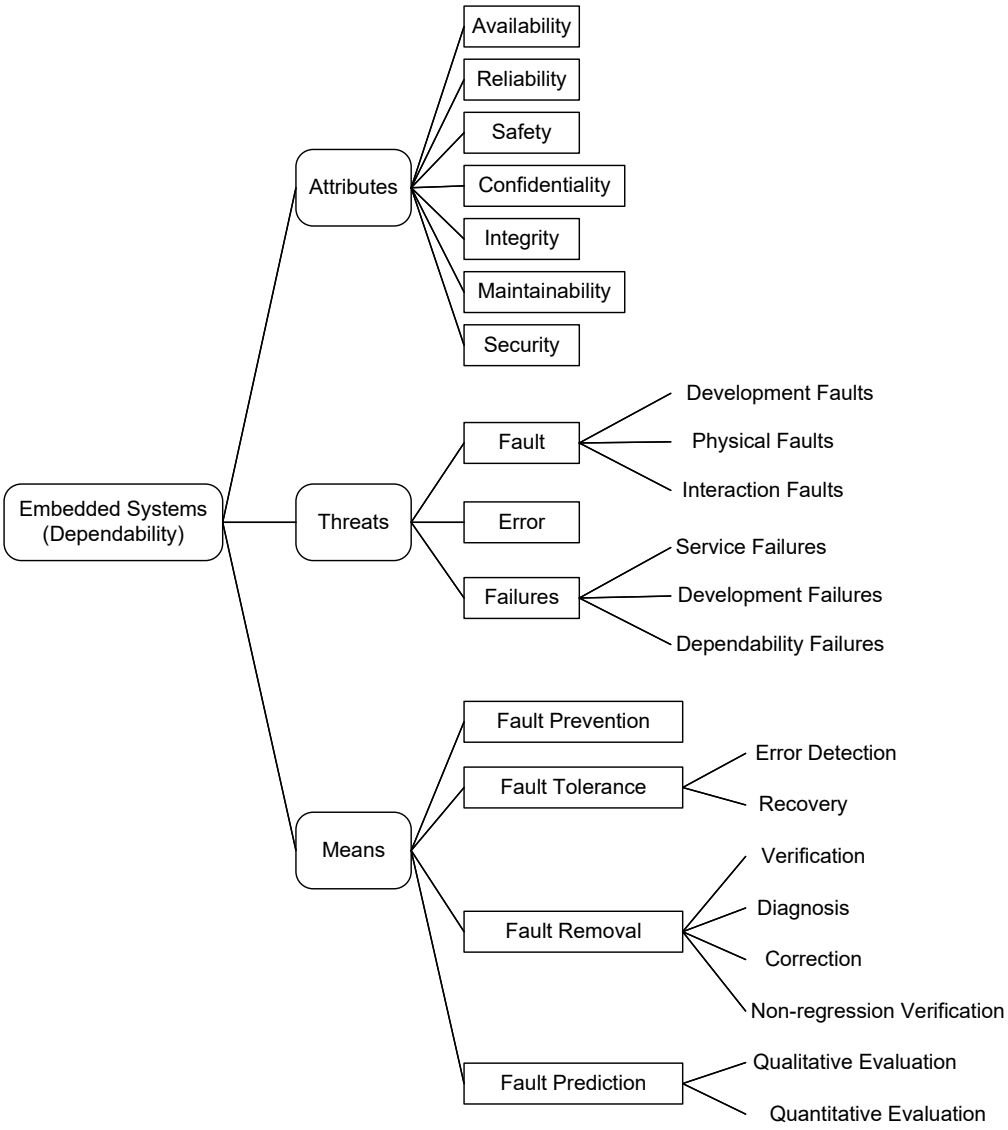

Figure 2.3: Dependability Tree (with Attributes, Threats and Means)

2.3 Taxonomy of Faults

At this point, it is important to define and differentiate terms that are misused throughout the literature. Definitions below are taken from [55], IEC standard area 192 - Dependability.

- **Fault:** is the event of an item characterised by its inability to perform a required function. This excludes the deliberate disabling of the item to

accommodate planned activities, such as planned maintenance, or due to a lack of external resources. [IEC 192-04-01]. After fault, the item may have a failure. Fault is an event and is distinguished from a failure which is a state.

- **Failure:** is defined as the cessation of the ability of an item to perform its intended function to a defined accuracy [IEC 192-03-01].
- **Error:** Discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. [IEC 192-03-02].
- **Anomaly:** Any deviation from required, expected, or desired performance of the object system [5].

Fig. 2.4 depicts the relationship between the terms within this subsection.

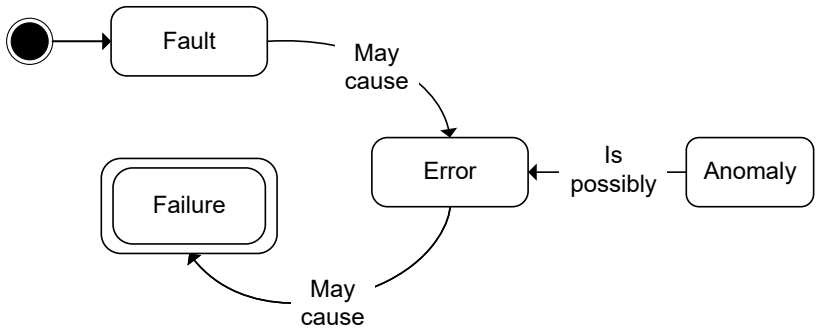

Figure 2.4: Failure, Fault, Error and Anomaly and their Relationship

2.4 Machine Learning Techniques Review

Machine learning (ML) is the scientific study of algorithms and statistical models that computer systems use to perform a specific task without being explicitly programmed [56].
Many studies have sought to review and categorize machine learning algorithms for various applications. A non-exhaustive list of such works which provides insights into the classification and suitability of different ML approaches across multiple domains can be found in [1], [8], [56], [57], [58], [59], [60], [61].
The majority of the literature classifies machine learning algorithms into four main categories: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Each category encompasses different approaches and techniques tailored to specific types of tasks and data availability. Fig. 2.5 shows the machine learning algorithms with their categories and relevant methods.

Supervised learning is a machine learning paradigm that involves learning a function that maps an input to an output based on sample input-output pairs. It relies on labeled training data, where each input example is paired with the correct output, allowing the model to infer patterns and relationships. This approach is widely used in tasks such as classification and regression, where the goal is to predict outcomes for new, unseen data [58].
Unsupervised learning analyzes unlabeled datasets without requiring human supervision. It is commonly used to uncover hidden patterns, structures, and relationships within data. This makes it useful for extracting generative features, identifying trends, and exploring data distributions [56]. Key applications of unsupervised learning include:

- **Clustering:** for grouping similar data points.
- **Density estimation:** to understand data distribution.
- **Feature learning:** to identify important characteristics.
- **Dimensionality reduction:** to simplify datasets.
- **Association rule mining:** for discovering relationships between variables.
- **Anomaly detection:** to identify unusual patterns in data.

Semi-supervised learning is a hybrid approach that combines elements of both supervised and unsupervised learning by utilizing both labeled and unlabeled data.

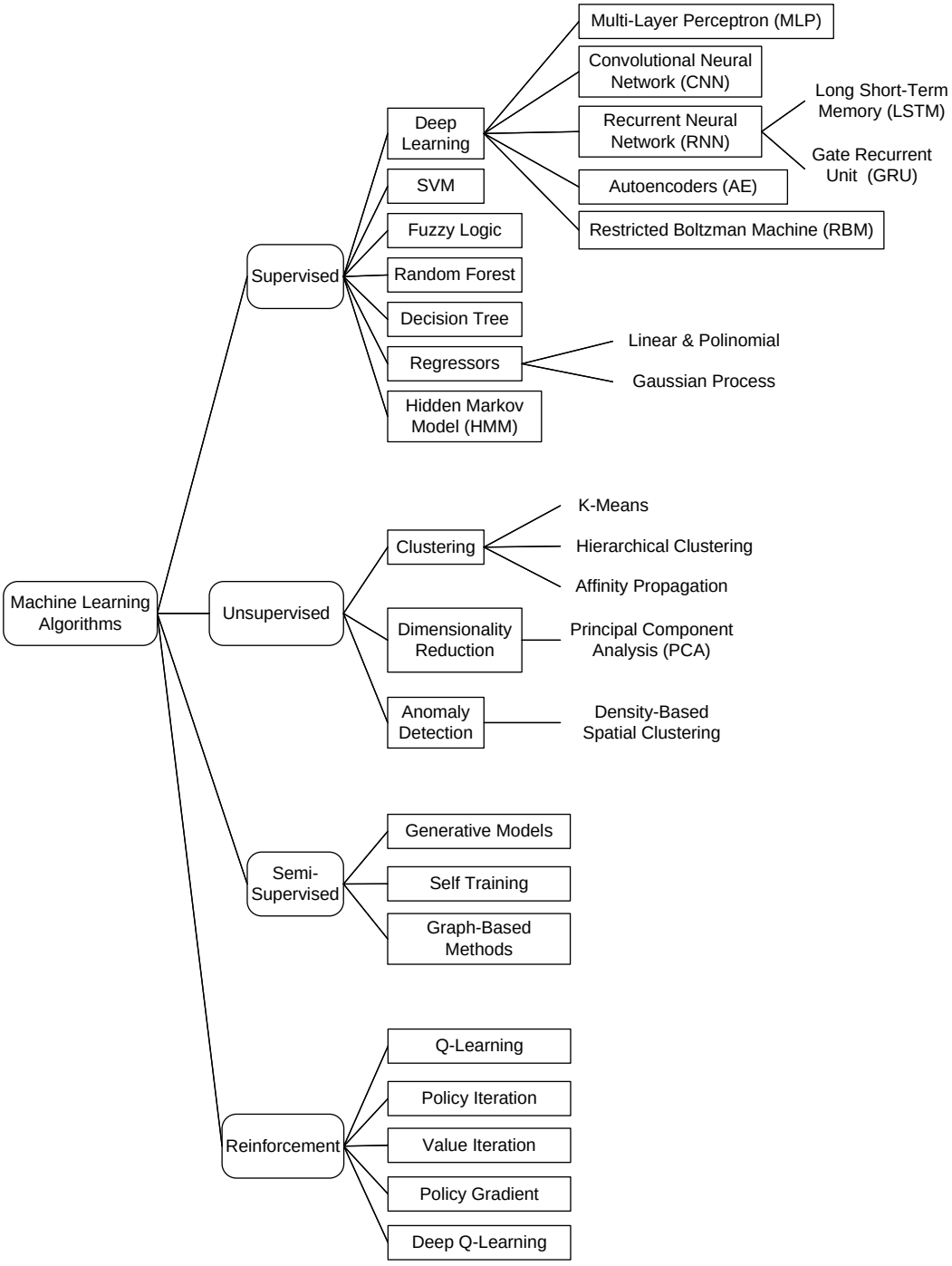

Figure 2.5: Machine Learning Algorithms

It serves as a middle ground between purely supervised and unsupervised learning, making it particularly useful when labeled data is scarce but unlabeled data is

abundant [58], [56]. Key applications of semi- supervised learning include machine translation, fraud detection, data labeling and text classification.

Reinforcement Learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment. The agent takes actions and receives feedback in the form of rewards or penalties, allowing it to optimize its strategy over time. Unlike supervised learning, RL does not rely on labeled datasets but learns through trial and error. Applications of reinforcement learning include:

- **Robotics:** for training robots to perform complex tasks autonomously.
- **Autonomous Vehicles:** helps self-driving cars make navigation and control decisions.
- **Manufacturing & Supply Chain:** optimizes warehouse automation and logistics.
- **Finance and Trading:** optimizes portfolio management and automated trading strategies.

Supervised machine learning is the most widely used paradigm in Prognostics and Health Management (PHM) due to its ability to learn patterns from historical labeled data and make accurate predictions about system health and failures [1]. Among various supervised learning approaches, deep learning algorithms have gained the most popularity, ranking as the top AI method in PHM research. Neural network-based techniques, such as Artificial Neural Networks (ANNs) and Recurrent Neural Networks (RNNs), are frequently employed in PHM applications due to their ability to model complex, nonlinear relationships in time-series data [8].
The success of deep learning in PHM is largely attributed to its ability to handle large volumes of sensor data, extract hidden patterns, and adaptively improve fault detection and failure prediction models.

2.5 Machine Learning in PHM

Data-driven techniques in PHM have traditionally relied on statistical methods. However, with advancements in sensor technology and signal processing, artificial intelligence (AI) techniques have gained increasing popularity [8]. Today, AI and machine learning (ML) algorithms are integrated into PHM approaches, enabling more complex fault diagnosis [1], and gradually ML is becoming a cornerstone in PHM reflecting the potential for innovative advancements in future PHM development [62]. The availability of abundant data and exponentially increasing computational power provide significant opportunities to develop advanced data-driven frameworks to determine the patterns, classify faults and assess system degradation trends [63].

A review of machine learning algorithms adopted for the PHM of electronic systems is presented in [1]. The authors were not able to find significant number of ML applications in the system level - it partially shows the difficulty of entire system monitoring. Regression-based methods are used for levels 1 (component) and 2 (assembly) electronics' PHM: a Gaussian Process (GP) regression is used to estimate RUL of a power MOSFET (level 1) under thermal load and to evaluate time to failure in solder joints (level 2) using radio-frequency (RF) impedance. The paper covers deep learning methods, including multi-layer perceptron (MLP), convolutional neural networks (CNN), recurrent neural networks (RNN), and long short-term memory (LSTM), under the artificial neural network (ANN) category. This broad classification makes it challenging to accurately attribute the most suitable method for each reviewed application. The paper cites a number of applications using ANN-category to monitor capacitors, IGBTs and other components as well as ANN to predict creep strain accumulation in solder joints.
A comprehensive review of state-of-the-art machine learning techniques adopted for PHM in engineering systems is presented in [8]. The review highlights deep learning architectures as particularly attractive compared to traditional machine learning algorithms, as deep learning eliminates the need for hand-crafted features. Instead, the network learns features directly from the input data. The authors rank deep learning algorithms as the most commonly used AI methods in PHM research, followed by hybrid/fusion approaches, ensemble techniques, and support vector machines (SVM) in that order. The deep learning algorithms that have been used for PHM include Autoencoder, Restricted Boltzmann Machine (RBM),

Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Gated Recurrent Units (GRU). A combination of these algorithms has also been used to solve PHM problems, such as CNN-LSTM for wind turbines blade icing monitoring. For the hybrid/fusion model, the paper cites a hybrid PHM approach for bearing degradation monitoring, obtained from training an adaptive predictive model on the available data and then adopting a regression-based approach to predict the RUL. Finally, examples of SVM is used to estimate the RUL of bearings utilising feature reduction technique and to predict RUL for turbofan engines where a number of points in the data were missing.

A systematic literature review (SLR) on machine leaning in the field of PHM of industrial mechanical systems and equipment is presented in [34]. According to the authors, machine learning is arising as one of the major approaches for PHM and RUL estimates. The main research question proposed was what are the most used ML algorithms for PHM of industrial systems. The results obtained throughout a consolidated methodology showed that 82% of 50 analysed papers used deep learning for the PHM of the mentioned systems. Deep learning in the paper are based on neural networks with multiple hidden layers: deep neural networks (DNN), RNNs and its variants, CNNs, autoencoders, RBMs and hybrids, which is a combination of these algorithms. Moreover, among the deep learning techniques, CNN and RNN resulted as to be the most applied. The study still covers the main performance metrics of ML algorithms adopted in PHM of these industrial systems. The study outcome showed that for classification tasks, accuracy is the most used KPI, whereas for regression, RMSE, MAE and MAPE are more balanced used.

A comprehensive overview of the current state-ofthe-art machine learning approaches for diagnostics and prognostics (PHM) of industrial systems is presented in [62]. It emphasizes the momentum for implementing ML in industrial settings, leveraged by the advent of the Industry 4.0 and Internet of things (IoT), for a strategic shift towards ML-based enhanced predictive maintenance.

An in-depth quantitative analysis of the development trends of PHM methods in general and their AI-based approaches is reported by [61]. The paper also proposes insightful AI-based implementation guidelines for different applications and datasets available. One guideline for selecting heath indicators is proposed: it basically takes the data dimensionality, amount of data and need to integrate expert knowledge to recommend between different approaches - statistical, deep learning, genetic programming or mathematical model. A very useful guideline for selecting neural

networks based fault detection and diagnostics methods is also presented, where a high amount of labeled data is basically classified between three main characteristics: sequential data, very high dimensional data, highly nonlinear and complex dynamic systems. For sequential data, the guideline recommends LSTM to be used, for very high dimensional data, CNN and for nonlinear complex systems neuro-fuzzy.

Data-driven approaches depend on historical system data [3]. The data is normally expected in the form of time series. A time series is a time-oriented or chronological sequence of observations on a variable of interest [64]. Time series data are the data collected sequentially in chronological order and at regular time intervals, and when this order or sequence of elements matter, it is called sequential data. This is the case for the electronic systems envisaged in this thesis, where data is collected as they are made available on a sequential and timely order. For the modern deep learning algorithms just mentioned, CNN is mainly used for images or video data [61], therefore, LSTM appears as a good candidate for the type of data handled in this thesis.
LSTM has been used in multiple applications for fault prediction and diagnosis and RUL estimation [57]. As a representative deep learning structure, LSTM is very powerful in discovering the changing patterns behind time series and is often used to deal with the long-term dependence of time series data [65]. LSTM networks possess feedback connections, hence classifying them as a variant of recurrent neural network (RNN). The utilization of LSTM enables the processing of not just individual data points, but also full sequences of data. PHM plays a vital role in ensuring the safety and reliability of aircraft systems. Within the field of aviation prognosis, LSTM networks have demonstrated considerable efficacy in the analysis and prediction of aircraft trajectory data [66]. A study evaluated various prognostic models for estimating the RUL of aviation engines. The models effectiveness were compared against an LSTM-based approach. Using the C-MAPSS (Commercial Modular Aero-Propulsion System Simulation) simulation dataset, the study demonstrated that a modified LSTM with an attention mechanism outperformed the other models, improving predictive accuracy, leading to superior RUL estimation.
A PHM dynamic predictive maintenance framework based on a real application case, with multiple sensor measurements of a turbofan engine to determine RUL, using LSTM with promising abilities for industry applications is presented in [67]. A key advantage of LSTM is its ability to learn from long time sequences and retain memory. This makes it particularly useful for system prognostics, as it can analyze

the history of degradation processes and effectively track system states for accurate RUL prediction.
Bidirectional LSTM (Bidi-LSTM) is used to determine probabilistic forecasts for short-term scheduling in electricity power markets for one day-ahead operational planning [68]. Two different models for characterizing the uncertainty are compared. In this way, the Bidi-LSTM network is trained to either generate a Gaussian or a non-parametric predictive distribution of the dependent variables. It enables to confront the Gaussian assumption of prediction errors with an empirical approach that makes no assumption on the underlying probability distribution of variables.
A LSTM network with two bidirectional layers is used for the fault diagnosis of a hydraulic pump in [69]. Initially, feature engineering is used over 11 available measurements, such as pressure, flow, temperature and vibration. Statistical values, as mean, variance, autocorrelation and others are synthetically added to the dataset, and then a process called fault sensitivity analysis is carried out, which utilises an additive feature attribution method called SHAP (SHapley Additive exPlanation) [70], [71] to extract the features really meaningful to the task. Finally, the outcome of the LSTM is combined with transfer learning [72], [73] for an improved result.
A new bidirectional LSTM architecture, called bidirectional handshaking LSTM, is also proposed by [44] for predicting the RUL when given short sequences of monitored observations with random initial wear. The handshake is done by interconnecting the forward and backward states. This enables the LSTM network to gain deeper insights when identifying sequence trends in both directions. Additionally, the handshaking mechanism facilitates collaboration between forward and backward units, enhancing the learning process and improving results.

As cited above in [69], fault sensitivity analysis, causality and explainability are important aspects for machine learning algorithms application and justification, especially for critical operations. Complex ML models have become less interpretable, often functioning as black boxes. This lack of transparency can pose challenges for predictions in various systems where safety and reliability are essential, including autonomous driving, aeronautics and nuclear generation for instance. In nuclear power, strict regulatory oversight requires clear explanations of predictive model outputs. Providing these explanations fosters stakeholder trust, meets regulatory requirements, and supports more informed operational decisions [74].
Explainable AI (xAI) tackles these challenges by developing machine learning (ML)

techniques that either provide insights into black box models or design inherently more transparent models [75]. xAI aims to make AI methods more transparent, explainable, and understandable to end users, stakeholders, and non-experts, fostering trust in AI systems [76]. And yet, xAI refers to the development of AI systems and algorithms designed to provide clear and understandable explanations for their decisions and predictions [77].

Two xAI approaches are found in the literature, intrinsic and extrinsic (or post-hoc) models. Intrinsic approach use algorithms that produce interpretable models - inherently transparent and understandable by design, while post-hoc methods, which are independent of the underlying predictive model, are applied after training to interpret and explain the predictions of complex, opaque models. Examples of intrinsic models are linear regression, logistic regression, tree-based models and others. Post-hoc explainable approaches can be used to interpret more complex nonlinear models, black box models where it is not possible to understand the underlying decision-making process. Examples of post-hoc explainability techniques found in the literature are functional decomposition, individual conditional expectation (ICE), partial dependence plot (PDP), global surrogate, local surrogate (LIME), Shapley values and its derivative SHAP (SHapley Additive exPlanations) [75], [77], [78].

LIME and SHAP emerge for xAI of regression time series. LIME (Local Interpretable Model-agnostic Explanations) generates local explanations by approximating a complex model's decision boundaries with a simpler, more interpretable model near a specific data instance. It is model-agnostic and can be applied to any machine learning model, regardless of its algorithm or architecture [77]. SHAP determines feature importance by measuring each feature's contribution to the model's output. Based on coalitional game theory, it treats features as players in a coalition. SHAP enhances transparency by generating SHAP values for each data instance [74]. A further review of SHAP method is provided in Chapter 4, subsection 4.2.2. A further review of related works on explainable Artificial Intelligence (xAI) applied to PHM presented in the aforementioned literature, especially in [75], [76] and [78].

2.6 Multi-Step Ahead Forecasting

The ultimate goal in time series analysis is the prediction of future values by comprehending the sequence of past observations, accumulated in historical data [79]. Time series forecasting is a growing field of interest playing an important role in nearly all fields of science and engineering [80]. One area where time series forecasting has proven to be particularly beneficial is in the energy sector [81], [82], [83], [84]. For these applications, deep learning algorithms based on back-propagation through time (BPTT) such as recurrent neural networks (RNN) have been proven very efficient to deal with the temporal features of energy datasets with variable sequence lengths over different time horizons [81].

Energy Load Forecasting (ELF) is critical for the operation and design of power systems. It enables utility providers to model electricity consumption and prepare for future power loads. It also assists Distribution System Operators (DSOs) in managing and matching future energy generation with consumption. Various types of ELF are vital for the continuous and efficient operation of power systems. Short-Term Load Forecasting (STLF) ensures operational security and power system savings, Medium-Term Load Forecasting (MTLF) supports planning and operation, and Long-Term Load Forecasting (LTLF) aids in planning future investments in power system infrastructure. In [83], one-step ahead (OSA) forecasting, regardless of the timestep (15 minutes, one hour, one day, etc.), is demonstrated with high accuracy, offering applicability across various forecasting scenarios. The study compares five forecasting models (MLP, LSTM, XGBoost, SVR, and LR) along with three ensemble models (EAP, EWA, and EPE).

While most literature on power forecasting focuses on one-step ahead predictions, these are insufficient for long-term planning applications like generation scheduling, where bids are submitted a day in advance [81]. Moreover, with predictions for multiple timesteps, the variability and potential anomalies in the grid can be analysed more effectively to detect faults. However, forecasting longer time horizons with multi-step ahead predictions presents significant challenges.

2.6.1 Multi-Step Ahead Strategies

Unlike one-step ahead forecasting, multi-step ahead forecasting involves more complexities, such as error accumulation, reduced accuracy, and increased uncertainty. This subsection explores the strategies for MSA forecasting and the associated challenges.
Multi-step ahead (MSA) forecasting has a wide range of real-world applications [22], including models for predicting abnormal physiological signal levels, flood forecasting using RNNs, nitrogen oxide emission forecasting, electric power load forecasting, and even earthquake and seismic response predictions. In the context of electrical power dispatching, a Bidi-LSTM was used to schedule power demand and dispatching with a one-day ahead window in [68], while wind power forecasting was proposed using LSTM in [82]. According to [80], there are five primary strategies for performing MSA forecasting tasks: Recursive, Direct, Multi-Input Multi-Output (MIMO), and the derived DirRec and DirMO strategies:

- **Recursive:** This method is also called iterated, because it iterates H (horizon) times a one-step ahead forecasting model to obtain the H forecasts. After estimating the future series values, it is fed back as an input for the following forecast.
- **Direct:** The Direct strategy, however estimates a set of H forecasting models, each returning a forecast for the ith value.
- **Multi-Input Multi-Output (MIMO):** The MIMO strategy returns a vector of future values in a single step, in order to preserve, between the predicted values, the stochastic dependency characterizing the time series. This approach replaces the H models of the direct approach by one multiple-output model [85].
- **DirRec:** It is a combination of the direct and recursive strategies. A different model is used at each step but the approximations from previous steps are introduced into the input set.
- **DirMO:** It is a combination of the direct and miMO strategies. This strategy aims to find a trade-off between the property of preserving the stochastic dependency between the forecasted values and the flexibility of the modelling procedure.

In this thesis, the Recursive and the MIMO strategies were implemented and

tested. The reason being is that the Recursive strategy is widely used and a single model is trained, which performs a one-step ahead forecast and then recursively sweeps the entire forecasting horizon. The MIMO strategy, likewise uses a single model, but returns a vector of the future values in a single step. It also takes into account the stochastic dependence between variables and their future values, which affects the forecast accuracy [86]. Finally, these strategies are executed in real-time environments and have to comply with the system dynamics and run cycles.

Conventional approaches to MSA prediction, like iterated and direct methods, belong to the family called Single-Output prediction strategy, since they both model from data a multiple-input single-output mapping. A MIMO method belong, however to a Multiple-output prediction strategy, where the returned prediction is not a scalar but a time series itself. A review of single-output versus multiple-output approaches in [85] showed multiple-output approaches as a promising alternative to conventional single-output strategies. A study of the properties of iterated and direct multi-step forecasting techniques in the presence of in-sample location shifts (breaks in the mean) is performed in [87]. For this particular case, the work shows that direct strategy provides prediction values that are relatively robust and the benefits increases with the prediction horizon.
A multi-output (MIMO) RNN forecasting architecture, where explicit temporal dependencies between outputs capture the relationship between the predictions is proposed in [88]. This work introduces and differentiates two forecasting methods, the recursive and the multi-output. Recursive forecasting is the primary form of multi-step forecasting [80], where forecasting is done by the factorization of previous values, amplifying potential errors and leading to lower quality predictions as the time horizon increases. The MIMO forecasting aims to estimate the future values in one step. Multi-output approaches sidestep the issue of error feedback by jointly estimating over the prediction window [88].

2.6.2 Data-Driven Methods for MSA Forecasting

Several models and architectures have been proposed for time series forecasting. From state-space model to neural networks, many approaches have been used. Seasonal ARIMA (SARIMA) is used as a state-space model to compare with deep learning methods in [89]. Sequence-to-sequence RNNs and LSTMs have become

popular due to their ability to learn temporal patterns and long range memory.
Multi-step time series predictions based on pure LSTM have been investigated by [27], [90] and [91]. In [90], a multi-step input method is used with a stacked-LSTM network. The future *n-steps* are predicted by the parallel input of *m-steps* of the time series. In [27], a standard LSTM model is implemented and tested on three different datasets. The results are compared with ARIMA and RNN models. The conclusion is that the LSTM model can fit a wider range of data patterns compared to the traditional models and no additional time is spent on the modelling process, usually required by the statistical models, such as stability checking, autocorrelation and partial autocorrelation function checking. The downside is that LSTM requires more resources for the forecast process.
In [91], a comprehensive survey of anomaly detection for system-level anomalies using LSTM networks is presented. The same type of anomaly found in the dataset of Chapter 5 is defined as collective anomaly. According to the screening, a number of LSTM architectures have been used, but not especifically the Bidi-LSTM.
Further MSA forecasting using machine learning techniques in the energy sector are found in [68], [81], [82], [92] and [93]. A long-term forecasting strategy for multi-step ahead (MSA) predictions with no prior knowledge is proposed in [81]. This method incorporates a Bayesian probabilistic approach with bidirectional LSTM (Bidi-LSTM) for renewable generation forecasting, using solar generation data. The results show the effectiveness of point and probabilistic predictions, with lower error values. A MSA forecasting for electric power load using different models for comparison, such as ARIMA and others not investigated herein, such as Prophet and its variations [94] can be found in [92]. LSTM still features amongst the best results, considering the MAPE (mean absolute percentage error) of the models. In [82], MSA wind power forecasting is done based on LSTM or GRU (gated recurrent unit), which is relatively simpler variant of LSTM. Results are compared with ARIMA and SVM methods, with predominance of the former ones.
A MSA forecasting strategy is presented in [82] using LSTM and GRU on a recursive mechanism. The study estimates wind power forecasts up to four steps ahead, comparing results with ARIMA (Autoregressive Integrated Moving Average) and SVM methods. The LSTM and GRU models show superior performance.
Multi-step forecasting with deep learning is used for short-term load forecasting for energy and distribution management in power grid in [93], which proposes a TCMS-CNN (multi-scale CNN with time-cognition) model to integrate multi-scale

convolutions and periodical coding into an end to end trainable neural network, which optimises the structure of CNN and extracts more relationships with periodical characters raising the accuracy of multi-step load forecasting. Bidi-LSTM is used to determine probabilistic forecasts for short-term scheduling in electricity power markets for one day-ahead operational planning [68]. Then, two different models for characterizing the uncertainty are compared. In this way, the Bidi-LSTM network is trained to either generate a Gaussian or a non-parametric predictive distribution of the dependent variables. It enables to confront the Gaussian assumption of prediction errors with an empirical approach (that makes no assumption on the underlying probability distribution of variables).

Considering the implementation of a PHM framework for batteries' state of charge estimation, it was noted that not many works have been found for a MSA prediction of the Li-ion batteries state of charge (SOC) though. An ARIMA-NARX model, which is capable of predicting SOC for higher charging/discharging rates (C-rates) is proposed in [24]. NARX stands for nonlinear autoregressive network with exogenous inputs. The use of different multi-step prediction techniques for long-term prognosis of the Lithium-ion batteries condition is presented in [25]. The paper proposed the use of adaptive neuro-fuzzy inference systems, random forest, and group method of data handling, along with various MSA strategies: iterative, direct and DirRec (which is a combination of the former ones). These methods were then evaluated for prediction of capacity over the long horizon. A novel machine learning enabled method to perform real-time multi-forward-step SOC prediction for battery systems using a recurrent neural network with LSTM is presented in [26].
Most of the SOC predictions in published studies are basically single-step predictions (estimates) based on experimental data. By using a multi-forward-step SOC prediction for battery systems, battery anomalies/faults caused by SOC anomalies (such as low SOC, SOC jumping, etc.) can also be diagnosed in advance, thereby avoiding more serious battery faults/failures or even battery thermal runaways.

2.7 Chapter Summary

This chapter focuses on the current state of Prognostics and Health Management (PHM) for electronic systems, highlighting that this field is not fully developed yet. While traditional statistical methods have been commonly used, the chapter discusses how machine learning (ML), particularly deep learning techniques, is emerging as a promising alternative. However, there are few reported applications of ML at the system level, which presents an opportunity for further research and exploration. The growing availability of data is further driving the shift towards supervised ML algorithms. This context sets the foundation for the research question [**RQ1**], which will be explored in detail throughout this thesis.

Recent studies on fault sensitivity analysis, causality, and explainability have been published to support the application and justification of machine learning algorithms, particularly in critical operations. Explainable AI (xAI) helps to address the black-box nature of machine learning by making AI methods more transparent, understandable, and interpretable to end users, stakeholders, and non-experts, thereby fostering trust in AI systems. For time-series regression, explainability techniques such as LIME (Local Interpretable Model-agnostic Explanations) and SHAP (SHapley Additive exPlanations) have gained prominence, with SHAP being the most widely cited method in the literature. Explainability is also an important topic of this research, particularly in relation to research question [**RQ2**], which aims to apply the developed machine learning algorithms in applications requiring high dependability and reliability.

This thesis introduces a novel approach to multi-step-ahead (MSA) forecasting for embedded systems. This chapter begins with a general literature review of MSA forecasting applications, followed by an overview of foundational methods and data-driven techniques used for MSA estimation. Current approaches predominantly rely on statistical methods, with machine learning techniques—especially deep learning—becoming increasingly prevalent for MSA forecasting. Furthermore, the research on MSA prediction of the state-of-charge (SOC) of Li-ion batteries is limited, a gap that will be comprehensively addressed in Chapter 6.

The next chapter provides the mathematical foundation for this thesis, exploring both model-based and data-driven approaches in detail. For the model-based

approach, a state-space representation is selected and Kalman filters are used to estimate the system states at real-time. For data-driven approach, a bidirectional LSTM, as a representative of the recursive neural networks, is selected to estimate the same system states, but purely based in the input/output mapping.

Chapter 3

Theoretical Background

This chapter provides the theoretical foundation for the subsequent chapters, introducing two modelling approaches: model-based and data-driven. The model-based approach is formulated using state-space notation, with the Kalman filter selected as the state estimator in two variations: iterated and unscented. The data-driven approach leverages recurrent neural networks (RNNs) and explores their theoretical basis as universal approximators for both linear and complex nonlinear systems. Finally, long short-term memory (LSTM) networks and their variant, bidirectional LSTM (Bidi-LSTM), are introduced as specialized RNN architectures designed to handle sequential data and mitigate the vanishing gradient problem affecting traditional methods.

3.1 Model-Based and Data-Driven Approaches

Process modelling is based on a mathematical framework that characterizes the dynamic behaviour of a system, assuming input and output measurements are available and supplemented by prior knowledge of the process [12]. There are two primary types of models commonly used in control and identification theory: state-space models and input-output models. The state of a dynamic system refers to a set of variables, known as state variables, which are the minimum set essential in determining the system's behavior. These variables, when combined with the

knowledge of the system's input for a given time $t \geq t_0$, allow to fully understand and predict the system's behaviour for any time $t \geq t_0$ [95]. The state-space approach assumes full or partial accessibility of the system's states, enabling the derivation of a mathematical model, which can also incorporate uncertainties in both the model and measurements. A well-established state estimator to determine the system's current state is the Kalman Filter, further detailed in this chapter.

This thesis focuses on complex nonlinear systems, for which accurate modelling remains a significant challenge in practical applications [96]. In contrast to model-based approaches, data-driven methods assume that the state space is inaccessible and use the input-output measurements available [1], [11]. Additionally, unlike model-based models, data-driven methods are often regarded as black-box approaches due to the absence of explicit equations. These black-box systems map input features to a target output without revealing the underlying reasoning, leaving little room for interpretability [3], [97].

Data-driven approaches are well-suited for complex nonlinear systems, as they can capture correlations between parameters, interactions among subsystems, and the effects of environmental factors using in situ system data. These methods can model degradation characteristics based on historical sensor data, uncovering underlying relationships and causalities while enabling the estimation of system information such as remaining useful life (RUL) [13]. Recently, there has been increasing interest in developing mathematical models based solely on observed data. Neural models, in particular, play diverse roles, serving as simulators for fault detection, models for controllers, and more [12].

In the previous chapter, several authors proposed the use of deep learning algorithms to address complex nonlinear dynamic systems, as discussed in [8], [34], [61], and others. Therefore, this chapter provides a further review of deep learning algorithms, with a particular focus on recurrent neural networks (RNNs) and their variants.

3.2 Model-Based Approach

Notation:
It shall be noted that the following notation is generally referred in this thesis when representing dynamic systems, such as batteries in Chapter 6.

- $u(t)$ or u_k as system input
- $x(t)$ or x_k as system internal states
- $h(t)$ or h_k as hidden states, applicable for neural networks
- $y(t)$ or y_k as system output
- t as timestep in a continuous-time system
- k as timestep in a discrete-time system

A state-space representation of a particular dynamic system can be given by a differential equation as seen in [11, eq. (1)] and also discussed in [95]:

$$\dot{x} = f[x(t), u(t)] \tag{3.1}$$

$$y(t) = g[x(t), u(t)] \tag{3.2}$$

where, $x(t) \in \mathbb{R}^m$ represents the system state vector at time $t \in \mathbb{R}^+$, the vector $u(t) \in \mathbb{R}^l$ is the input and $y(t) \in \mathbb{R}^n$ is the output.

The function $f(\cdot)$ is a mapping from $\mathbb{R}^m \times \mathbb{R}^l$ to $\mathbb{R}^m$, and $g(\cdot)$ is also a mapping from $\mathbb{R}^m \times \mathbb{R}^l$ to $\mathbb{R}^n$.

Both $f(\cdot)$ and $g(\cdot)$ can be either linear or nonlinear; multiple-input/multiple-output (MIMO) and/or time invariant or time varying systems.

The continuous-time system introduced in (3.1) and (3.2) can be represented as a discrete-time system by the following equations, as seen in [11, eq. (2)] :

$$x_{k+1} = f[x_k, u_k] \tag{3.3}$$

$$y_k = g[x_k, u_k] \tag{3.4}$$

where, $k \in \mathbb{Z}^+$ and the inputs, internal states and outputs are discrete sequences.

Nonlinear discrete system dynamic model resolution involving $f(\cdot)$ and $g(\cdot)$ are not trivial, even when these functions are known. Consider the following nonlinear system, described by the differential equation and the observation model with additive noise, as seen in [98, eq. (3.13) and (3.14)]:

$$x_{k+1} = f(x_k, u_k) + w_k \tag{3.5}$$

$$y_k = g(x_k, u_k) + v_k \tag{3.6}$$

where $\mathbf{x}_k \in \mathbb{R}^m$ is the system state vector at time index k.

The nonlinear system is also depicted in the block diagram of Fig. 3.1, as illustrated in [99, Fig. 4].

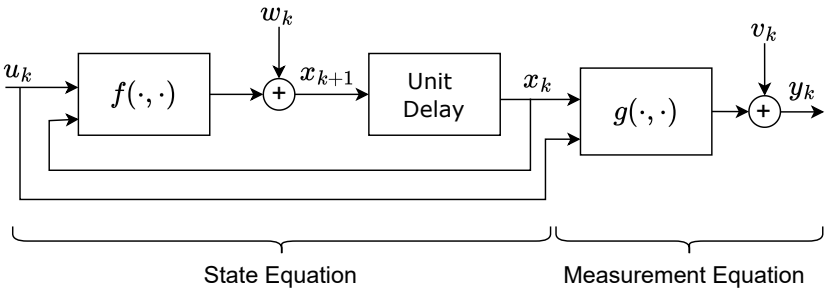

Figure 3.1: Nonlinear Discrete-Time System Block Diagram

The initial state $\mathbf{x}_0$ is a random vector with known mean $\mu_0 = \mathbb{E}[\mathbf{x}_0]$ and the state error covariance is given by $P_0 = \mathbb{E}[(\mathbf{x}_0 - \mu_0)(\mathbf{x}_0 - \mu_0)^T]$, as stated in [98].

The input, $u_k \in \mathbb{R}^l$ and the measured output $y_k \in \mathbb{R}^n$, also called observation vector are known variables of the system.

The vectors w_k and v_k represent uncertainties in the model and the measurement noise, respectively. It is further assumed that the vectors w_k and v_k consist of zero-mean white Gaussian distributed noise, stochastic processes temporally uncorrelated, with known covariances, as seen in [98, eq. (3.3) and (3.4)]:

$$\mathbf{Q}_k = \mathbb{E}[w_k w_k^T] \tag{3.7}$$

$$\mathbf{R}_k = \mathbb{E}[v_k v_k^T] \tag{3.8}$$

The system depicted above, whose hidden states can be modelled and estimated, is subjected to a number of estimation algorithms. An example very diffused in

the literature is the Kalman Filter (KF). Kalman filters are used to estimate states based on linear dynamical systems in state-space format. The Kalman Filter uses a set of differential equations to model and predict the state of a physical system. It minimizes the error between the measured and predicted outputs of a linear system by adjusting the state variables [98]. The following subsections detail two different types of KF, the Iterated and the Unscented versions.

3.2.1 Iterated Extended Kalman Filter

According to [99], the Kalman filter problem is to use the entire observable system, consisting of the data $\{u_0, u_1 \ldots u_k\}$ and $\{y_0, y_1 \ldots y_k\}$ to find the minimum mean squared error estimate $\hat{x}_k$ of the true state x_k. Therefore, the objective function is:

$$\hat{x}_k = \underset{\hat{x}\in\mathbb{R}^n}{\arg\min}\, \mathbb{E}[(x_k - \hat{x})(x_k - \hat{x})^T]\Big| \begin{Bmatrix} u_0 \; u_1 \ldots \; u_k \\ y_0 \; y_1 \ldots \; y_k \end{Bmatrix} \tag{3.9}$$

Assuming the nonlinearities in the dynamic and the observation models are smooth, the functions $f(\cdot)$ and $g(\cdot)$ are differentiable and can be expanded in Taylor Series. If the time deviation of these two functions is small, they can be approximated by the first order Taylor expansion. Hence, the Extended Kalman Filter (EKF) is also called First Order Filter.

At each timestep, $f(\cdot)$ and $g(\cdot)$ are linearised by a first order Taylor-series expansion. EKF assumes that, at all operating points of x_k and u_k, these functions are differentiable. Then, as seen in [99, eq. (5) and (6)]:

$$f(x_k, u_k) \approx f(\hat{x}_k, u_k) + \frac{\partial f(x_k, u_k)}{\partial x_k}\Big|_{x_k=\hat{x}_k} (x_k - \hat{x}_k) \tag{3.10}$$

$$g(x_k, u_k) \approx g(\hat{x}_k, u_k) + \frac{\partial g(x_k, u_k)}{\partial x_k}\Big|_{x_k=\hat{x}_k} (x_k - \hat{x}_k) \tag{3.11}$$

The partial derivatives of $\partial f(\cdot)/\partial x$ and $\partial g(\cdot)/\partial x$ in (3.10) and (3.11) are the Jacobian matrices, called herein A_k and H_k, respectively.

Combining these equations with (3.5) and (3.6), it comes the linearised equations

describing the true system state as a function of itself, known inputs and states u_k, $\hat{x}_k$ and unmeasurable noise inputs w_k and v_k:

$$x_{k+1} \approx \hat{A}_k x_k + f(\hat{x}_k, u_k) - \hat{A}_k \hat{x}_k + w_k \tag{3.12}$$

$$y_k \approx \hat{H}_k x_k + g(\hat{x}_k, u_k) - \hat{H}_k \hat{x}_k + v_k \tag{3.13}$$

In the EKF, $g(\cdot)$ is linearised about the predicted state estimate $\hat{x}_k$. The Iterated EKF tries to linearise it about the most recent estimate, improving therefore its accuracy.

EKF uses a two-step prediction-correction algorithm: The discrete-time EKF computes two different estimates of the state and covariance matrix each sampling interval. The first estimate, $\hat{x}_k^-$, is based on the prior state estimate (also called *a priori* estimate) as computed in the previous iteration, $\hat{x}_{k-1}^+$. At this stage, the state covariance matrix, $\hat{P}_k^-$ is also computed. The *a priori* estimate is calculated before the up-to-dated measurement to be made.

The second step estimate, $\hat{x}_k^+$ (also called *a posteriori* estimate) fine tunes the first estimate after measuring the system input u_k and the measured output y_k. At this stage the Kalman gain K is computed and the new prediction for $\hat{x}_k^+$ and $\hat{P}_k^+$ is provided.

This is achieved at each iteration by the following calculation sequence, as presented in [99, Tab. 5]:

Initialisation:
For $k = 0$, set

$$\hat{x}_0^+ = \mathbb{E}[x_0] \tag{3.14}$$

$$\mathbf{P}_0 = \mathbb{E}[(\mathbf{x}_0 - \hat{x}_0^+)(\mathbf{x}_0 - \hat{x}_0^+)^T] \tag{3.15}$$

Now, for $k = 1, 2, \ldots,$ compute

Prediction (*a priori* estimate):

$$\hat{x}_k^- = \mathbf{A}_{k-1}\hat{x}_{k-1}^+ + \mathbf{B}_{k-1}u_{k-1} \tag{3.16}$$

$$\mathbf{P}_k^- = \mathbf{A}_{k-1}\mathbf{P}_{k-1}^+\mathbf{A}_{k-1}^T + \mathbf{Q} \tag{3.17}$$

Correction Gain:

$$y_k = \mathbf{H}_k\hat{x}_k^- + \mathbf{D}_k u_k \tag{3.18}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R})^{-1} \tag{3.19}$$

Correction (*a posteriori* estimate):

$$\hat{x}_k^+ = \hat{x}_k^- + \mathbf{K}_k(U_k - y_k) \tag{3.20}$$

$$\mathbf{P}_k^+ = \mathbf{A}_{k-1}\mathbf{P}_{k-1}^+\mathbf{A}_{k-1}^T + \mathbf{Q} \tag{3.21}$$

3.2.2 The Unscented Kalman Filter

The usage of the Extended Kalman Filter (EKF) may lead to inaccuracies and divergence of the filter due to errors in linearization and the neglect of higher-order derivatives in the Taylor approximation. To address these issues, the Sigma Point Kalman filter (SPKF) has been developed. Unlike the EKF, the SPKF does not require derivatives and approximates linearization using a set of sigma points [98]. The SPKF offers an alternative approach for state estimation in nonlinear systems. Instead of using Taylor series expansions to approximate covariance matrices, the SPKF performs multiple function evaluations to compute an estimated covariance matrix [100]. The unscented Kalman filter(UKF) is a typical type of SPKF. The Unscented Transformation (UT) is a technique used to compute the statistics of a random variable that goes through a nonlinear transformation. [101], [102], [103].

Equations (3.5) and (3.6) also apply to more general cases where the system is nonlinear, the state is not Gaussian-distributed, and the noise influence is nonlinear as well.

The process and measurement noise can be added to the unscented transformation to provide an augmented state, as seen in [103, eq. (3)]:

$$\mathbf{x}_k^{aug} = [\mathbf{x}_k^T \;\; \mathbf{w}_{k-1}^T \;\; \mathbf{v}_k^T]^T \tag{3.22}$$

Considering the propagation of a random variable $\mathbf{x}$ (dimension n) through the nonlinear function in (3.5) and $\bar{\mathbf{x}}$ and $\mathbf{P}_\mathbf{x}$ the mean and covariance of $\mathbf{x}$, respectively. Then, to compute the statistics of $\mathbf{y}_k$ in (3.6), a matrix $\mathbf{X}$ of $2n+1$ *sigma* vectors (with corresponding weights W_i) is used, according to the reasoning below.

Consider $\mathcal{X}_{k-1}$ as a set of $2n+1$ sigma points, where n represents the dimension of the state-space, with their associated weights given by [103, eq. (4)]:

$$\mathcal{X}_{k-1} = \{(\mathbf{x}_{k-1}^j, \mathbf{W}^j) | j = 0 \dots 2n\} \tag{3.23}$$

A selection that incorporates higher order information in the selected points is then considered for the sigma points, as seen in [103, eq. (5) to (10)]:

$$\mathcal{X}_0 = \bar{\mathbf{x}} \tag{3.24}$$

$$-1 < \; W^0 \; < 1 \tag{3.25}$$

$$\mathbf{x}_{k-1}^i = \bar{\mathbf{x}}_{k-1} + \left(\sqrt{\frac{n}{1-W^0}\mathbf{P}_{k-1}}\right)_i \quad i = 1, \cdots, n \tag{3.26}$$

$$\mathbf{x}_{k-1}^{i+n} = \bar{\mathbf{x}}_{k-1} - \left(\sqrt{\frac{n}{1-W^0}\mathbf{P}_{k-1}}\right)_i \quad i = n+1, \cdots, 2n \tag{3.27}$$

$$W^j = \frac{1-W_0}{2n} \quad j = 1, \cdots, 2n \tag{3.28}$$

$$\sum_{j=0}^{2n} W^j = 1 \tag{3.29}$$

W^0 controls the position of sigma points around the mean $\bar{\mathbf{x}}$: $W^0 \geqslant 0$ points tend to move further from the origin, $W^0 \leqslant 0$ points tend to be closer to the origin.

The nonlinear propagation of the sigma point is [103, eq. (11)]:

$$\hat{\mathbf{x}}_k^j = f(\mathbf{x}_{k-1}^j) \tag{3.30}$$

The transformed points are used to compute the mean and covariance of the forecast value of $\mathbf{x}_k$, as seen in [103, eq. (12) and (13)]:

$$\hat{\mathbf{x}}_k = \sum_{j=0}^{2n} W^j \hat{\mathbf{x}}_k^j \tag{3.31}$$

$$\hat{\mathbf{P}}_k = \sum_{j=0}^{2n} W^j \{\hat{\mathbf{x}}_k^j - \bar{\mathbf{x}}_k\}\{\hat{\mathbf{x}}_k^j - \bar{\mathbf{x}}_k\}^T + \mathbf{Q}_{k-1} \tag{3.32}$$

The sigma points are then propagated through the nonlinear observation model [103, eq. (14)]:

$$\hat{y}_{k-1}^j = g(x_{k-1}^j) \tag{3.33}$$

And then calculate the target values mean and variance [103, eq. (15) and (16)]:

$$\bar{y}_{k-1} = \sum_{j=0}^{2n} W^j \hat{y}_{k-1}^j \tag{3.34}$$

$$Cov(\tilde{y}_{k-1}) = \sum_{j=0}^{2n} W^j (\hat{y}_{k-1}^j - \bar{y}_{k-1})(\hat{y}_{k-1}^j - \bar{y}_{k-1})^T + \mathbf{R}_k \tag{3.35}$$

And then cross-covariance between $\hat{x}_k$ and $\hat{y}_{k-1}$ is [103, eq. (17)]:

$$Cov(\tilde{x}_k, \tilde{y}_{k-1}) = \sum_{j=0}^{2n} W^j (\hat{x}_k^j - \hat{x}_k)(\hat{y}_{k-1}^j - \hat{y}_{k-1})^T \tag{3.36}$$

The estimate update has the following form, with updates of $\bar{x}_k$, $\mathbf{K}_k$ and $\mathbf{P}_k$ [103, eq. (18)]:

$$\bar{x}_k = \hat{x}_k + \mathbf{K}_k(y_k - \hat{y}_{k-1}) \tag{3.37}$$

The gain $\mathbf{K}_k$ is given by [103, eq. (19)]:

$$\mathbf{K}_k = Cov(\tilde{x}_k^f, \tilde{y}_{k-1}^f) Cov^{-1}(\tilde{y}_{k-1}) \tag{3.38}$$

The posterior covariance is updated after the following formula [103, eq. (20)]:

$$\mathbf{P}_k = \hat{\mathbf{P}}_k^f - \mathbf{K}_k Cov(\tilde{y}_{k-1}) \mathbf{K}_k^T \tag{3.39}$$

For model-based prognostics and health management at the system level of multiple engineered systems using the Kalman Filter, several references can be found in the literature, including applications in fault detection and diagnosis [104], [105] and remaining useful life (RUL) prediction [106], [107].

It is worth noting that when using a model-based approach for PHM with Kalman filter estimation, Section 3.2 remains fully applicable. A state-space model must be established for the system under evaluation. Therefore, equations in the form of (3.5) and (3.6) must be defined, explicitly specifying the system states.
Then, equations (3.14) to (3.21) have to be established, for the iterative Kalman fiter, for instance. This will cover the four main steps of the KF recursive process: initialisation, *a priori* estimate, correction gain and *a posteriori* estimate. Evaluation metrics are those typically used for a regression model, such as root mean square error (RMSE) and mean absolute error (MAE) [104].

3.3 The Data-Driven Approach

The classic machine learning problem involves determining a nonlinear mapping, (3.40) as in [102, eq. (3)], where $G(\cdot)$ is the nonlinear mapping between input and output and is parameterised by the vector $\mathbf{w}$.

$$y_k(\mathbf{x}) = G(\mathbf{x}_k, \mathbf{w}) \tag{3.40}$$

The vector $\mathbf{w}$ herein represents the weights of the input-output nonlinear connections. The equation (3.40) can be graphically represented by Fig. 3.2, where $\mathbf{x}_k$ represent a matrix or vector containing real values each variable contained in $\mathbf{x}$.

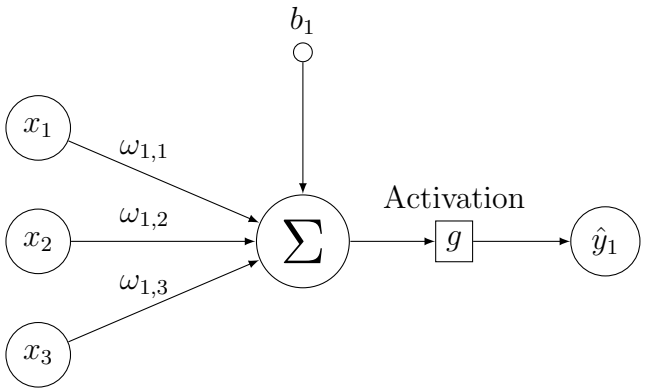

Figure 3.2: An Abstract Neuron - Input-Output Nonlinear Mapping

The nonlinear mapping can take the form either of a feedforward neural network (FNN) or recurrent neural network (RNN), where the weights $\mathbf{w}$ represent the connections. Fig. 3.3, based on [12, Fig. 6.1], depicts examples of the feedforward network (also called multiple layer perceptron). If there are more than one hidden layer in a neural network, the neural network is a deep neural network (DNN). Multiple layer perceptron (MLP) networks are loop-free and fully connected. Unlike MLPs, the recurrent neural networks (RNN) have feeback loops at the hidden states, which allows the RNNs to keep longer sequences in memory to process at the current time.

This type of map has a wide range of applications in classification, regression and dynamic modelling. Learning refers to the process of estimating the parameters involved. Typically, a training set is provided, consisting of known input-output pairs. The machine's error is defined as the difference between the measured and the

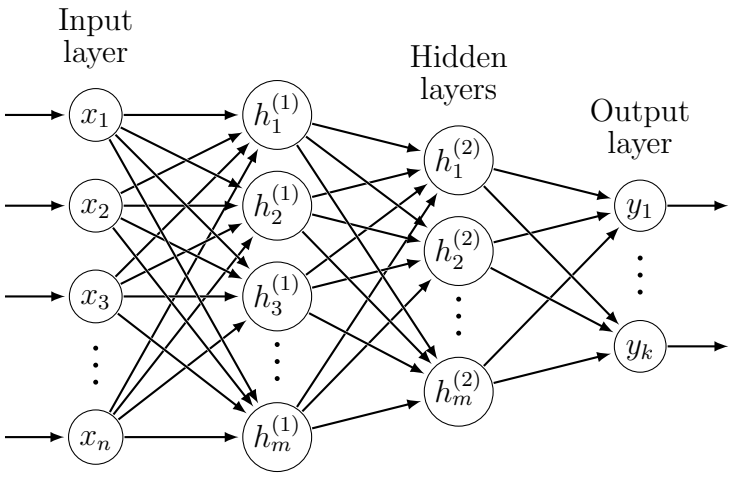

Figure 3.3: Feedforward Network (also called Multiple Layer Perceptron)

estimated output. The objective of learning is to find the values of the parameters $\mathbf{w}$ that minimize the expected squared error, and this can be achieved by a number of optimization approaches, including gradient descent using back-propagation [102].

It shall be noted that on feedforward neural networks, data flows in a single direction, resulting in loss of information from previous layers. There is no internal state or memory, preventing therefore, the network from retaining knowledge. However, in recurrent neural networks, data traverses a loop, allowing it to not only incorporate new information but also remember past data. RNNs are dynamic systems, having an internal state at each timestep. This is achieved through circular connections between neurons in higher and lower layers, as well as optional self-feedback connections. By utilizing these feedback connections, RNNs are able to carry information from earlier events to current processing steps. As a result, RNNs are capable of creating a memory of time series events.

Herein, Recurrent Neural Networks (RNNs) are the preferred architecture due to their superior capability to process sequential data and effectively learn from long time-series datasets

3.3.1 The RNN as an Universal Approximator

A neural network can be applied to both linear and nonlinear systems and do not require modelling neither an intrinsic knowledge of the system-of-interest [108].

According to [109], a multilayer neural network can approximate continuous functions (or systems) provided the number of the hidden nodes is sufficiently large.
However, this approach has also some disadvantages. Neural networks require a labor-intensive and computationally demanding training process, necessitating a large amount of data from the system of interest. Additionally, the trained model is typically specific to the system it was designed for.
Consequently, if the system's dynamics, internal states, or input-output data profiles change, the network will likely need to be retrained to produce accurate parameters [108].
Neural networks are widely used nowadays as an alternative for system identification and modelling, allowing to treat the system as a black box [11], [110].

Two works were conclusive to demonstrate that standard multilayer feedforward networks are capable of approximating any continuous function of n real variables from one finite dimension space to another one, to any desired degree of accuracy, provided sufficiently many hidden units are available [111], [112]. The sets of theorems and corollaries established in these works, led to the universal approximation theorem, as below:

Definition 3.1. *An activation function is defined as sigmoidal (σ) if*

$$\sigma(k) \rightarrow \begin{cases} 1 & as\ k \rightarrow +\infty \\ 0 & as\ k \rightarrow -\infty \end{cases} \tag{3.41}$$

Theorem 3.1. *The Universal Approximation Theorem by Superposition of Sigmoidal Function [111]*

Let $\sigma(k)$ be a nonlinear, bounded and monotonically increasing continuous function. Let I_n indicate the n-dimensional unit cube which represents the cartesian product of unit intervals $[0,1]^n$. The space of continuous functions on I_n is referred as $C(I_n)$. Then, for every continuous function $f \in (I_n)$ and $\epsilon > 0$, there exists a sum, $y(\mathbf{x})$

such that $\| y - f \| < \epsilon$ *for all* $x \in I_n$*:*

$$y(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i \sigma(\mathbf{w}_i^T \mathbf{x} + b_i) \tag{3.42}$$

where $i = 1, \ldots, N$*;* σ *is a sigmoidal function,* $\mathbf{w}_i$ *and* $\mathbf{x} \in \mathbb{R}^n$*. In mathematical words, functions of form* $y(\mathbf{x})$ *are dense in the function space* $C(I_n)$*.*

In summary, the Universal Approximation Theorem [112], [113], [114] states that there is a neural network capable of approximating any continuous function on $I_n = [0, 1]^n$ with great accuracy. To achieve this, the number of hidden nodes, learning iterations, weight parameters, and biases must be adjusted specifically for the function being approximated.

Other results followed, like [115], where the Universal Approximation Theorem was introduced for width-bounded ReLU networks.
Neural networks are commonly described by their architecture, which is determined by the network's width and depth. The depth h of a network refers to the number of layers it has, including the output layer but excluding the input layer. On the other hand, the width d_m of a network is defined as the maximum number of nodes in any given layer. The input dimension, or the number of input nodes, is represented by the symbol n.

Instead of using a sigmoidal activation function, it is proposed in [115] the use of a ReLU (Rectifier Linear Unit) activation function.

Definition 3.2. *An activation function is defined as Rectifier Linear Unit (ReLU :* $\mathbb{R} \to \mathbb{R}$*) if*

$$ReLU(k) \to \begin{cases} k & as\ k > 0 \\ 0 & as\ k \leq 0 \end{cases} \tag{3.43}$$

Theorem 3.2. *Width-bounded ReLU Networks as Universal Approximator*
For any Lebesgue-integrable function $f : \mathbb{R}^n \to \mathbb{R}$ *and any* $\epsilon > 0$*, there is a fully connected ReLU neural network A with width* $d_m \leq n + 4$*, such that the function*

F_A which represents this network satisfies

$$\int_{\mathbb{R}^n} |f(x) - F_A(x)|dx < \varepsilon \tag{3.44}$$

In summary, the theorem states that there is a deep ReLU neural network capable of approximating any continuous function $f : \mathbb{R}^n \to \mathbb{R}$ with great accuracy, for any width $\leq n + 4$.

3.3.2 Recurrent Neural Network Selection

A recurrent neural network (RNN) is a neural network that is capable of simulating a discrete-time dynamical system that has an input u_k, an output y_k and a hidden state h_k, where $k \in \mathbb{Z}^+$ represents the timesteps in a discrete-time system [116]. It also uses feedback loops to store past input information, hence reduce the complexity and number of layers in its structure [11].

The discrete-time dynamical system is defined by, as in [116, eq. (1) and (2)]:

$$h_k = g_1(u_k, h_{k-1}) \tag{3.45}$$

$$y_k = g_2(h_k) \tag{3.46}$$

$g_1(\cdot)$ and $g_2(\cdot)$ are nonlinear activation functions. Fig. 3.4, as illustrated in [11, Fig. 1], shows a simple RNN graphical structure.

Therefore, the RNN can be expanded to a general mathematical representation, as in [117, eq. (1) and (2)]:

$$h(k) = g_1(\sum_j w(u)_{ij} u_k + \sum_j w(h)_{ij} h_{k-1} + b_k) \tag{3.47}$$

$$\hat{y}(k) = g_2(\sum_j w(\hat{y})_{ij} h_k + b_y) \tag{3.48}$$

where $w(h)$, $w(u)$ and $w(y)$ are the respectively the hidden states, input and output weights. Finally, (3.47) and (3.48) can also be presented in a matrix form, as in

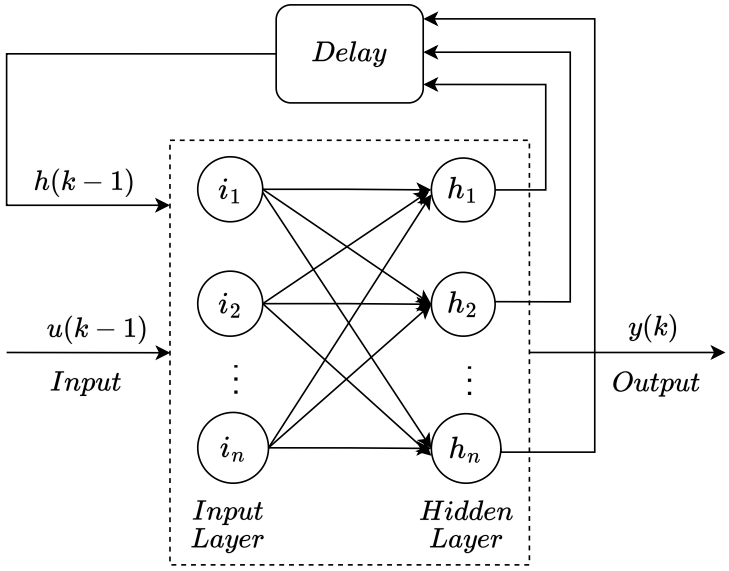

Figure 3.4: RNN Structure for a Discrete-Time Dynamical System

[117, eq. (1) and (2)]:

$$h_k = \mathcal{H}(\mathbf{W}_{uh} \cdot x_k + \mathbf{W}_{hh} \cdot h_{k-1} + b_h) \tag{3.49}$$

$$\hat{y}_k = \sigma_y(\mathbf{W}_{hy} \cdot h_k + b_y) \tag{3.50}$$

where $\mathcal{H}$ is a nonlinear activation function, usually an element-wise application of a sigmoid function, $\mathbf{W}_{uh}$, $\mathbf{W}_{hh}$ and $\mathbf{W}_{hy}$ are respectively the input-hidden layer, hidden-hidden layers and hidden-output layer weight matrices; b_h and b_y are the hidden and output layers bias vectors.
The activation function of the output layer is a sigmoid σ_y. It can be depicted that the output of a RNN $y(k)$ depends only on two parameters, its prior values y and the feedback internal hidden state $h(k)$.

3.3.3 Back-propagation and Vanishing Gradient Issue

The error back-propagation algorithm is the most commonly used technique for training neural networks. It employs gradient descent to adjust the weights in multilayer networks. This learning process occurs in small iterative steps, moving from the output layer back to the input layer. However, it is essential for the

activation function of the neuron to be differentiable in order for this algorithm to work effectively [118].
Back-propagation through time (BPTT) RNN algorithm uses both current and prior inputs each timestep as a new input for the network [119]. The back-propagation algorithm is used to find a local minimum of the error function of the network. The error function, or training loss for a set $S = (u_k, y_k)_{k=1}^{n}$ is denoted by $L_S(\theta)$. The selected loss function is the mean square error (MSE), also known as L2 loss, as seen in [17, eq. (2)]:

$$E = \mathcal{L}_S(y, \hat{y}) = \frac{1}{m}\sum_{k=1}^{m}\|y_k - \hat{y}_k\|^2 \tag{3.51}$$

where m is the entire observable system's data points, y are the observed (or true) values and $\hat{y}$ the predicted ones.

The network is initialised with randomly chosen weights. The gradient of the error function is computed and used to correct the initial weights recursively [120].
The weights in the network are the only parameters that can be modified to make the quadratic error E as low as possible. Gradient descent is used to minimize E:

$$\nabla E = (\frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \cdots, \frac{\partial E}{\partial w_l}) \tag{3.52}$$

Each weight w is updated using the increment:

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i} \text{ for } i = 1, \cdots, l \tag{3.53}$$

where η is the learning rate. Ultimately, the learning problem is reduced to the calculation of the network function gradient with respect to its weights to find a minimum of the error function where $\nabla E = 0$.

Standard RNN are limited to look-back in time for approximately ten timesteps [118]. Over many timesteps the error therefore typically explodes or vanishes. When it vanishes, it prevents the network from learning within an acceptable time period. Full theoretical background is provided in [121].

Herein, this issue is addressed with the use of Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN). LSTMs are well-suited for capturing nonlinear dynamics in time-series sensory data and learning effective representations of

machine conditions. Their ability to model long-term dependencies makes them superior to traditional RNNs. Due to this capability, LSTMs have been successfully applied in various fields, including speech recognition, image captioning, handwriting recognition, genomic analysis, and natural language processing.

A comprehensive review of machine learning algorithms, particularly deep learning (including LSTM) is provided in Chapter 2, Section 2.5.
LSTM and its relevance to this thesis, along with its applications in Prognostics and Health Management (PHM), are discussed in the following chapters. Chapter 4 details the proposed data-driven method and its adaptability to other PHM applications. Chapter 5 presents the first experimental implementation of the data-driven framework, with a full discussion of the methods and results.

3.4 Adapted Long Short-Term Memory (LSTM)

A solution that addresses the vanishing gradients issue for time-dependent and sequential data series described above is a method called long short-term memory (LSTM) [118]. Unlike traditional neural networks, the recurrent neural networks LSTM is an extremely efficient tool when the information is sequential [109]. This model replaces the traditional neuron of the perceptron with a memory block [122]. As stated in [118] and well difused in the literature, LSTM can learn how to bridge minimal time lags of more than 1,000 discrete timesteps.
LSTM demonstrates its effectiveness in tasks requiring the retention of a limited amount of data over a long period. That is thanks to the use of memory blocks, which have access control in the form of input and output gates, ensuring that only relevant information enters or exits the block. They also have a forget gate that assess the significance of the information stored within the cells. When certain cells no longer require previous information, the forget gate resets the state of those cells within the block. Moreover, forget gates enable continuous prediction, as cells can completely disregard their prior state, thus mitigating biases in prediction [118].
The LSTM structure calculation process, shown in Fig. 3.5 based on [109, Fig. 1], is such that at each time iteration k, the hidden layer maintains a hidden state h_k, and updates it based on the layer input u_k, and previous hidden state h_{k-1}.
In Fig. 3.5, $u(k)$ and $y(k)$ are measurable scalar input and output, respectively. LSTM has three gates to protect and control the cell state: forget gate $F(k)$, input gate $I(k)$, and output gate $O(k)$. Additionally, $\tilde{x}(k)$ and $I(k)$ are considered inner states.

Below are the governing equations referring to the gates, states and output [109]:

Gates:

$$f(k) = \sigma(\mathbf{W}_{hf} \cdot h(k-1) + \mathbf{W}_{uf} \cdot u(k) + b_f) \tag{3.54}$$

$$i(k) = \sigma(\mathbf{W}_{hi} \cdot h(k-1) + \mathbf{W}_{ui} \cdot u(k) + b_i) \tag{3.55}$$

$$o(k) = \sigma(\mathbf{W}_{ho} \cdot h(k-1) + \mathbf{W}_{uo} \cdot u(k) + b_o) \tag{3.56}$$

States:

$$\tilde{x}(k) = \tanh(\mathbf{W}_{hx} \cdot h(k-1) + \mathbf{W}_{ux} \cdot u(k) + b_x) \tag{3.57}$$

$$x(k) = f(k)x(k-1) + i(k)\tilde{x}(k) \tag{3.58}$$

Output:

$$y(k) = h(k) = o(k)\tanh(x(k)) \tag{3.59}$$

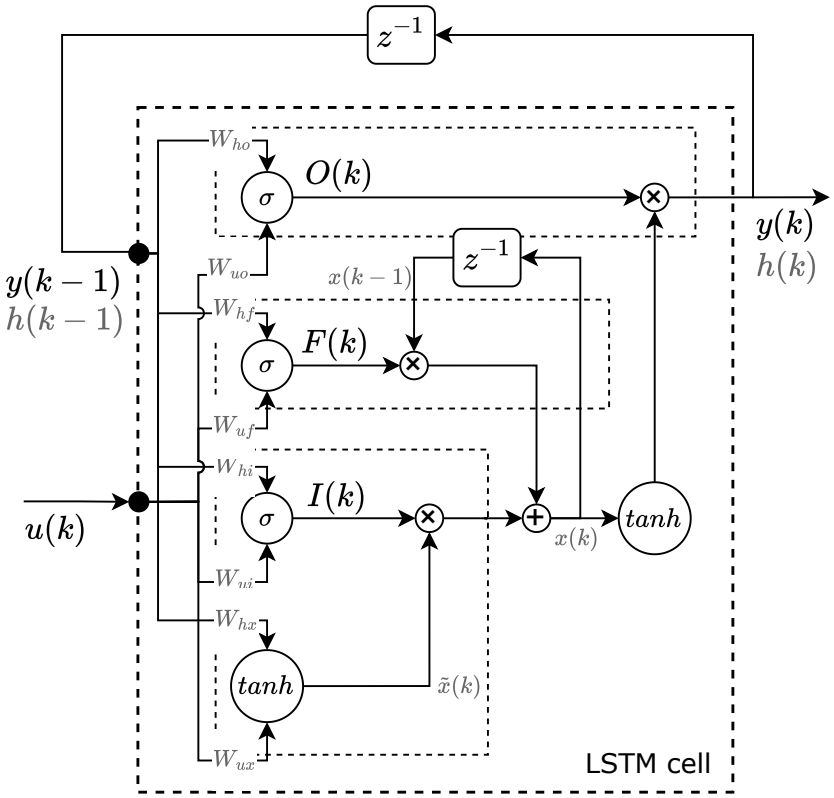

Figure 3.5: Long Short-Term Memory Cell

3.4.1 Multivariate Bidirectional Adapted LSTM

Traditional RNNs including unidirectional LSTM (also called forward-pass) are suitable for processing sequential data but are trained only in the forward path. Bidirectional learning trains on both forward and reverse paths with two separate hidden layers [123] and uses the output for prediction as well. Bidi-LSTM uses the information by independently calculating both the forward path and the reverse path [124]. The output, which results from the flow of information, is also used for learning so that features are better extracted and have higher accuracy than the existing LSTM.

Bidirectional LSTM removes the one-step truncation originally present in LSTM, and implements a full error gradient calculation. This full error gradient approach eased the implementation of bidirectional LSTM, and allowed it to be trained using standard BPTT [118].

As stated in (3.49), a RNN computes only the forward hidden sequence $\overrightarrow{h}$. By implementing the backward hidden sequence, $\overleftarrow{h}$, the output sequence y is obtained by iterating layers from $k = \{1 \dots n\}$ in the forward direction and $k = \{n \dots 1\}$ in the reverse direction [125], [117]. The formulation of the bidirectional LSTM backward direction is given in (3.60) and the output function is finally achieved in (3.61), as in [117, eq. (9) and (10)]:

$$\overleftarrow{h}_k = \mathcal{H}(\mathbf{W}_{x\overleftarrow{h}} \cdot x_k + \mathbf{W}_{\overleftarrow{h}\overleftarrow{h}} \cdot \overleftarrow{h}_{k-1} + b_{\overleftarrow{h}}) \tag{3.60}$$

$$\hat{y}_k = \sigma_y(\mathbf{W}_{\overrightarrow{h}y} \cdot \overrightarrow{h}_k + \mathbf{W}_{\overleftarrow{h}y} \cdot \overleftarrow{h}_k + b_y) \tag{3.61}$$

Fig. 3.6 shows an example of Bidi-LSTM architecture, containing a forward LSTM layer and a backward LSTM layer.

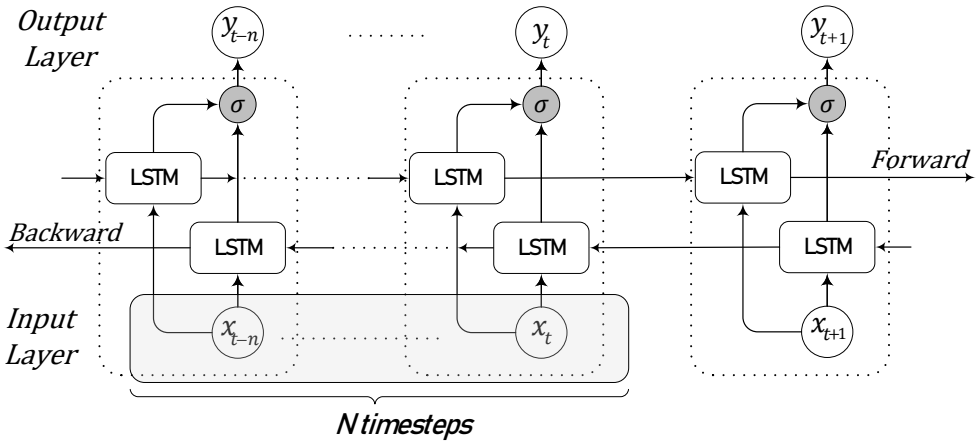

Figure 3.6: Bidirectional LSTM Architecture

For data-driven prognostics and health management at the system level of multiple engineered systems, including embedded ones, bidirectional LSTMs remain under-explored in the literature. Some references found during the literature review cover fault diagnosis applications [69], remaining useful life (RUL) prediction [13], [44], machine wear monitoring [126] and other applications such as speech recognition [117], traffic speed prediction [123].

The proposed Bidi-LSTM implementation for PHM in embedded systems employs multiple bidirectional LSTM fully connected layers. Deep architectures enable the network to learn higher-level representations of raw input data and are widely used due to their strong performance [13]. The implementation is discussed in more detail in the following chapters. As depicted in Fig. 3.6, the LSTM archicteture benefits from using N-timesteps as input, therefore for a time sequence series, a sliding window can be specified to meet the required performance criteria. An example of how the sliding window is implemented is given in [69] and further explored in the subsequent chapters of this thesis.

3.5 Chapter Summary

This chapter focuses on developing mathematical models for two distinct Prognostics and Health Management (PHM) approaches: model-driven and data-driven. As one of the objectives of this thesis is to assess whether data-driven machine learning algorithms can be applied to the PHM of electronic systems [**RQ1**], we propose testing our PHM framework on a system and comparing the results with a model-based approach to evaluate the consistency of the outcomes.

Aiming at generalisation, the proposed system is modelled as a nonlinear, multiple-input, multiple-output dynamic system, which represents the complexity of many real-world applications. A state-space representation was chosen, as it effectively models these systems using differential equations, accounts for uncertainties in both the model and measurement noise, and reveals the system's internal states, which are not directly observable as inputs or outputs. Kalman filtering techniques, specifically the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), were employed to estimate the system's states in real-time, based on the sequence of measurements observed over time. The theoretical foundations of both the EKF and UKF are thoroughly explained, with the relevant equations provided for their application in the second experiment in Chapter 6.

Next, the data-driven approach is discussed, aiming to model the same nonlinear systems addressed in the model-based approach. While a variety of machine learning algorithms were implemented and tested in this thesis, the focus was placed on supervised methods, particularly Recurrent Neural Networks (RNNs), which were identified in the literature review of Chapter 2 as promising techniques for PHM applications. A detailed justification for the preference of RNNs for the selected systems is provided, along with a discussion of the Long Short-Term Memory (LSTM) version of RNNs. LSTMs were introduced to address the vanishing gradient problem that affects standard RNNs when dealing with time-dependent and sequential data. Additionally, LSTMs demonstrate effectiveness in tasks that require retaining limited data over extended periods. Finally, a multivariate bidirectional LSTM is proposed for implementation in the subsequent chapters, owing to its enhanced ability to learn from both forward and reverse sequences, resulting in higher accuracy compared to traditional LSTMs.

This algorithm is implemented in both Chapter 5 and Chapter 6, serving as a candidate machine learning technique for the real-time PHM of embedded systems, as addressed in [**RQ3**].

The next Chapter 4 details the novel data-driven system-level PHM (Prognostics and Health Management) framework for real-time condition monitoring and prognostics of embedded systems.

Chapter 4

Novel Data-Driven PHM Methodology

This chapter introduces a novel data-driven Prognostics and Health Management (PHM) methodology based on bidirectional Long Short-Term Memory (LSTM) networks for embedded electronic systems. The proposed PHM technique enables real-time condition monitoring, fault diagnosis, and multi-step-ahead forecasting within a unified data pipeline, deployed directly on the edge device. Additionally, the framework is designed to manage noisy measurements, a common challenge in real-time systems due to sensor limitations. Feature engineering is applied to multivariate systems to extract the most relevant and impactful features, enhancing predictive performance.

The remainder of this chapter is structured as follows: first, the overall methodology is introduced, followed by a detailed review of the offline processing framework, which consists of four key steps—noise handling, feature engineering, model optimization, and cross-validation. Next, real-time inference is discussed, outlining the steps for deploying the optimized model on the target embedded hardware. Finally, multi-step-ahead forecasting is examined, considering real-time constraints and modeling approaches.

4.1 Method Overall Approach

Data-driven estimation methods have emerged with advancements in big data and computational power. As discussed in the previous chapter, these methods rely primarily on data without requiring prior knowledge of the underlying processes [127].
A stepwise approach is proposed, consisting of three main stages:

I. **Offline Processing:** This step involves deriving the optimal machine learning model for the system-of-interest.
II. **Real-Time Inference:** This step deploys the model on real-time data to determine the appropriate health indicator.
III. **Multi-Step Ahead Estimator:** This estimator provides a forecasted advisory window for the system's status, enabling condition-based maintenance.

Fig. 4.1 shows the proposed data-driven PHM methodology data flow diagram with its three main processes, which will be further detailed in the subsequent sections.

The proposed framework is a machine learning-based methodology for the PHM of embedded systems, aiming at generalisation and real-time performance.
This methodology is designed to be applied to a variety of systems with minimal or no modifications, regardless of the case study or dataset, as demonstrated in [17] and [124].
A robust machine learning model training process, performed offline, is proposed to ensure an optimized model is developed to fit the available system data. Once deployed, this model serves a dual purpose in real-time: a) system inference and b) system forecasting.

Considering a dataset as a collection of samples, as in [17], let (4.1) be a representative set at a given present time k. Defining the array $\mathbf{u}$ as a subset of $\mathcal{U}$ which represents a set of signal or sensors' measurements, also called data variables or independent features.

$$(\mathbf{u}, \mathbf{y}) = [(u_0, y_0),\ (u_1, y_1)\ \ldots\ (u_k, y_k)] \in \mathbb{R}^{m \times n} \tag{4.1}$$

Similarly, the vector $\mathbf{y}$ is a subset of $\mathcal{Y}$ and represents the class-of-interest, also called target or dependent variable. The goal of the proposed data-driven model is

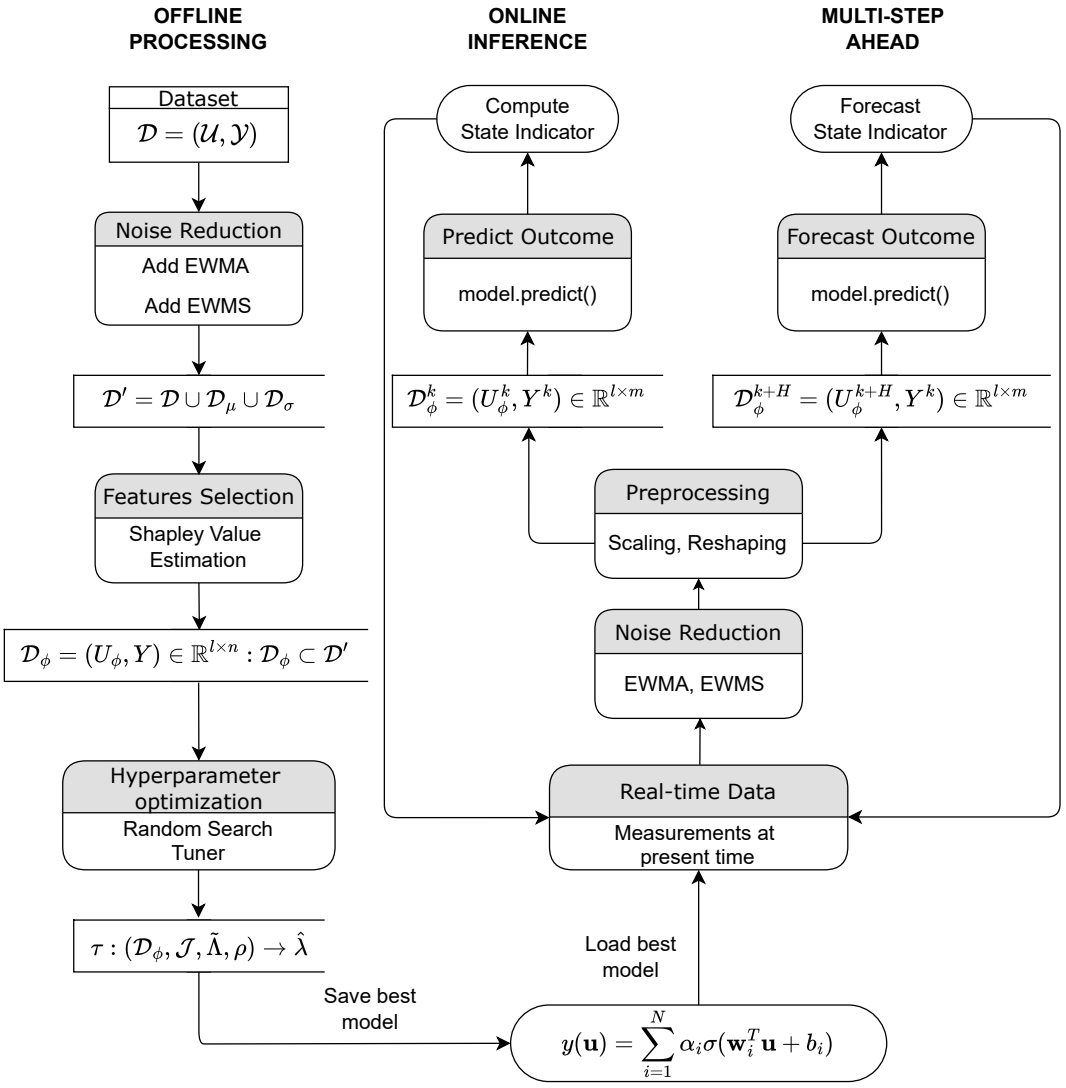

Figure 4.1: Data Flow Diagram for Data-driven PHM Method

to estimate a set of parameters, θ, from a given set $S = \{(\mathbf{u}, \mathbf{y})\}$, so an accurate predictor (or estimator) $\hat{F} : \mathbb{R}^m \to \mathbb{R}^n$ can be found such that $\hat{\mathbf{y}} := \hat{F}_\theta(\mathbf{u})$.
The offline processing, thoroughly explored here, consists of four main stages: measurements' noise reduction, additive features attribution, automated model's hyperparameter optimization and models cross-validation.

Initially, the raw data obtained from the system's measurements is filtered to reduce inherent measurement noise [128]. This thesis proposes smoothing the signals using the exponentially weighted moving average (EWMA) and the exponentially weighted moving standard deviation (EWMS) of the original measurements. A further review

of noise reduction techniques in machine learning, particularly EWMA and EWMS is presented in subsection 4.2.1.
Subsequently, the original dataset is augmented with both EWMA and EWMS values. A detailed feature selection process is then performed using SHAP (SHapley Additive exPlanation), an additive feature attribution method. This method quantifies the contribution of each individual feature to the model's overall performance on the dataset [129], [74]. The result of the SHAP method is a subset of the augmented dataset, containing only the most impactful features (also referred to as explanatory variables) for the model's output.
SHAP has attracted a lot of attention due to its benefits, solid theoretical background and for providing local and global explainability to the model [74]. A further review of features selection, with focus on SHAP and its implementation in the proposed data pipeline is presented in subsection 4.2.2.

Once the relevant features are selected, the next step is to optimise the hyperparameters for the proposed model. The chosen model is a bidirectional LSTM, as described in [124], due to its inherent ability to address the vanishing gradient problem in time-dependent and sequential data series. A random search process is employed to optimise the hyperparameters for the Bidi-LSTM model. The key hyperparameters considered include the number of hidden layers, neurons per layer, learning rate and timesteps, which define the time interval the network will buffer during each processing step.
For practical reasons, following [130], the number of layers in the network is limited to between one and six. This selection is based on factors such as training time and model complexity, particularly considering the constraints of deploying the model on an embedded system.
Secondly, the number of neurons per layer is selected based on a tradeoff: while more neurons enable the model to learn complex nonlinear patterns, they also increase the risk of overfitting and add computational cost. To ensure broad coverage, the number of neurons per layer is set in the interval between 32 and 256.
Learning rate is another important parameter for the network to reduce the local error during the stochastic gradient descent (SGD) model training [131]. Learning rates can be chosen to be constant or to adapt as training loss decreases. In this thesis, the learning rate selection was automated to values between 0.01 and 0.001.
Finally, the dropout rate is also included as a key hyperparameter in the selection process. Dropout is a regularization technique that randomly deactivates neurons

during training with a specified probability. This helps prevent overfitting by making the network less sensitive to specific neuron weights. The recommended dropout rate is typically recommended to be between 20% and 50%. Here, the dropout rate selection was automated to values between 0% and 50%.

Finally, the model is generated and trained using cross-validation techniques until it achieves an acceptable mean square error (MSE) loss value. The model is then validated by demonstrating the smallest estimated risk [132]. Once validated, the model is saved for future use, completing the offline processing stage.
The derived model will then be deployed on the target system for real-time inference. Real-time inference is crucial for enabling a variety of latency-critical intelligent services, such as autonomous vehicles, power plants, and augmented reality [14].
The target system must meet the minimum requirements to load the saved model into memory, allowing the real-time process to be executed.
The systems envisaged in this thesis are primarily the mixed-criticality systems, which shall address varying requirements related to safety, security, determinism, availability and performance [21].
Mixed-criticality systems (MCSs) are classified based on the tasks they perform and the consequences of failing to meet their requirements. A modern vehicle is an example of an MCS, where tasks vary in criticality. High-criticality tasks, such as the antilock braking system (ABS), require meeting strict timing and operational requirements to ensure safety. In contrast, tasks like infotainment and system connectivity are considered low-criticality. Navigation systems typically fall into the medium-criticality category [133].
Additionally, these systems are equipped with heterogeneous embedded systems [14], whose architecture consists of a diversified number of computing devices, such as CPU, GPU, FPGAs and others. Once the optimised model is deployed to the target system, the system can be monitored and its status determined in real-time through the online inference.

Machine learning algorithm performance can vary significantly based multiple factors, as further explained in Section 4.3. There is no one-size-fits-all approach — an algorithm that performs well in one domain may be inefficient or less effective in another, therefore optimizing efficiency becomes increasingly crucial for the success of real-time machine learning solutions [134]. Many machine learning algorithms, particularly deep learning models, require intensive computation for both training

and inference. This computational burden can delay real-time processing, making them less suitable for fast-paced environments [135].
For real-time applications, algorithms with high computational complexity may be impractical for large-scale or real-time applications due to excessive processing time and resource demands. The advantage of combining the hyperparameters optimization with real-time applications requirements is that the model complexity can be reduced as much as the performance criteria allows, in terms of accuracy, precision and also computational complexity. A compromise between performance criteria and model complexity can be reached using the offline processing schema proposed by this thesis.

Finally, a system status forecast is generated. By defining a window of interest, determined by the horizon H, which represents the number of future steps to be forecasted, the model can predict the system's future behaviour with a certain level of accuracy. This is particularly important for safety-critical systems, where abnormal states could lead to catastrophic failures.
Multiple multi-step ahead models have been tested and optimised as part of this thesis. The proposal herein, is to forecast the multivariate independent variables, considered they can be predicted through the knowledge of the selected recent past timesteps.

4.2 Offline Processing Modelling

The purpose of the offline processing stage is to derive the best possible model for the system-of-interest. The approach proposed herein focuses on four main steps during offline processing:

a) measurement noises handling

b) additive features attribution

c) automated model's hyperparameters optimization

d) model cross-validation

The complete flowchart of the offline processing derived from Section 4.2 is shown in Fig. 4.2 below.

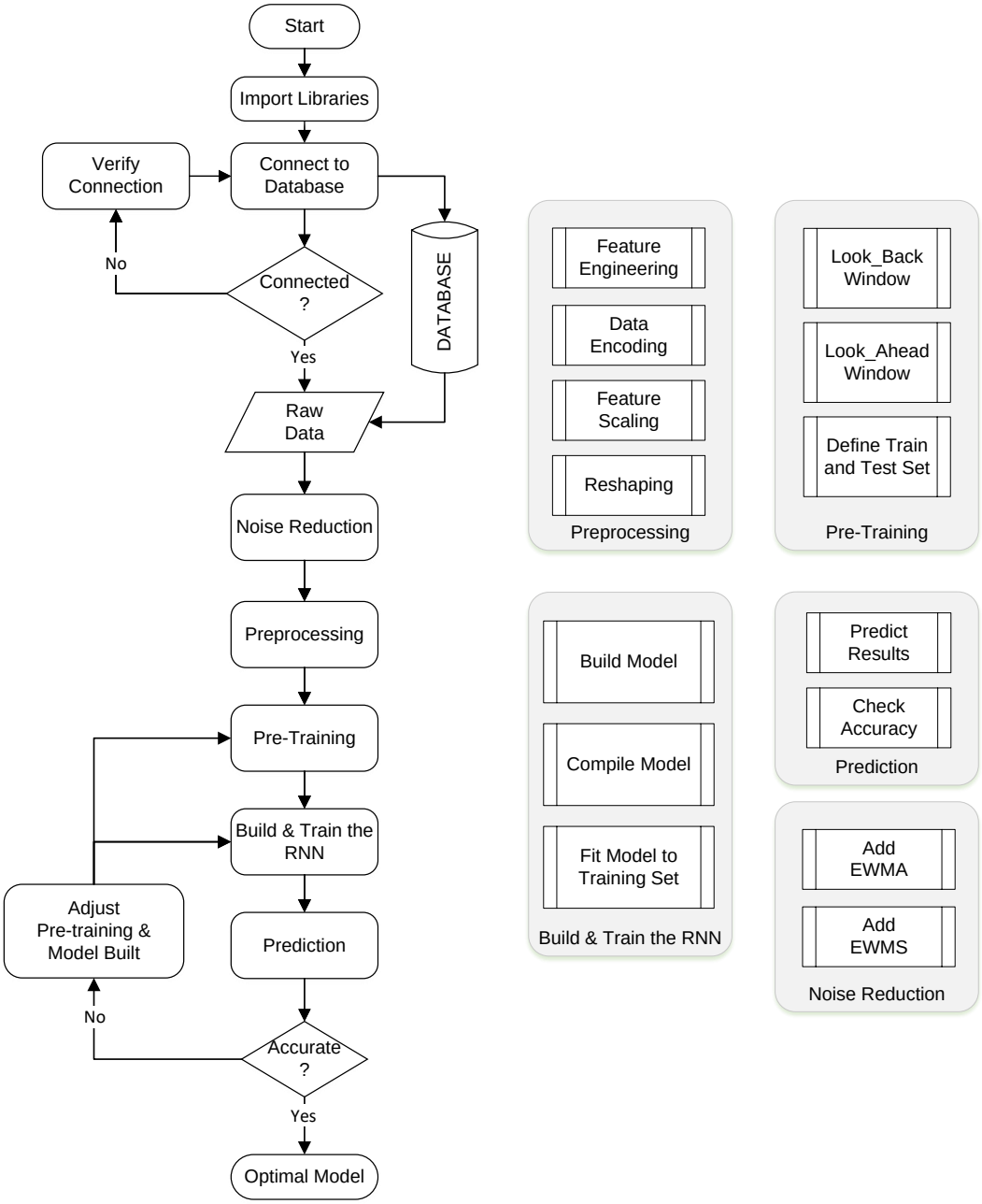

Figure 4.2: Electronic Control Unit Offline Training Flowchart

4.2.1 Handling Noisy Measurements

There are two types of measurement errors: bias error and random error. Bias error is the difference between the actual value of the measured variable and the mean of multiple individual measurements. It is often referred to as accuracy — a measurement with high accuracy has a small bias error. Random error is the deviation of each individual measurement from the average value. It is associated with precision — a measurement with high precision has low random error. Measurement devices have limited precision and are also subject to drift, a gradual shift in measured values over time. This introduces uncertainty into the system, primarily associated with random errors in measurements. Therefore, the uncertainty in the measurements should be evaluated and removed from the system in order to make them more robust. The random error in measurements can be filtered before taken as the input of the system model [136]. According to [126], recursive neural networks models, such as long short-term memory, built on top of noisy raw sensory data may not be robust, therefore a noise reduction strategy would benefit the features mapping.
Two main approaches for noise reduction are found: real-time processing and offline post-processing. For the latter, principal component analysis (PCA), also used for feature selection is the most popular. For real-time algorithms, Fourier Transform, wavelet transform and bandpass filters (which includes EWMA) are used [137].

Recent works have addressed the issue of dealing with uncertainties in the measurements due to unmodelled noises using the exponentially weighted moving average (EWMA) of past data [128], [136], [138].
In the field of signal processing, EWMA serves as a window function application: a low-pass filter, effectively eliminating high frequency noise from a given signal. Moving averages are closely connected to the notion of rolling window estimation. This involves fitting models repeatedly to a fixed-sized window comprising past data. Rolling window models are extensively employed in various domains such as time series analysis, as well as in the fields of economics, finance, and engineering for the purpose of forecasting [139].

The EWMA, represented by μ_k, of a time series represented by $\mathbf{u} = (u_0, u_1 \dots u_k) \in$

$\mathbb{R}^m$ is a vector represented by, as seen in [128, eq. (7)]:

$$\mu_k = \frac{\sum_{i=0}^{k} w_i u_{k-i}}{\sum_{i=0}^{k} w_i} \tag{4.2}$$

where $w_i = (1-\alpha)^i$ with $\alpha = 2/(s+1)$ and s is an arbitrary span, where $\alpha \in \{0, 1\}$. The parameter α controls how closely the model will follow the original time series.

It can be demonstrated that μ_k in (4.2) can be approximated by a linear regression of the historical input data, which is more computationally efficient form, denoted in (4.3). This representation is highly relevant especially for applications where embedded systems run on cost-optimised hardware, as seen in [128, eq. (8)]:

$$\mu_k \approx (1-\alpha)\mu_{k-1} + \alpha u_k \tag{4.3}$$

In the equation, α represents the importance of the previous value, which signifies the trend, and $(1-\alpha)$ determines the importance of the current value.

The concept of the exponentially weighted moving standard deviation (EWMS) of past data is also introduced in [140]. The EWMS represented by σ, of every input signal in u_k at a discrete time k is given by, as seen in [140, eq. (2)]:

$$\sigma = \frac{\sum_{i=0}^{k} w_i (u_k - \mu_k)^2}{\sum_{i=0}^{k} w_i} \tag{4.4}$$

Multiple, differently spanned EWMAs and EWMSs lead to various smoothed versions of all time series and their standard deviation, which are then added to the original dataset to provide noise reduction data to the dataset. Naturally, the number of additional variables to be added is limited by the selected span "s", and shall be carefully selected to limit the system's memory demand.

It is important to emphasize that, although EWMA is widely used to reduce noise in time series data, it can influence the signal distribution. EWMA functions as a low-pass filter, attenuating high-frequency components, which are often associated with noise. Sharp spikes, such as outliers, may also be smoothed; however, sporadic outliers in continuous time series data may not significantly impact the network's learning process. The degree of impact on the original data can be controlled by

the smoothing factor α. A higher α (with values near to 1) retains more of the original dataset structure.
The method proposed in this thesis adds both EWMA and potentially EWMS to the original dataset, not changing the original features. In the next stage then — feature selection using an additive feature attribution method — the algorithm will identify and retain the most relevant features for the studied dataset, ensuring that the noise reduction strategy minimizes any unintended alterations to the original data structure.

4.2.2 Additive Features Attribution

In machine learning, dimensionality refers to the total number of explanatory features in a dataset. When the number of features is too high relative to the number of observations, some algorithms may struggle to produce effective models. Dimensionality reduction techniques aim to select or extract features, creating lower-dimensional spaces while preserving as much useful information as possible. This helps improve model simplicity and performance. Common techniques include Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and t-Distributed Stochastic Neighbor Embedding (t-SNE), with PCA being the most widely used.
Given a set of observations $\mathbf{u}$ with dimension $\mathbf{M}$, $\mathbf{u} = (u_0, u_1, \ldots, u_M) \in \mathcal{U}$, PCA is the standard technique for finding the single best (in the sense of least-square error) subspace of a given dimension, m. This algorithm is based on the search of orthogonal directions explaining as much variance of the data as possible [141].

Feature selection is a crucial process for identifying the relevant features for the system or process under analysis [142]. Feature selection is widely used for model explainability (xAI) [75], [77], [78]. In the proposed framework, the feature selection process is used to retain only the relevant features for training the model, while disregarding those with little or no impact on the final algorithm. Features may also be eliminated if they are redundant [143].
Another important contribution of feature selection is to determine causality [144]. Causal relationships are increasingly seen as the next logical step in building robust solutions. They emphasize that predicting an outcome is not the same as understanding its underlying causes. Causal models can help identify the precise

cause-and-effect relationships by pinpointing the root causes of outcomes [70], while also enabling the modelling of interventions [145].
Recently, feature selection has been used to enhance the explainability of machine learning data-driven models [76], in the context of xAI. This approach involves determining the contribution of each feature to the model's performance, helping to identify an optimal subset of features.
It is crucial to accurately understand the output of a prediction model [146]. One effective way to explain models is through feature attributions. In this approach, each feature is assigned a score (attribution) based on its contribution to the prediction [147]. The definition of additive feature attributions is as follows:

Definition 4.1. *Additive feature attributions*
Suppose $f : \mathcal{U} \to \mathbb{R}$ *is a explanation model mapping an* M*-dimensional feature space* $\mathcal{X}$ *to real-valued predictions. Additive feature attributions for* $f(u)$ *at input* $\mathbf{u} = (u_0, u_1, \ldots, u_M) \in \mathcal{U}$ *comprise of a baseline reference attribution* ϕ_0 *and feature attributions* $\phi = (\phi_0, \phi_1, \ldots, \phi_M)$ *corresponding to the* M *features such that*

$$f(\mathbf{u}') = \phi_0 + \sum_{k=i}^{M} \phi_i u'_i \tag{4.5}$$

where M *is the number of simplified input features,* $u'_i \in \{0, 1\}^M$*, and* $\phi_i \in \mathbb{R}$*.*

Methods that employ explanation models matching Definition 4.1 attribute an effect to each feature. By summing the effects of all feature attributions, these methods approximate the output $f(u)$ of the original model.
An attribute of the class of additive feature attribution methods is the presence of a single unique solution in this class with three desirable properties: local accuracy, missingness and consistency [146].
Several methods align with Definition 4.1, as discussed in Section 2.5. Among them, SHAP has gained significant attention due to its strong theoretical foundation and ability to provide both local and global model explainability [74]. Given its advantages and widespread adoption in the AI community, this thesis employs the classic Shapley value estimation as the chosen explainability method.

SHAP (SHapley Additive exPlanation), a solution based on cooperative game theory, stands out as one of the most prominent explainable AI (xAI) techniques [148]. The Shapley values of features quantify the contribution of individual features to the model's performance on a set of data points. Shapley values are used to measure the contributions of input features to the output of a model at the instance level. Given a specific data point, the goal is to decompose the model's prediction and assign Shapley values to the individual features of that instance [129].
A simple sampling approximation relies on the fact that the Shapley value can be expressed as the expected marginal contribution a player has when players are added to a coalition in a random order [147]. Let $\pi(M)$ be the ordered set of permutations of M, and $\boldsymbol{\mathcal{O}}$ be an ordering randomly sampled from $\pi(M)$. Let $\mathrm{pre_i}(\boldsymbol{\mathcal{O}})$ be the set of players that precede player i in $\boldsymbol{\mathcal{O}}$. The Shapley value of player i is the expected marginal contribution of the player under all possible orderings of players, as seen in [147, eq. (2)]:

$$\phi_i(\upsilon) = \mathbb{E}_{\boldsymbol{\mathcal{O}} \backsim \pi(M)}[\upsilon(pre_i(\boldsymbol{\mathcal{O}}) \cup (i)) - \upsilon(pre_i(\boldsymbol{\mathcal{O}})] \tag{4.6}$$

By sampling a number of permutations and averaging the marginal contributions of each player, one can estimate this expected value for each player and approximate each player's Shapley value.

In this thesis, a Shapley tree-based model is used to obtain and interpret the features with highest impact on the target calculation, as in [149]. According to [150], as oppose to several inconsistent common feature attribution methods for tree ensembles, SHAP values consistently attribute feature importance, better align with human intuition, and better recover influential features.
A Tree SHAP algorithm, a high-speed algorithm for estimating SHAP values of tree ensembles, then extend this to SHAP interaction values [150] in order to narrow down the relevant features for the case study.

4.2.3 Automated Hyperparameters Optimization

The objective of supervised machine learning is to train a model using a dataset of n observations, drawn from an unknown distribution P_{xy}, so that a labeled dataset $\mathcal{D} \sim (P_{xy})$ in order to ensure that it can effectively generalize to new observations

that arise from the same underlying data generation process.

Typically, a machine learning algorithm transforms a problem that needs to be solved into an optimization problem, using various methods to find a solution. The goal, before the training phase, is to identify a set of hyperparameter values that achieve the best performance on the data within a reasonable amount of time. This process is known as hyperparameter optimization (HPO) or tuning. It plays a critical role in the prediction accuracy of machine learning algorithms [151]. HPO involves optimizing a loss function over a graph-structured configuration space [152].

Automated HPO has several important use cases, as highlighted by [130]: reducing the human effort required to apply machine learning, improving the performance of algorithms, and enhancing the reproducibility and fairness of scientific studies. Automated HPO is clearly more reproducible than manual search. It enables fair comparisons, as different methods can only be compared fairly if they receive the same level of tuning for the given problem [131]. Hyperparameters that have a stronger effect on weights during training are more influential for neural network training. Currently, the most commonly adopted optimizer for training deep neural networks is stochastic gradient descent, as shown in (3.52), along with its variants. In addition to the choice of optimizer, the corresponding hyperparameters are critical for certain networks [153]. Let $\mathcal{D} = (\mathcal{U}, \mathcal{Y})$ be a given dataset, with a vector of hyperparameters denoted by the objective function, as seen in [17, eq. (1)]:

$$\lambda^* = \arg\min_{\lambda \in \mathbf{\Lambda}} \mathbb{E}_{(D_{train}, D_{test}) \sim \mathcal{D}} \boldsymbol{V}(\mathcal{L}, \hat{\mathcal{F}}_\lambda, D_{train}, D_{test}) \tag{4.7}$$

where $\mathbf{\Lambda}$ represents the overall hyperparameter configuration space.
$\boldsymbol{V}(\mathcal{L}, \hat{\mathcal{F}}_\lambda, D_{train}, D_{test})$ measures the loss of a model generated by algorithm $\hat{\mathcal{F}}$ with a vector of hyperparameters $\lambda \in \mathbf{\Lambda}$ on training data D_{train} and evaluated on validation data D_{test}, where $(D_{train}, D_{test}) \subset \mathcal{D}$.

A review of the HPO algorithms and hyperparameters search space and their applications is presented in [153]. It introduces and explains three main search algorithms, Grid Search, Random Search and Bayesian Optimization. Another optimization method with increasing interest in the literature is called Neural Architecture Search (NAS), which differs from the HPO algorithms by focusing on finding an optimal architecture for neural network-based systems [154].

Grid search is a basic method for HPO. It performs an exhaustive search on the hyperparameter set specified by users. Users must have some preliminary knowledge on these hyperparameters because it is they who generate all candidates. Grid search is suitable for optimizing several hyperparameters with a limited search space. Random search is a basic improvement over grid search, as it randomly explores hyperparameters from specific distributions of possible parameter values. The search continues until the predetermined budget is exhausted or the desired accuracy is achieved. Bayesian optimization (BO) is a sequential, model-based method designed to find the global optimum with the fewest trials. It balances exploration and exploitation to avoid getting stuck in local optima. Grid search and manual search are the most widely used strategies for hyperparameter optimization. However, it has been demonstrated that random search is able to find models that are as good or better within a fraction of the computation time [155].

In this thesis, random search was selected as the hyperparameter optimization method for all evaluated models. Although more advanced techniques like Bayesian Optimization and Neural Architecture Search (NAS) could have been employed, random search yielded highly satisfactory results for the tested datasets. Therefore, it was deemed an effective and appropriate choice for this research.
Table 4.1 shows a list of typical hyperparameters subjected to optimization for the algorithms used in the subsequent chapters. The list in agreement with [130] and [152].

Table 4.1: Typical Hyperparameters for Optimization in Chapter 5

Algorithms	Hyperparameters	Possible Values
Decision Tree	criterion	{'*gini*', '*entropy*', '*log_loss*'}
	max_leaf_nodes	int
	min_samples_split	int or float
	max_depth	int
	splitter	{'*best*' or '*random*'}
Support Vector Regression	kernel	{'*linear*', '*poly*', '*rbf*', '*sigmoid*', '*precomputed*'}
	"C"	float
	gamma	{'*scale*', '*auto*'} or $float$
	degree	int
	coef0	float
Logistic Regression	penalty	{'*l*1', '*l*2', '*elasticnet*'} or none
	"C"	float
	class_weight	{'*dict*', '*balanced*'} or none
	solver	{'$lbfgs$', '*liblinear*', '*newton-cg*', '*newton-cholesky*', '*sag*', '*saga*'}
	multi_class	{'*auto*', '*ovr*', '*multinomial*'}
Artificial Neural Network	nbr_layers	int
	nbr_neurons	int
	nbr_epochs	int
	batch_size	int
	learning_rate	float
	dropout_rate	float
Long Short-Term Memory	nbr_layers	int
	nbr_neurons	int
	nbr_epochs	int
	batch_size	int
	learning_rate	float
	dropout_rate	float

4.2.4 Cross-Validation for Model Selection

Cross-validation (CV) is used to estimate the risk of an estimator and to facilitate model selection. It is also employed to prevent overfitting, which occurs when a

model fits too closely to a particular training dataset, thereby reducing its accuracy on new, unseen data. The main idea behind CV is to split the data, either once or multiple times, to estimate the risk of each algorithm. A portion of the data, known as the training sample D_{train}, is used to train each algorithm while the remaining portion, the validation sample D_{test}, is used to estimate the algorithm's risk. CV then selects the algorithm with the smallest estimated risk [132].
Two groups of CV techniques are organised in [156], subject-wise and record-wise. Subject-wise division ensures that the subjects in the training and holdout (validation) sets are independent. In other words, the records from each subject are assigned to either the training or the holdout set. Contrarily, record-wise division splits the dataset randomly, without taking into account that the training set and the holdout set could share records from the same subjects.
In this thesis, a *k-fold* CV technique (record-wise) is used, where the dataset is divided into k blocks (folds). One of the k blocks is designated as the validation set, while the remaining $\{k-1\}$ blocks serve as the training set. In *k-fold* CV, there is no overlap between test sets due to the use of a random sampling technique. The learning algorithm is applied to each training set, and the resulting model is evaluated on the corresponding test set. This process is repeated k times, where $k = n$, and the performance is estimated as the average over all k blocks (test sets) [156].

To ensure the reliability of the research, all experiments conducted in this study used a *10-fold* cross-validation on the training set. This allows for a fair evaluation of models obtained using different machine learning algorithms.
To reduce the variance of cross-validation (CV) results, CV is performed with random splits. Generally, increasing k reduces bias but comes at a higher computational cost and may increase variance. Given that the datasets used in the subsequent chapters are sufficiently large, a *10-fold* CV provides a statistically sound distribution and represents a good compromise based on the experimental results in this work.
For the scoring metrics in this thesis, it is selected the R^2 (R-squared), given by (4.8), since the objective is to minimize the variance of the regression model. In the equation, SS_{res} is the residual sum of squared error of the regression model, whereas SS_{tot} is the total sum of squared errors, as seen in [17, eq. (3)]:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{k=1}^{n}(y_k - \hat{y}_k)^2}{\sum_{i=k}^{n}(y_k - \bar{y}_k)^2} \tag{4.8}$$

As a result, each run of the cross-validation produces an array of k-scores for the estimator. These scores are analysed from a statistical perspective to assess the model's adherence to the available data, considering metrics such as mean, standard deviation, variance, and bias. This provides a solid guideline for model selection, ensuring its suitability for the specified system.

4.3 Real-Time Inference

A key aspect of mixed-criticality systems (MCS) is that system parameters, such as tasks' worst-case execution times (WCETs), depend on the criticality level of the tasks. The demand for real-time deep learning inference is steadily increasing to support latency-critical intelligent services, such as autonomous driving, remote unit control, power plants, and more. Meeting the deadlines for real-time inference workloads is crucial for these services to prevent hazardous or catastrophic situations and to ensure optimal user experiences [14].
Online inference addresses real-time requirements in the present moment. It utilizes the optimized model generated during the previous stage and processes real-time data obtained from measurements at the current time k, as described in (4.1).

The objective of real-time inference is to identify suitable models for real-time prediction, Prognostics and Health Management (PHM), and Remaining Useful Life (RUL) estimation. This enables the algorithm to provide failure predictions, supporting preventive maintenance and system reliability. The system is expected to be trained on a powerful platform, like a GPU, for instance, and to be deployed to prediction either on cloud based applications or at the edge, like embedded systems, FPGAs and others. Therefore, for the real-time prediction, those models shall be able to run within the specific required system dynamics. For example, as exemplified by [29], some systems require a polling frequency of one (1 Hz) (one reading per second). Other systems, like motor-pumps assemblies can reach up to 6,000 rpm at 50/60 Hz. For most of the mechanical variables, 1 kHz data sampling would suffice, whereas to the electrical monitoring a higher data sampling is required, around 4 to 5 kHz. Yet, for motor-compressors systems, similar monitoring is applicable with the addition of magnetic bearings which require a higher sampling rate, from 10 to 20 kHz.
Hence, any PHM system based on machine learning techniques shall cater for the system dynamics to which it has been designed. Naturally, the system performance is strictly linked to the hardware/topology/architecture/system in use. Many studies have been conducted in this direction. For example, an evaluation of the feasibility of training some deep learning models for a mobile device is provided in [157]. The paper concludes that choosing an appropriate batch size has a big impact on whether training a deep learning model is feasible. The paper also asserts that

balancing workloads on CPU and GPU to maximize system throughput is a good strategy. A survey of hardware platforms for deep neural networks is provided in [158], covering CPU, GPU, FPGA and ASIC (Application-Specific Integrated Circuit); and different hardware devices (multi-core, SIMD, GPU, and FPGA) are presented and compared in [159] against a specific algorithm and concludes that best performance is obtained by using GPU, under large dataset and FPGA for small dataset because of its lower power, especially for some embedded applications.

Machine learning algorithm performance can vary significantly based on factors such as data characteristics, computational resources, and task-specific requirements. As mentioned before, there is no one-size-fits-all approach — an algorithm that performs well in one domain may be inefficient or less effective in another, therefore optimizing efficiency including computational time, memory usage, and scalability becomes increasingly crucial for the success of real-time machine learning solutions [134]. Many machine learning algorithms, particularly deep learning models, require intensive computation for both training and inference. This computational burden can delay real-time processing, making them less suitable for fast-paced environments [135].
For real-time applications, computational complexity refers to the amount of resources - time, memory, processing power - required by an algorithm to train and deploy. Algorithms with high computational complexity may be impractical for large-scale or real-time ML applications due to excessive processing time and resource demands. In [134], computational complexity is assessed by measuring the training and testing times required for each algorithm, and concludes observing that while deep learning, including neural networks, often achieve top performance, selecting the most efficient algorithm should account for the specific needs of the application, including available computational resources, memory usage, and energy consumption.
The advantage of combining the hyperparameters optimization with real-time applications requirements is that the model complexity can be reduced as much as the performance criteria allows, in terms of accuracy, precision and also computational complexity. A compromise between performance criteria and model complexity can be reached using the offline processing schema proposed by this thesis.

4.3.1 Real-Time Data Processing

The complete flowchart of the real-time data processing is depicted in Fig. 4.3 below.

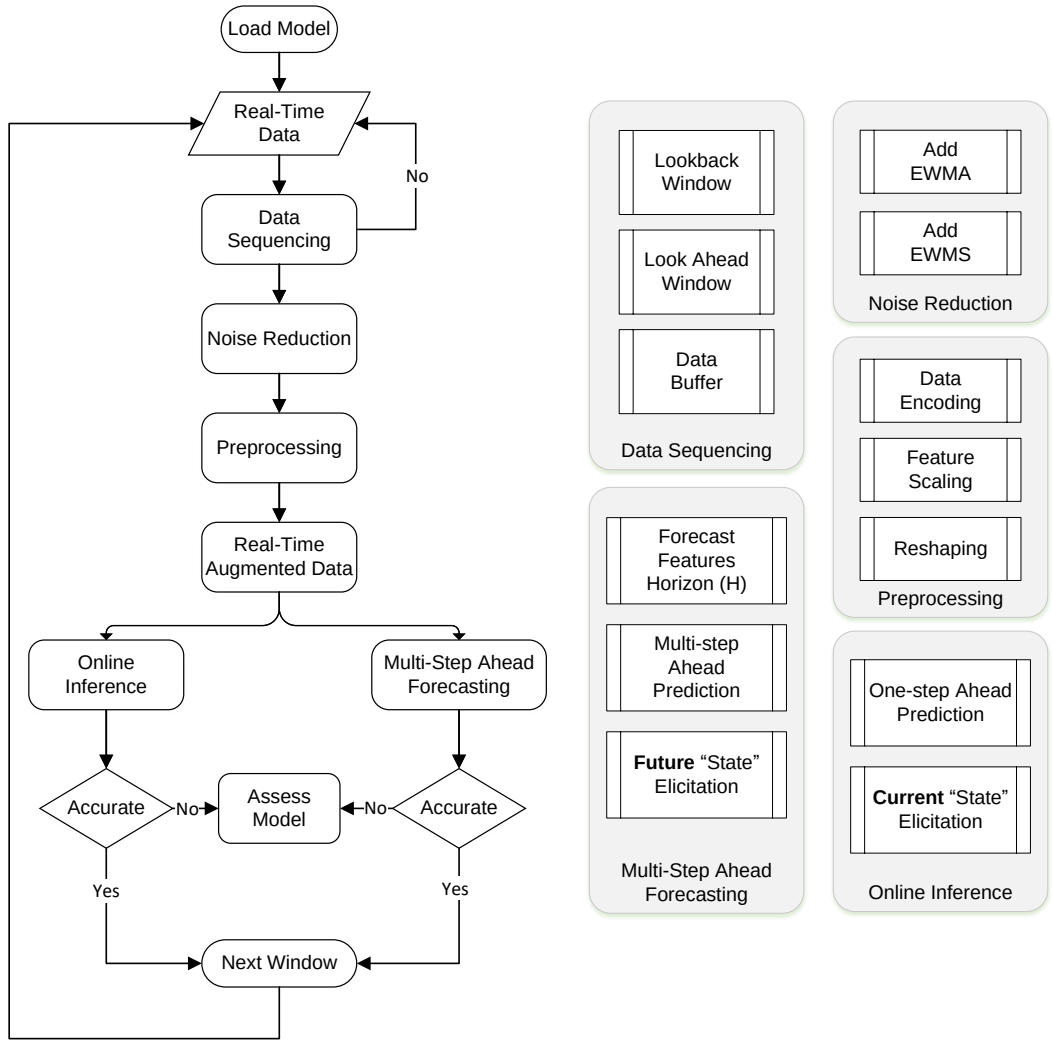

Figure 4.3: Novel PHM Real-Time Processing Flowchart

At the system startup, the optimised network model is loaded onto the system's memory as the most accurate predictor (or estimator).

At every k-interval, a new set of real-time data $S = \{(\mathbf{u}, \mathbf{y})\}$ is read and loaded onto the embedded system's memory.

LSTM can use lagged observations of a time series as timesteps to improve its

prediction performance, as stated in Section 3.4. Herein, a careful selection of the past data timesteps *n-timesteps* is made as a trade off between the predictor model accuracy and real-time constraints. The model to be loaded is the one trained and optimised for the selected *n-timesteps*, otherwise the prediction will fail. In case the system estimates multi-step ahead, the value of the window with future values, horion H, to be forecasted is also required. This is part of data sequencing in Fig. 4.3.

4.3.2 Real-Time Noise Reduction

The raw data obtained from the system's measurements is augmented with the relevant Exponentially Weighted Moving Average (EWMA) and, potentially, the Exponentially Weighted Moving Standard Deviation (EWMS) of the original measurements. In real-time applications, this augmentation is applied only to the features identified during the feature selection process, ensuring an optimized set of features that retains the most relevant information for the process. This approach is proposed to mitigate inherent measurement noise [128], improving the robustness of the data while maintaining the integrity of the original signal.
The EWMA sequence will be recursively calculated according to (4.3). In addition to the computational efficiency, the recursive implementation allows the algorithm to compute μ_k without the need to store multiple past values of it, then only keeping track of the previous state μ_{k-1}.
The EWMS will be derived from (4.4), which takes the current μ_k, u_k and the span s as inputs. If demonstrated during the offline processing that the added variables related to the EWMS are not relevant for the real-time processing, these ones shall not be included to avoid a higher computational cost.

4.3.3 Real-Time Prediction

After augmentation, the data undergoes standard scaling to normalize the feature distributions, ensuring consistency across different magnitudes. Following this, the data is reshaped into the appropriate format required by the model. Once these preprocessing steps are completed, the dataset is ready to be processed by the

online inference estimator for real-time predictions.
At every k-interval then, the optimised model predicts the $k+1$ system status, $\hat{F} : \mathbb{R}^m \to \mathbb{R}^n$ such that $\hat{\mathbf{y}}_{k+1} := \hat{F}_\theta(\mathbf{u}_k)$.
This prediction serves as a key indicator for condition monitoring, enabling immediate actions when necessary. Based on these indicators, system controls—such as alarms, automated warnings, or even shutdown protocols—are triggered to prevent failures and ensure operational safety.

The system is also designed to provide an advisory window through multi-step ahead forecasting, allowing operators to anticipate potential failures and take preventive actions. This aspect is further explored in the next section.

4.4 Multi-step Ahead Forecasting

A multi-step ahead (also called long-term) time series forecasting task consists of predicting the next H values of $(\mathbf{x}, \mathbf{y})$ as in (4.9), as seen in [160, eq. (1)], of a historical time series $[(x, y)_1 \ \ldots \ (x, y)_N]$ composed of N observations, where $H > 1$ denotes the forecasting horizon [80],

$$(\mathbf{x}, \mathbf{y}) = [(x_{k+1}, y_{k+1}) \ \ldots \ (x_{k+H}, y_{k+H})] \in \mathbb{R}^{m \times n} \tag{4.9}$$

According to [80] and [161] there are several strategies for multi-step ahead forecasting, which the most used are Recursive (also known as Iterated), Direct and MIMO. Recursive forecasting is the primary form of multi-step forecasting [80], where forecasting is done by the factorization of previous values, amplifying potential errors and leading to lower quality predictions as the time horizon increases. The MIMO forecasting aims to estimate the future values in one step. Multi-output approaches sidestep the issue of error feedback by jointly estimating over the prediction window [88].

4.4.1 Real-Time MSA Strategy Selection

Initially, from the MSA strategies described in the Section 2.6 of the literature review, the Recursive and the MIMO ones were selected and compared to forecast the multivariate independent features $X's$.
Combined with the MSA strategies above, in this thesis, two different statistical regression methods are used to forecast, in real-time, the individual $X's$ for the determined horizons: autoregressive (AR) model and ARIMA (autoregressive integrated moving average). The first one, AR, only takes into consideration the lagged values, whereas ARIMA also considers the stationarity and moving average. AR can be seen as a subset of ARIMA, however both were considered to verify their performance on dataset fitness and real-time scenarios.
The most well-known method is univariate "Autoregressive Moving Average (ARMA)" for a single time series data in which Autoregressive (AR) and Moving Average (MA) models are combined. Univariate "Autoregressive Integrated Moving Average (ARIMA)" is a special type of ARMA where differencing is taken into account in

the model [162]. ARIMA is also a well-used approach for time series forecasting [23] and is the model considered herein.

The next step is to use the forecasted $X's$ as inputs for the selected pre-trained and optimised machine learning models. These models will then provide the multi-step-ahead forecast for the future $Y's$, based on the defined horizon (H) of interest.

This thesis proposes a novel hybrid approach for real-time multi-step-ahead (MSA) forecasting. In this method, autoregressive models operate in real-time at every k-interval, utilizing available data within the permitted processing window. Subsequently, an offline pre-trained model is employed to infer target values for the specified forecasting horizon H.

As highlighted in Section 2.6 of the literature review, LSTM models are widely used for MSA forecasting applications. In this thesis, the three most frequently cited variants of LSTM models for MSA forecasting were implemented and tested:

- **Stacked LSTM** (stacked-LSTM) as the one in [90].
- **Bidirectional LSTM** (Bidi-LSTM) as in [68] and [123].
- **CNN-LSTM** as in [79].

One method of particular interest is the Multivariate Multi-step Bidirectional LSTM (Bidi-LSTM), which is notable for its ability to adapt to multiple time-series sequential datasets, as mentioned in the previous chapter.

4.4.2 Multivariate Multi-step Bidirectional LSTM

A multivariate multi-step bidirectional LSTM architecture has been implemented, containing forward and backward LSTM layers. This architecture was trained to provide at each time the forecast of next H values of $(\mathbf{y}) = (y_{k+1}, \ldots, y_{k+H})$, using N past timesteps of the input $(\mathbf{x}) = (x_{k-N}, \ldots, x_k)$.
Fig. 4.4 illustrates an example of MSA Bidi-LSTM architecture, which includes both a forward LSTM layer and a backward LSTM layer. Unlike the architecture depicted in Fig. 3.6, this model is capable of forecasting the next H values of the series in a single pass.
Two models for the Bidi-LSTM are generated: one that considers a standard LSTM output for one-step ahead forecasting, and another, as shown in Fig. 4.4,

for multi-steps ahead forecasting. The models are different from each other by one reason: besides the look-back window, the MSA also requires the definition of the forecast window. The dense output layer for one-step ahead is "1" (one) and for the multi-step ahead is the horizon "H".

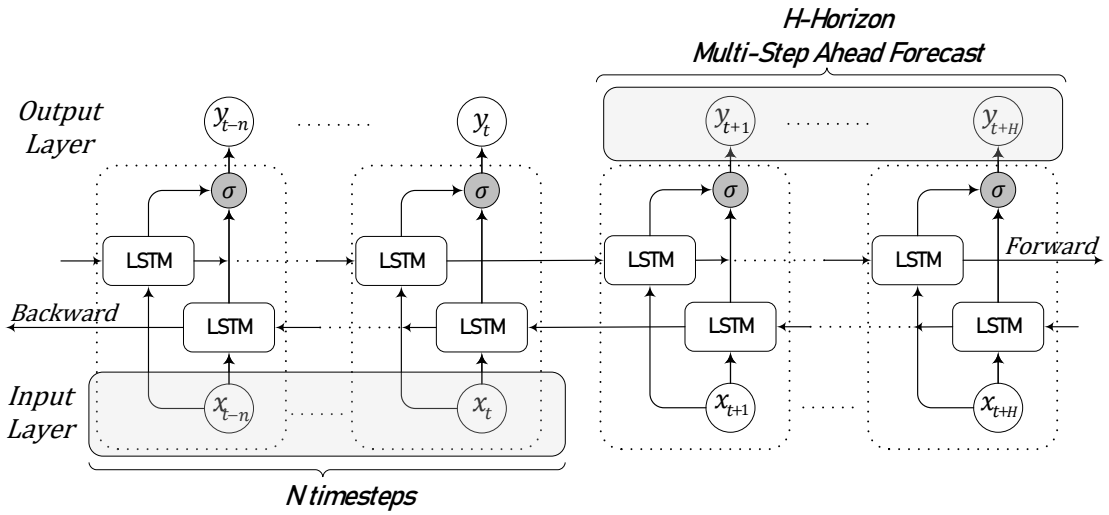

Figure 4.4: Multi-Step Ahead Bidirectional LSTM Architecture

The adapted multivariate bidirectional LSTM model, designed for hyperparameter optimization, is illustrated in Fig. 4.5. This model is highly configurable, with key hyperparameters, including the number of layers, LSTM cells, dropout rate, learning rate, batch size, and number of epochs, being optimized for performance. Notably, the entire architecture, including various LSTM layer configurations, can be adapted and optimized for the specific dynamic system of interest. A full implementation of the proposed architecture is provided in Appendix A for MSA computation in the first experiment. The results demonstrate that the system is well-optimized for the dataset under study, highlighting the accuracy of the proposed model.

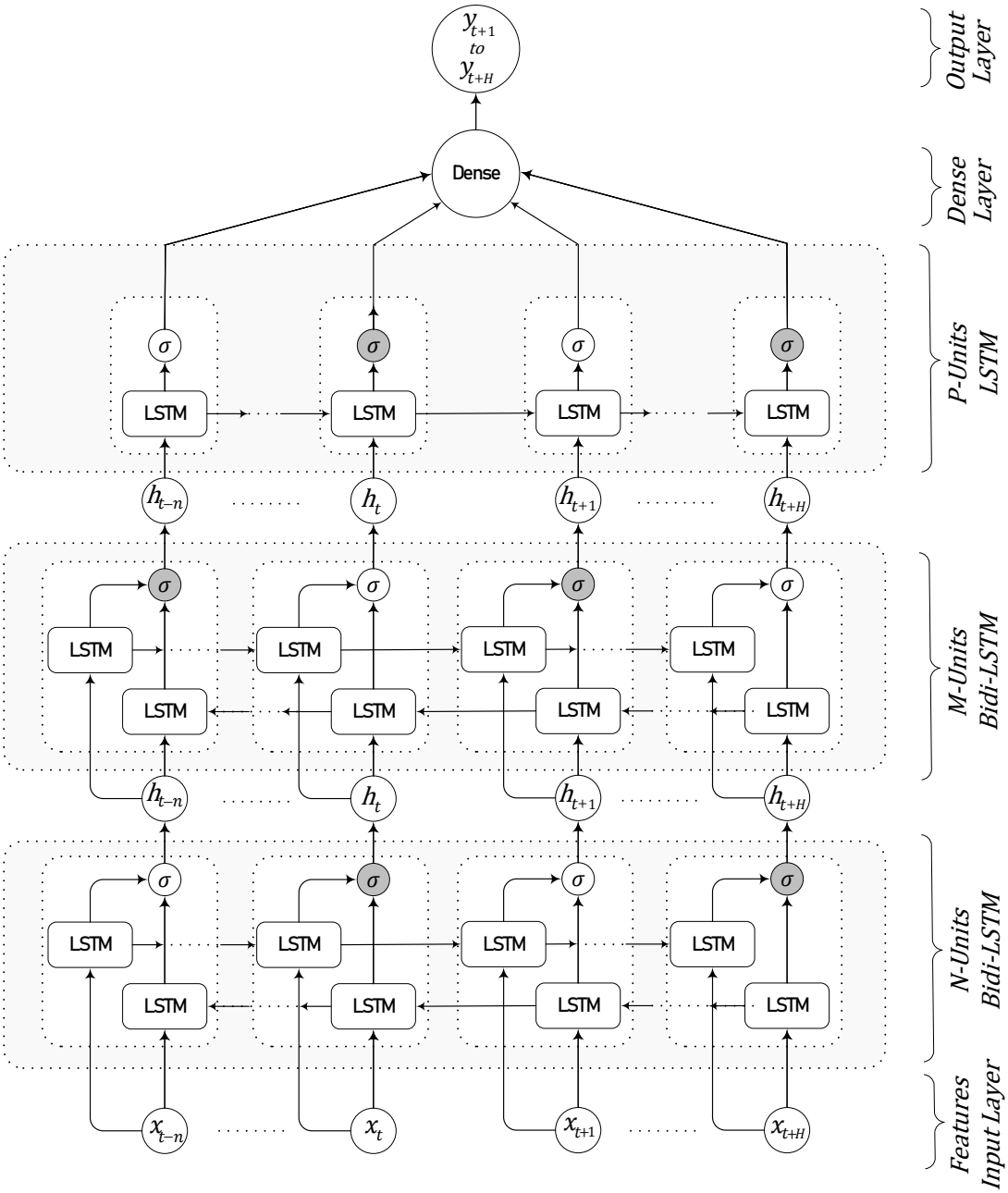

Figure 4.5: MSA Bidi-LSTM Proposal with Hyperparameters Optimization

4.4.3 Autoregressive Integrated Moving Average (ARIMA)

In this thesis, ARIMA is selected for comparison with both the stacked-LSTM and Bidi-LSTM models. ARIMA is a widely used approach for time series forecasting [23]. It is particularly effective for addressing non-stationarity in datasets.

Non-stationarity refers to time series data where the mean, variance, and covariances change over time. Non-stationary behaviors can include trends, cycles, random walks, or combinations of these. Such data are generally unpredictable and cannot be effectively modeled or forecasted. Using non-stationary time series data may lead to spurious relationships between variables that do not actually exist. To obtain reliable results, it is necessary to transform non-stationary data into stationary data [139].

ARIMA holds a relationship between the current observation and past observations (Autoregression - AR) with a capability of differencing of actual observations in order to make the time series stationary (Integrated - I) with lags of the forecast errors of the moving average mode (Moving Average - MA).

These components are included in an ARIMA model as a set of parameters. The standard notation for the ARIMA model is usually given as $ARIMA(p, d, q)$; where p is the number of lag observations, d is the degree of differencing and q is the size (or span) of the moving average window.

Initially the system shall be assessed for non-stationarity, so the measured values $x(k)$ are replaced by the results of a recursive differencing process $\nabla^d x$, where d is the number of times the differencing process has been applied. The first order differencing is shown in (4.10) [160, eq. (2)]:

$$\nabla^d x^*(k) = \nabla^{d-1} x(k) - \nabla^{d-1} x(k-1) \tag{4.10}$$

Finally, the ARIMA model to estimate $\hat{x}$ is given below [160, eq. (3)]:

$$x^*(k) = \mu + \sum_{i=1}^{p} \phi_i x^*(k-i) + \sum_{i=1}^{q} \theta_i \epsilon(k-i) + \epsilon(k) \tag{4.11}$$

Where $x^*(k)$ is the current estimated value at time sequence k, μ is the mean of the series, $\epsilon(k)$ is the random error at time k; ϕ and θ are parameters for the AR and MA addends; and p and q are the autoregressive and moving average specific parameters respectively.

4.5 Chapter Summary

This chapter presents the main contribution of this thesis: a novel data-driven Prognostics and Health Management (PHM) methodology for system-level embedded electronics. The proposed approach consists of three main stages: a robust offline processing modelling, real-time inference and multi-step ahead forecasting.

As an initial processing step, a noise reduction strategy is proposed by introducing two computed values for each field measurement: its Exponentially Weighted Moving Average (EWMA) and Exponentially Weighted Moving Standard Deviation (EWMS). To preserve the original data structure, the dataset is augmented with these values rather than modifying existing features. A classic Shapley additive feature attribution model is then applied to identify the most relevant features for the system of interest, which may or may not include the newly introduced features. This strategy aligns with [**RQ2**], which focuses on maintaining predictor explainability and reliability.
Following this, a rigorous hyperparameter optimization (HPO) process is conducted using random search within a well-defined parameter space. Although alternative HPO techniques were available, random search yielded highly satisfactory results for both systems tested in this thesis.

The proposed PHM framework utilizes a specialized bidirectional LSTM architecture for the experiments, designed to handle structured time series data with high predictive accuracy requirements. This chapter establishes a comprehensive data pipeline for real-time processing, serving a dual purpose: real-time inference and multi-step forecasting.

For multi-step-ahead forecasting, we propose a hybrid approach that combines the reliability of the Autoregressive Integrated Moving Average (ARIMA) model for forecasting explanatory features within a defined horizon and the specialized bidirectional LSTM (Bidi-LSTM) for accurately predicting the system's operational state. During the assessment of machine learning algorithms, particularly in Chapter 5, various multivariate LSTM implementations were explored, with Bidi-LSTM emerging as the predominant choice. This approach forms the foundation of the novel PHM framework for real-time inference and multi-step-ahead forecasting in embedded systems, addressing [RQ3] by providing a robust machine learning

framework for PHM in real-time embedded applications.

The next two chapters present a complete implementation of the proposed methodology on two different real-world systems, with results compared to state-of-the-art methods. Specifically, the battery dataset is used, and the outcomes are comprehensively compared with two model-based estimators: the Extended Kalman Filter and the Unscented Kalman Filter.

Chapter 5

Experiment 1: Electronic Control Unit

This chapter implements the novel PHM framework proposed in Chapter 4 using a real-world industrial dataset to predict both the current and future operational states in real time. First, the case study, its environment, and its application are described, followed by feature selection to analyze the 47 available features and the target operational state, reducing the number of variables for improved interpretability. To evaluate machine learning performance on the dataset, seven supervised learning methods are proposed, implemented, and optimized according to the PHM framework. The optimized models undergo cross-validation and are then compared using well-established performance criteria, including execution time to assess feasibility on the target hardware. The chapter concludes with a multi-step-ahead forecasting assessment, where various strategies and models are tested. The most promising approaches are implemented on a standard test bed to evaluate real-time performance.

Many works have proposed an integrated edge-to-cloud architecture to deal with the multiple challenges of real-time systems, such as managing low latency, minimal bandwidth, and fault-tolerant applications, as well as for an architecture able to train large-scale real-world industrial applications data offline and deploy it effectively on the target system. These systems address the requirements for

edge computing, internet of things (IoT), increase of data volume, variety and velocity [163], [164]. The seamless integration between the end (or edge) devices and the cloud for intelligent management is also contemplated [165]. Big data analytics and AI service will be fully integrated to mine insights or knowledge about trends (predictive maintenance), bottlenecks, and how to set the parameters, processing algorithms/rules or coordination mechanisms of gateways and the cloud for data processing [166].

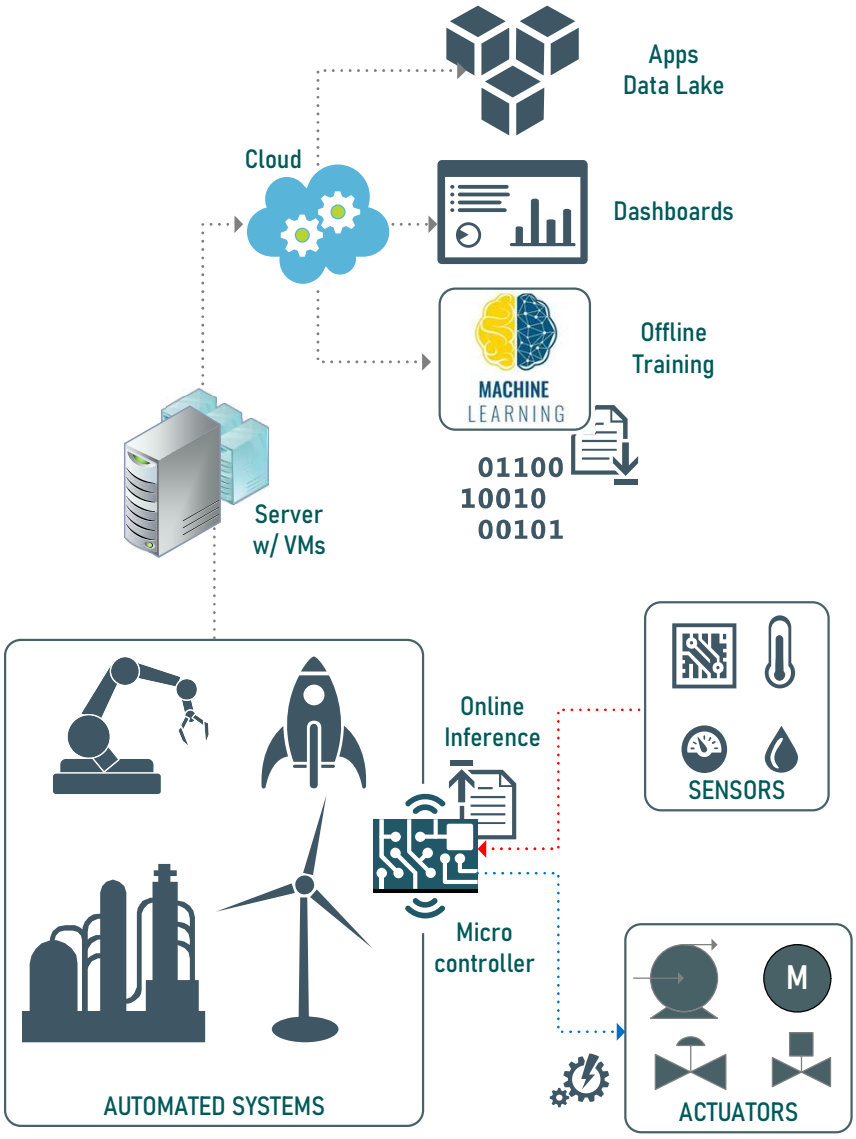

Figure 5.1: Edge to Cloud Automated System

The case study in this experiment is based on a real-world system that is part of an industrial asset. It involves a control module with a microcontroller-embedded system that receives data from field sensors and operates valves, motors, and pumps according to a predefined cause-and-effect automated system.

Fig. 5.1 shows a general edge-to-cloud automated system, exemplifying different industrial systems, like a wind turbine, a spacecraft, a robotic arm and a factory utility. The system described here processes a large volume of data and stores it on a local (edge) server, which hosts multiple virtual machines (VMs) dedicated to different subsystems and tasks.

One key task is to transmit data to a cloud-based server, where various applications run simultaneously, including apps, dashboards, and machine learning training for the edge system. On the cloud, new data is continuously added every minute, enabling the development of an optimized machine learning model through intensive offline training, as detailed in the previous chapter.

The model is then loaded onto the target embedded system for real-time inference and provides an advisory window of 30-minute, ie, 30 steps ahead forecasting.

5.1 ECU - PHM Assessment

The valuable real-world dataset used in this study was obtained from an electronic control unit (ECU), similar to the one described by [29]. The ECU manages various process devices and actuators while receiving data from field sensors. Additionally, the ECU itself provides a comprehensive set of housekeeping data. In this application, real-time data is collected by the ECU from multiple sensors and actuators and then transmitted to a cloud server. In the cloud, instant data is processed for condition monitoring, while historical data is analysed for trending and performance evaluation purposes.
For the case study, the ECU began exhibiting multiple faults, causing it to intermittently reset. During these unintended resets, which configured fault events, the ECU transitions through a predefined set of states, denoted as $\boldsymbol{s}$, referred to as operational modes.

For the system in case, the possible operational modes are:

Table 5.1: List of all Operational Modes and Description

Operational Modes	Mode Description
0	Unknown
1	Reset
2	Started
3	Initializing
4	Programming
5	Operational

States “0”, “1” and “2” are transitory states, however they are expected to be presented in long datasets. State “0” may be related to loss of communication and consequent loss of record rather than a real problem with the controller.
State “5” is the normal operational mode, expected to keep the dependable system attributes availability, reliability and integrity, according to Table 2.3 at an acceptable level.
The real-world system includes a built-in reset function: whenever a fault occurs, the master controller instructs the ECU to restart and execute the necessary steps to return to an operational state. However, such resets are not expected to happen frequently, as the system is located in a remote area, and redundancy is implemented

to ensure fault tolerance, high reliability, and availability.
Therefore, the ECU is specifically designed to maximize its operational state, $\boldsymbol{s = 5}$, while minimizing reset events, $\boldsymbol{s = 1}$. Consequently, the PHM system uses $\boldsymbol{s}$ as the target variable, with a higher prediction accuracy requirement for the critical state $\boldsymbol{s = 1}$.

Fig. 5.2 depicts the state diagram of the system operational modes. When the ECU is initially powered on, it briefly remains in the unknown state ($s = 0$) during the boot-up process. Once the bootloader process is complete and all ECU services become available, the system transitions to the started state ($s = 2$). At the started state, services are available, but the ECU remains in a pre-operational mode. If the operator decides to actively use the ECU, a command is issued, and the system temporarily enters the initialization state, transitioning as ($s = 2 \rightarrow s = 3$). Once the ECU becomes fully operational, it reports a valid transition, either ($s = 2 \rightarrow s = 5$) or ($s = 3 \rightarrow s = 4$), as illustrated in the machine state diagram.
The states $s = 0$, $s = 2$ and $s = 3$ are considered transitory and are not expected to be frequently observed in the dataset.
If maintenance, a firmware update, or a software upgrade is required, the system can transition from started to programming ($s = 2 \rightarrow s = 4$).
At a full operational status, the ECU assumes operational state, ($s = 5$) and is expected to remain in this state unless scheduled maintenance or other anticipated events occur. As a dependable system, the ECU is designed to maintain high availability and reliability. Finally, the remaining operational state is described as reset ($s = 1$). Contrary to intuition, a reset in this context is associated with an unexpected and often undesirable condition. When a reset occurs, the system controller attempts to restore services by forcing the system through the predefined machine state transitions. Consequently, if a reset happens, the system will actively attempt to reestablish its operational state.

During a certain period, the ECU resets were noticed on a much higher threshold than expected, reaching nearly 4% of all events. As described, the system attempted to reestablish the operational state each time a new reset occurred, leading to a disruptive operational condition where production had to be halted, and equipment replacement became necessary. This situation could have been prevented if the proposed PHM framework had been in place, providing early notifications to the operator at the onset of resets, or even before the issue escalated to a critical failure.

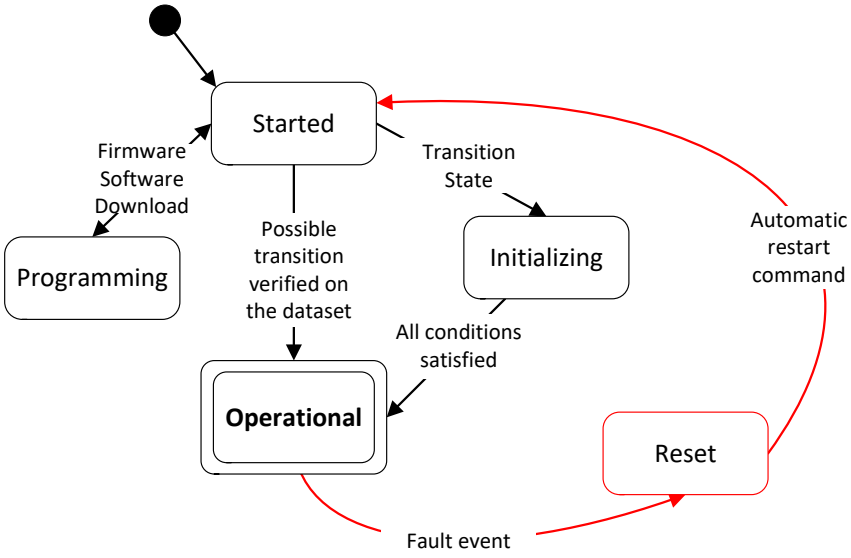

Figure 5.2: System Operational Modes and Transitions

Table 5.2 shows the main characteristics of the available dataset. Additional information about the dataset used in this experiment is found in the Appendix A: full dataset statistical description, including its 47 independent features and the operational mode, is presented in. Table A.1.
Additionally, the 17 features selected during the additive feature attribution process have their density distribution shown in Fig. A.1.

Table 5.2: Real-Case Dataset Description

Dataset Characteristics	Value
Description	System Housekeeping and Process Data
Datapoints (timesteps)	131,040
Pooling Interval (min.)	1
Features (independent variables)	47
Target (dependent variables)	1
% of Resets	4%
% Training Set Split	80%
% Test Set Split	20%

5.1.1 Features Selection Outcomes

The initial step involves feature scaling of the independent variables, which standardizes the features by removing the mean and scaling them to unit variance — resulting in a mean of zero and a standard deviation of one. Herein, the standard scaler technique is applied.
Then, the training and validation sets are defined and split according to Table 5.2.

The tree SHAP algorithm is initially applied, as described in [150], using the full set of 47 (forty seven) available features. This algorithm identifies the contributions of each feature and aids in interpreting their impact on the target calculation. Analysis revealed that, out of the total available features, only 13 (thirteen) demonstrated a significant average impact on the model output magnitude.
Fig. 5.3 shows the selected features and their average contribution to the target estimation. It is worthy noting that the sum of the SHAP value for all relevant features shall equal *"1"*. It was verified that the target estimation using the reduced set of 13 features is identical to the estimation obtained using the full features set.

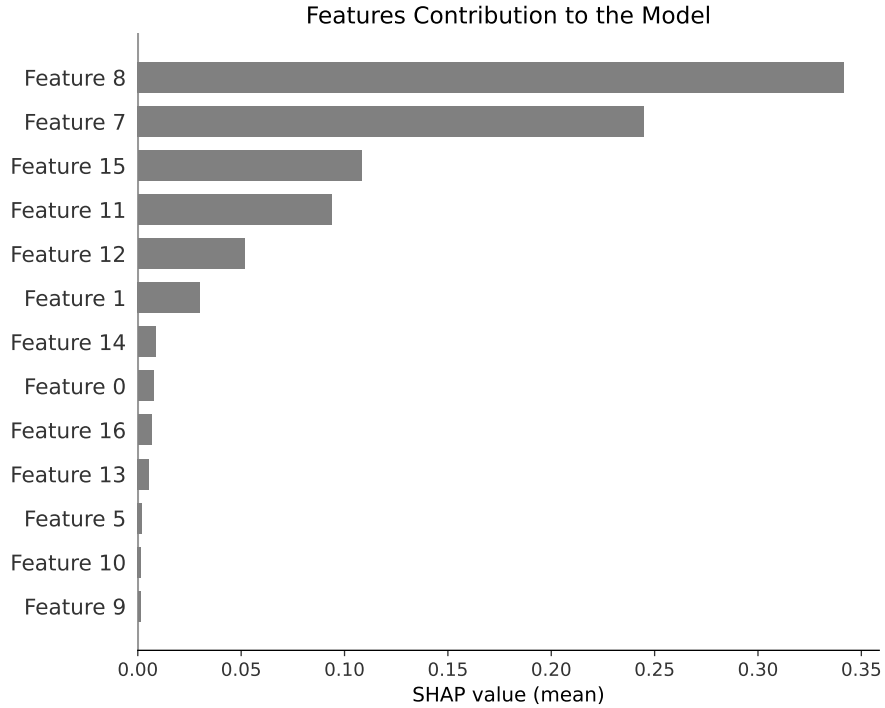

Figure 5.3: SHAP value for Features Impact on the Model

At this point, domain knowledge [142] combined with a human-in-the-loop ap-

proach [167], is utilized. Expert opinion and judgment enter into the practice of statistical inference and decision-making in myriad ways across many domains. A subject-matter expert (SME) reviews the SHAP-identified features and incorporates additional features as needed. In this case, for the current dataset, four additional features were included due to system redundancy requirements, as many variables are polled from two independent controllers, A and B, therefore valuable information can be lost if only one controller is considered. However, the framework proposed in this thesis aims to minimize the number of added features to enable higher process automation. As a result, with the inclusion of four additional features, the final number of independent features retained for analysis was 17.
A preliminary comparison of the main algorithms used in this chapter was conducted on two datasets: the original dataset with 47 features and the optimized dataset with 17 features, refined using the SHAP method. The results are presented in Table 5.3.
The algorithms were not optimized for their hyperparameters, as the primary goal was to assess whether reducing the number of features would result in a significant loss of information that could negatively impact the model's classification accuracy. For reproducibility, training set and test set are split according to Table 5.2. Statistical learning methods such as decision tree (DT), support vector regressor (SVR) and logistic regressor (LR) were executed using their default parameters. Neural network-based methods, artificial neural network (ANN) and long short-term memory (LSTM), were trained as follows:

- ANN: *hidden layers: 2, neurons by layer: 6, activation function: ReLu, optimizer: Adam, loss algorithm: MSE, metrics: accuracy, batch size: 32 and epochs: 100*
- LSTM: *hidden layers: 4, neurons by layer: 50, optimizer: Adam optimizer: Adam, loss algorithm: MSE, metrics: accuracy, batch size: 32 and epochs: 50*

The results indicate that reducing the number of features to 17 does not significantly deteriorate the model's performance compared to the original dataset. While statistical learning methods DT, SVR and LR show a slight decrease in their ability to classify resets, neural network-based methods maintain the same level of predictive performance. This holds true even with static model configurations, without considering further improvements that may be achieved during the optimization.

Table 5.3: Comparison between Original and Modified Datasets

Metrics	RMSE		Accuracy %		Resets		FP		FN	
No. Features	47	17	47	17	47	17	47	17	47	17
DT	4.1e−2	5.1e−2	99.98	99.96	4	5	3	3	1	2
SVR	4.0e−2	4.9e−2	99.98	99.97	4	5	3	3	1	2
LR	4.0e−2	5.4e−2	99.98	99.95	4	5	3	3	1	2
ANN	5.1e−2	4.3e−2	99.97	99.98	5	5	2	2	3	3
LSTM	3.6e−2	3.3e−2	99.97	99.99	2	2	2	2	0	0

5.1.2 Models and Optimization

For this experiment, seven different supervised machine learning models were implemented to evaluate their accuracy and determine if any could serve as a candidate for generalisation. Several of the most common and widely used machine learning algorithms, as identified in [8], [58] and [77] were selected for implementation and testing using the real-world dataset available in this chapter.
As reviewed in Section 2.4, supervised learning is the most prevalent paradigm in Prognostics and Health Management (PHM). Therefore, based on Fig. 2.5, the following popular supervised learning methods were chosen:

- Decision Tree (DT) Regressor
- Support Vector Regression (SVR)
- Logistic Regression (LR)
- Artificial Neural Network (ANN)
- Long Short-Term Memory (LSTM)
- Stack-1: DT + LR
- Stack-2: {DT+SVR} + LR

The Stack-1 and Stack-2 ensembled models are stacked regressors formed by 1) a decision tree regressor and a linear regressor and 2) decision tree regressor together with support vector regressor as base estimator, stacked to a linear regressor.
Ensemble methods are covered in [145] which states that the PHM community

has been using a portfolio of models with ensembles. Ensemble learning helps improve ML results by combining several models. Typically, an ensemble model is a supervised learning technique for combining multiple weak models to produce a stronger one for data sampling. This approach allows the production of better predictive performance compared to a single model and aims to decrease variance, bias and improve predictions [168], [169].

Now that the relevant features are selected, the next step is to optimise the hyperparameters for all the models. Table 5.4 shows each model tested for the dataset, the HPO method used as well as the hyperparameters affected and the resulted model accuracy. It can be noted the very high accuracy obtained on all models fitted to the dataset.
The values in Table 5.4 are respectively the mean and the standard deviation of the *10-folds* cross-validation executions for each algorithm. A higher average score across the folds indicates better generalization of the model. Additionally, the standard deviation is considered to assess the variance of the scores across folds. A lower standard deviation signifies lower variance, which is desirable for model robustness and helps prevent overfitting.

Fig. 5.4 depicts a *Box and Whisker* plot of the 7 models, DT, SVR, LR, LSTM, ANN and the 2 stacked ensemble models. The figure represents the scores for the *k-fold=10* cross-validation.
The graphic shows the minimum and maximum values, limited by the whiskers, the lower and upper quartiles, denoted by the boxes, the median marked by the orange line and some outliers, for the DT and ANN, when they happen.
All models demonstrate very high accuracy in predicting the different states in $y \in \{0, 5\}$. Based on the results shown in Fig. 5.4 and Table 5.4, it is evident that the two stacked models deliver the best overall accuracy.

As mentioned above, both Stacks have been implemented with their default parameters, without optimization. Stack-1 uses a decision tree regressor as a base model, that fits on the training data, and a linear regressor as a meta-model (or final estimator), which learns to combine the predictions of the base model. Stack-2, in its turn, uses two regressors as base models, decision tree regressor with support vector regressor and a linear regressor stacked as a final estimator.
With all models optimized and baselined, a performance comparison can now be conducted to assess their suitability for real-time applications.

Table 5.4: Hyperparameter Optimization for Different Models

MODEL	HPO Method	Hyperparameters Optimised	Model Accuracy	
			Mean	Std Dev
DT	Grid Search	max_leaf_nodes, min_samples_split, max_depth, splitter	0.99204	1.391E-03
SVR	Grid Search	kernel, "C", gamma, degree, coef0	0.99765	7.573E-04
LR	Grid Search	penalty, "C", class_weight, solver, multi_class	0.99841	9.313E-04
LSTM	Random Search	layers, neurons, dropout rate, learning rate, batch size and epoch	0.99897	2.288E-04
ANN	Random Search	layers, neurons, dropout rate, learning rate, batch size and epoch	0.99918	1.025E-03
Stack-1	None	Default	0.99922	5.543E-04
Stack-2	None	Default	0.99931	5.812E-04

5.1.3 Performance Comparison

If $\hat{Y} = \hat{F}_\theta(X)$ is an estimator of Y, then the bias of $\hat{Y}$ is the difference between its expectation and the 'true' value: i.e.

$$bias(\hat{y}) = \mathbb{E}_\theta(\hat{y}) - y \tag{5.1}$$

The estimator $\hat{F}$ is a function of the data, and hence is a random quantity. The main objective is that the estimates are as close as possible of the target true value of "y", whatever that value might be. The mean square error (MSE) represents

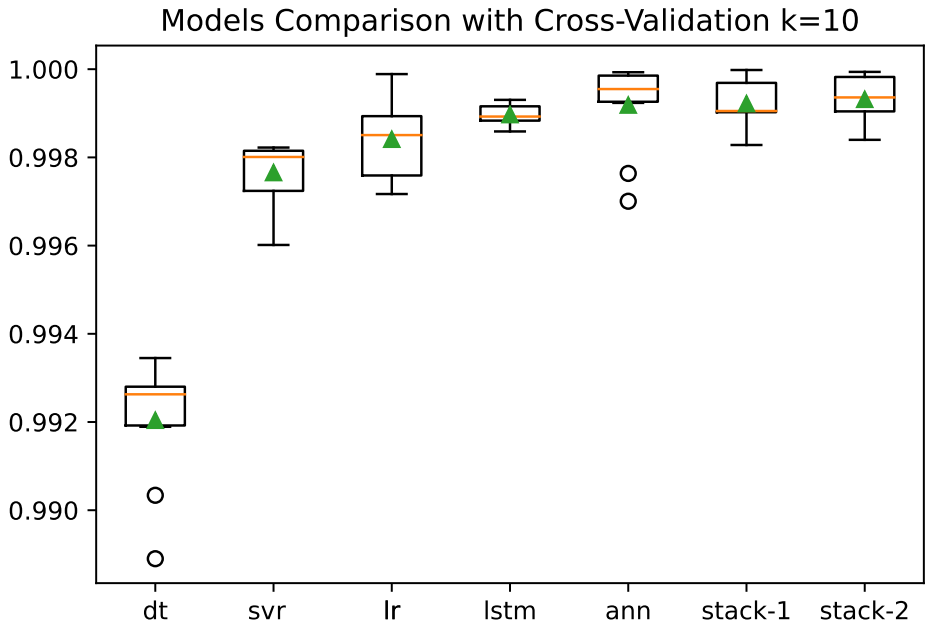

Figure 5.4: BoxPlot for Different Models Accuracy

how close the estimator is of the true value [170], by the expression $\mathbb{E}_\theta[(\hat{y} - y)^2]$, which leads to the relationship between MSE, variance (var) and bias, as shown in eq. 5.2.

$$MSE = \mathbb{E}_\theta[(\hat{y} - y)^2] = var_\theta(\hat{y}) + bias^2(\hat{y}) \tag{5.2}$$

Table 5.5 presents the quantitative evaluation of the prediction performance for each proposed model, using four indicators, along with the test results for the selected case study. The metrics include the MSE, variance, bias of the estimator, and R^2 (R-squared). The objective was to minimize the variance of the regression model. A lower MSE, variance, and bias, along with a higher R^2 indicate a better fit of the model to the data.

The results from Table 5.5, when analyzed alongside Table 5.4 and Fig. 5.4, highlight the relative significance of each model. Stack-1 emerges as the most significant model for the dataset, having the lowest bias, variance, and MSE, along with the highest R^2 score. However, stack-2 performs similarly across these metrics, with only a slightly higher bias (on the order of e^{-4}) but demonstrates better classification performance for reset states. At the other end of the spectrum, the decision tree regressor exhibits the highest error metrics (100 times larger) than those of the stack models, resulting in the poorest state classification performance among the tested algorithms.

Table 5.5: Metrics and Prediction Screening for Different Models

MODEL	Metrics				Inaccurate Predictions (Test Set)			
	MSE	VAR	Bias	R2	All States	Resets	FP	FN
DT	1.5e−2	1.5e−2	4.7e−3	0.9910	43	1	1	0
SVR	1.8e−3	1.8e−3	4.9e−3	0.9989	7	4	2	2
LR	3.3e−3	3.3e−3	5.3e−4	0.9980	7	4	4	0
ANN	1.7e−3	1.7e−3	6.2e−3	0.9989	8	2	1	1
LSTM	1.0e−3	1.0e−3	2.3e−4	0.9994	4	2	2	0
STACK-1	5.9e−4	5.9e−4	1.9e−5	0.9996	4	3	1	2
STACK-2	5.9e−4	5.9e−4	1.1e−4	0.9996	2	1	1	0

For the test results, as design and acceptance criteria, it was selected the total number of inaccurate predictions, for all states and particularly for the resets, and the inaccurate resets where classified between false positives (FP) and false negatives (FN).
In the context of this experiment, a false positive (FP) refers to a prediction outcome that differs from the operational mode *Reset* ($\boldsymbol{s} = \mathbf{1}$), but the predictor incorrectly classifies it as *Reset*.
Conversely, a false negative (FN) in this experiment refers to a prediction outcome where the actual operational mode is *Reset*, but the predictor fails to classify it as *Reset*.
True positive (TP) and true negative (TN) refer to correctly classified instances of the *Reset* state in all circumstances. A TP occurs when the predictor correctly identifies the *Reset* state $\boldsymbol{s} = \mathbf{1}$, while a TN occurs when it correctly identifies a non-Reset state ($\boldsymbol{s} \neq \mathbf{1}$).

The dataset is obtained from a real-world case, a private dataset with consent of usage, with 131,040 data points, as stated in Table 5.2 recorded every minute.
The dataset was split in 80% for training and 20% for test, performing 26,208 data points for testing. In the training set, the number of resets was 1,139, close to the

4% seen in the test set.
The objective is to accurately predict the fault state, $\{y = 1\}$, while minimizing false positives. The overall prediction performance for the other states is also of interest. A random seed was fixed across the different runs and models to ensure reproducibility and facilitate comparison of results.

5.1.4 Execution Times

The results presented in this chapter demonstrate the proposed framework ability to accurately determine the current operational state of the ECU. For benchmarking purposes, the seven models listed above are compared based on their fit execution time and prediction execution time, as proposed in . The target function is defined as $(y \in \{0, 5\})$ for each data point in X.
The computational complexity is assessed by measuring the training and testing times required for each algorithm, as proposed in [134], therefore, two results are provided in this section: Fig. 5.5 presents the time spent to fit the model to datasets of different sizes, representing the model's offline training; and Fig. 5.6 presents the time spent to predict the target value for the same datasets, which is the real-time, or online prediction. All models tested have been optimized according to Table 5.5. The experimental environment of the whole test is described in details in [17].

There is a significant discrepancy between the execution times of different models in both training (fit) and inference (prediction) phases. This is particularly relevant for real-time applications and impacts hardware selection, as discussed earlier.
For example, for the training (fit) phase:

- SVR vs LSTM: Training SVR is about 100 times faster than LSTM on smaller datasets and 10 times faster on larger ones.
- Stack-2 vs LSTM: Training Stack-2 is 10 times faster on smaller datasets and roughly equivalent on larger datasets.

These insights emphasize the importance of selecting a model that balances accuracy and computational efficiency, especially for embedded real-time applications.

However, when considering prediction times, Stack-2 performs well on smaller datasets, making it a promising candidate for real-time applications. For larger

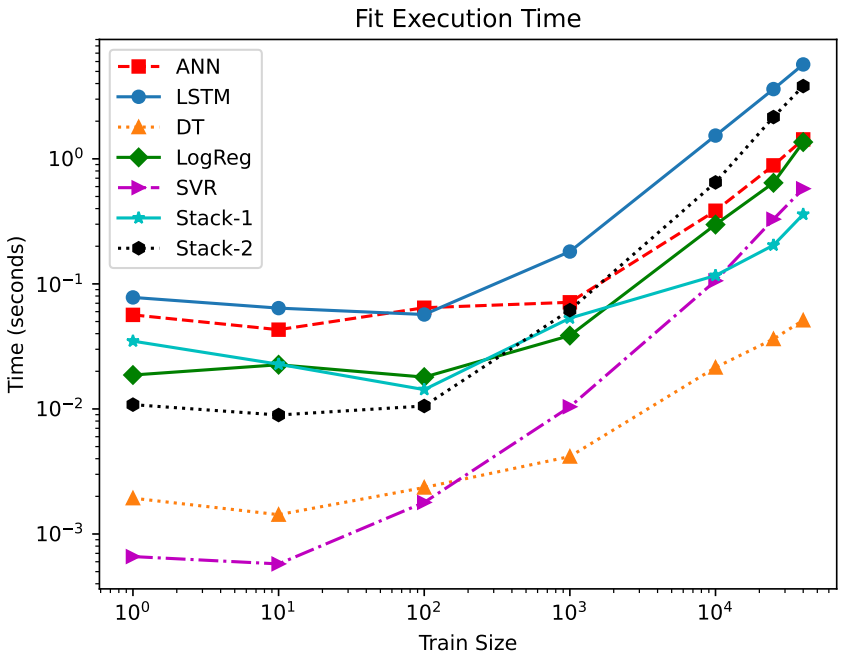

Figure 5.5: Model Fit Time for Different Dataset Sizes

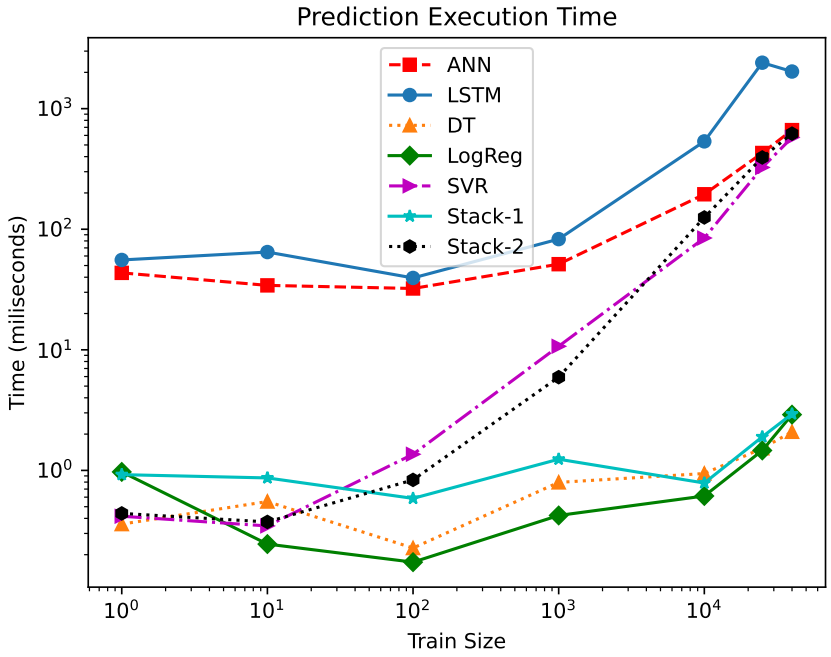

Figure 5.6: Model Predict Time for Different Dataset Sizes

datasets, other models, particularly Stack-1, demonstrate better efficiency. Interestingly, for real-time prediction on very small datasets (below 100 datapoints), all models except ANN and LSTM can achieve frequencies around 2 kHz (2,000

predictions per second) on the current CPU setup. This suggests that lighter models may be more suitable for ultra-fast, real-time state prediction, whereas ANN and LSTM might be better suited for batch processing or larger-scale PHM applications.

Notwithstanding, depending on the hardware platform choice and the system dynamics real-time requirements, neural network-based methods can be perfectly used, subjected to the mentioned constraints. Additionally, as demonstrated in Theorem 3.1 and preliminarily verified in Table 5.3, neural network-based methods are capable to accomodate any continuous function better than other methods.

5.1.5 Results Analysis

It can be observed that the metrics in Table 5.5 closely align with the boxplot in Fig. 5.4, further confirming the consistency and reliability of the model performance.

Overall, the two best models based on the metrics are stack-1 and stack-2, as they have the lowest MSE, variance, and bias, and the highest R^2. Regarding the determination of the output states, stack-2 outperforms all other models, missing only two state classifications, one of which is a reset incorrectly attributed by the model. The only model that, after hyperparameter optimization (HPO), produced inaccurate overall predictions on the test set higher than 10 (ten) data points was the decision tree (DT). This can be correlated with the lower model accuracy observed in Fig. 5.4.

Throughout the development of the case study, the potential and benefits of using machine learning (ML) for the regression of complex systems became evident. Most of the models tested on the real dataset less than 8 inaccurate predictions on a test set of $26,208$ data points, resulting in a model precision score of over 99.97%. This is a very positive outcome, particularly for safety-critical applications, where it is crucial to detect faults accurately, as false positives could trigger unnecessary safety protection measures. It is also worth mentioning that using techniques of xAI, the relevant features can be selected and for the fault diagnosis and root cause analysis, the main contributor for the system faults can be narrowed down and eventually uncovered.

It was also noted that, as stated by [145], the two best models for the metrics

and accuracy were the ensemble models, stack-1 and stack-2. These models were stacked regressors formed by 1) a DT and a linear regressor and 2) DT together with SVR as base estimator, stacked to a linear regressor. By combining 2 or more weak models, it was possible to achieve a robust estimator for the case study. Also regarding execution times, both stack-1 and stack-2 demonstrated implementation feasibility.

Despite its higher computational cost, LSTM showed promising results in reset classification. Given its flexibility and configurability, it remains a model worth further investigation. One of LSTM's key advantages is its ability to leverage an N-timestep sliding window as input, allowing it to be fine-tuned to meet specific performance criteria. Additionally, LSTM can simultaneously predict a future horizon H, making it highly valuable for multi-step ahead forecasting, which is further explored in the next section.

5.2 Multi-step Ahead Forecast

For the multi-step ahead forecasting using this experiment's dataset, a large number of models combining MSA strategies with different machine learning models have been implemented and tested herein and reported in [124].

The implementation of the method described in Section 4.4 is done in three steps, exploring different strategies and models:

- **MSA Strategies**: Recursive and MIMO.
- **MSA Statistical Models**: Autoregression (AR) and ARIMA.
- **Predictor Models**: One-Step Ahead (OSA) and Multi-Step Ahead (MSA) versions of Stacked-LSTM, Bidi-LSTM and CNN-LSTM.

This thesis compares the Recursive and MIMO strategies, and the statistical models AR and ARIMA, as presented in Section 4.4 used to forecast the multivariate independent features $X's$. Subsequently, the forecasted $X's$, as described in (4.9), are used as input data for the pre-trained predictor models mentioned above, both in OSA and MSA implementations. These models will then provide the operational state of the system as their target value.
The results are compared to determine which method offers higher accuracy and a better fit for the target trends, particularly in cases of interest where an anomaly occurs, leading to a change in the system's operational state.
Fig. 5.7 depicts the multi-step ahead forecasting strategies that were implemented and tested in this study, while Table 5.6 presents the resulting combination of strategies and models.

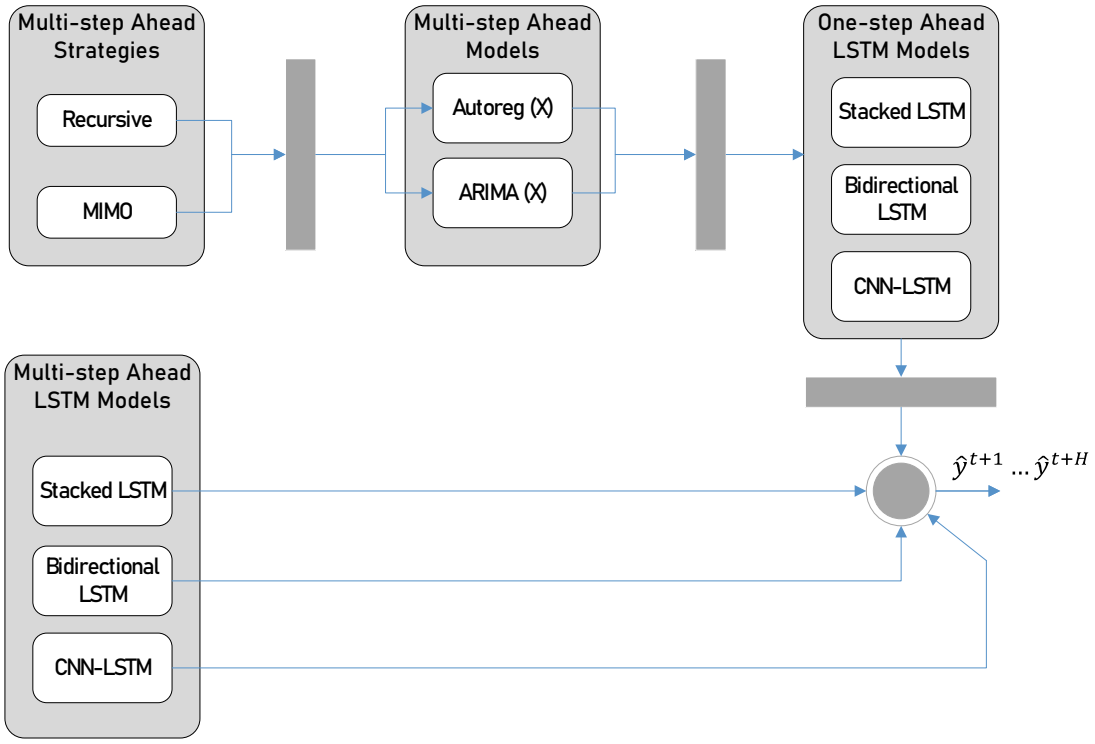

Figure 5.7: Multi-Step Ahead Forecasting Strategies and Models Tested

In real-time, we propose to use the last n-readings of each selected individual feature as input for the statistical model to predict its immediate horizon H. Therefore, each individual feature will generate an array of data that can be represented by the vector in eq. (5.3).

$$X_{t+H} = [x_{t-n} \;\; x_{t-n-1} \ldots \; x_{t-1} \;\; x_t \;\; x_{t+1} \;\; \ldots \; x_{t+H-1}] \tag{5.3}$$

Where the set $X^i = \{x^i_{t+1}, \; x^i_{t+2} \;\; \ldots \; x^i_{t+H-1}\}$ is the MSA forecast of the specific feature in X. The collection of all N features will be represented then by the matrix given in (5.4).

$$X^{0\ldots N}_{t+H} = \begin{bmatrix} X^0_{t+H} \\ X^1_{t+H} \\ \vdots \\ X^N_{t+H} \end{bmatrix} = \begin{bmatrix} x^0_{t-n} & x^0_{t-n-1} & \ldots & x^0_{t+H} \\ x^1_{t-n} & x^1_{t-n-1} & \ldots & x^1_{t+H} \\ \vdots & \vdots & \ddots & \vdots \\ x^N_{t-n} & x^N_{t-n-1} & \ldots & x^N_{t+H} \end{bmatrix} \tag{5.4}$$

Table 5.6: Real-Case Dataset Description

Method	Description
#1	{MSA} {Stacked-LSTM}
#2	{MSA} {Bidirectional-LSTM}
#3	{MSA} {CNN-LSTM}
#4	{Recursive} {Autogression} {Stacked-LSTM}
#5	{Recursive} {Autogression} {Bidirectional-LSTM}
#6	{Recursive} {Autogression} {CNN-LSTM}
#7	{Recursive} {ARIMA} {Stacked-LSTM}
#8	{Recursive} {ARIMA} {Bidirectional-LSTM}
#9	{Recursive} {ARIMA} {CNN-LSTM}
#10	{MIMO} {Autogression} {Stacked-LSTM}
#11	{MIMO} {Autogression} {Bidirectional-LSTM}
#12	{MIMO} {Autogression} {CNN-LSTM}
#13	{MIMO} {ARIMA} {Stacked-LSTM}
#14	{MIMO} {ARIMA} {Bidirectional-LSTM}
#15	{MIMO} {ARIMA} {CNN-LSTM}

5.2.1 Forecast Evaluation

Several performance metrics have been proposed for the evaluation of multi-step forecasting models, as discussed in [89], [171], [172], and [173]. The use of scaled errors, which are independent of the data scale, is recommended in [89].

Two metrics are used: the mean absolute scaled error (MASE) and the root mean squared scaled error (RMSSE). MASE and RMSSE are respectively, scaled metrics of the mean absolute error (MAE) and the root mean squared error (RMSE), by dividing them by the naive multi-step method for the dataset.

The dataset presented a challenge of non-uniformity in the state (target) values, particularly during transition periods, as shown in Fig. 5.8. Depending on the

system's state transition, this variation could lead to different metric values when absolute numbers are used.
To address this, we propose using a selection of scaled and normalized metrics, which will produce results within the range of $\{0, 1\}$, allowing for comparisons regardless of the specific transition step considered.

Three metrics are selected, which are the scaled mean absolute error (sMAE), the normalised root mean squared error (NRMSE), as in [173] and the symmetric mean absolute percentage error (sMAPE) as in [89].

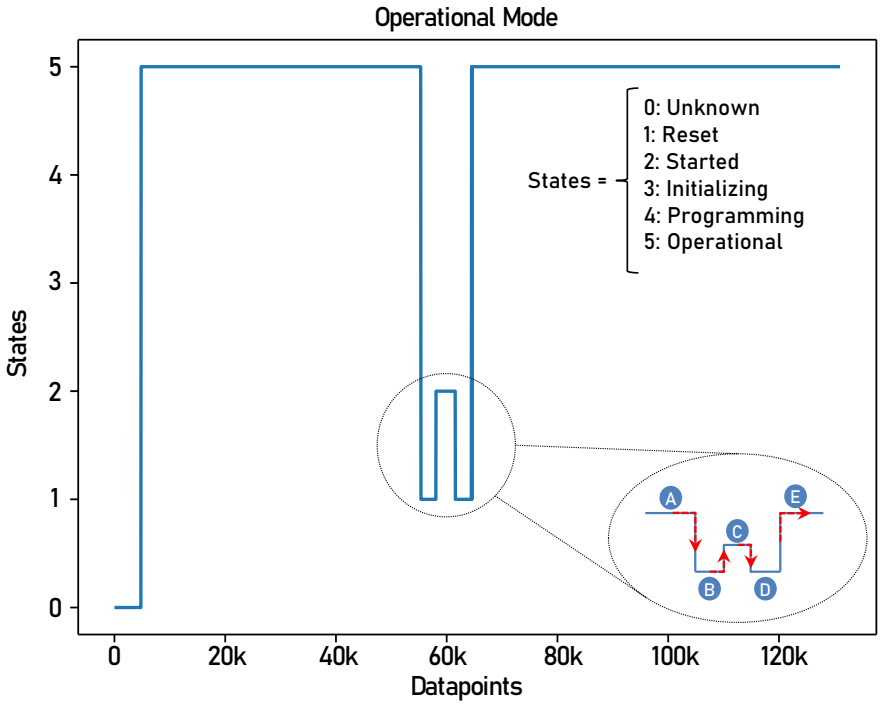

Figure 5.8: Operational Modes in the Dataset

The equations are as follows:

$$sMAE = \frac{1}{n}\sum_{t=1}^{n}\frac{|e_t|}{\frac{1}{T}\sum_{t=1}^{T} y_t} \tag{5.5}$$

where n is the number of data points, y_t are the observed (or true) values and $\hat{y}_t$ the predicted ones. The scale-dependent error e_t is defined as:

$$e_t = y_t - \hat{y}_t \tag{5.6}$$

We used the NRMSE, as defined in [173]:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(e_t^2)}}{\sum_{t=1}^{n}(|y_t|)} \tag{5.7}$$

Equally, sMAPE definition is given below:

$$sMAPE = \frac{1}{n}\sum_{t=1}^{n}(\frac{200|e_t|}{|y_t| + |\hat{y}_t|}) \tag{5.8}$$

sMAPE fixes the type of asymmetry seen with MAPE that the penalisation is different if y_t and $\hat{y}$ are exchanged [173]. sMAPE is more resilient to outliers compared to metrics without error bounds [89]. One of SMAPE advantages is that it makes it easier to analyze the output in a statistical way. According to its mathematical definition, SMAPE is non-dependent on the scale of the data since it enforces symmetry and properties that showcase diminishing high levels of biases [83].

For the sMAE, the error at timestep t in the forecast horizon is scaled by the mean of the actual values in the whole training region of the series, where error measures are scaled by a scaling factor dependent on the time series to make them scale-free. The NRMSE is the normalised RMSE, which relates the RMSE to the observed range of the variable.

These metrics were chosen because the target has discrete values and depending on the current state, the difference between states is not the same.
Similarly to [173], where a win-loss ranking is proposed based on MASE and RMSSE, we propose a KPI (key performance indicator) which combines the effects of the tested models accuracy and the proposed metrics, sMAE, NRMSE and sMAPE. This KPI searches for the models with the lowest prediction error, both relative and absolute, while privileging the higher accuracy. The proposed KPI is given in (5.9). Since we are looking for lowest prediction errors and highest accuracy, the smaller the KPI, the better the proposed model will fit into the dataset.

$$KPI = \frac{sMAE * NRMSE * sMAPE}{Accuracy} \tag{5.9}$$

5.2.2 Case Study Elicitation

Fig. 5.8 depicts the states throughout the whole dataset. The *x-axis* refers to a time period, with measurements taken every minute, therefore *120k* means *120,000* minutes.
During a certain period, as highlighted in Fig. 5.8, the ECU resets and fails to recover as expected. This behaviour is exactly what the proposed framework is designed to detect at the earliest possible opportunity.
In real-time, it is proposed to use the last N readings of each selected individual feature as input for the statistical model to predict its immediate horizon H.

For the implemented case herein, $N = 30$ (called as look-back window) is selected as a good compromise between module accuracy and real-time performance requirements. It can be seen in Fig. 5.9 that for $N = 30$ the model presents a median similar to $N = 60$ and $N = 120$ and the distance between the first quartile (Q1) and the third quartile (Q2) is less spread than the other look-back windows. Therefore, selecting $N = 30$ does not degrade the model prediction and enables real-time performance.

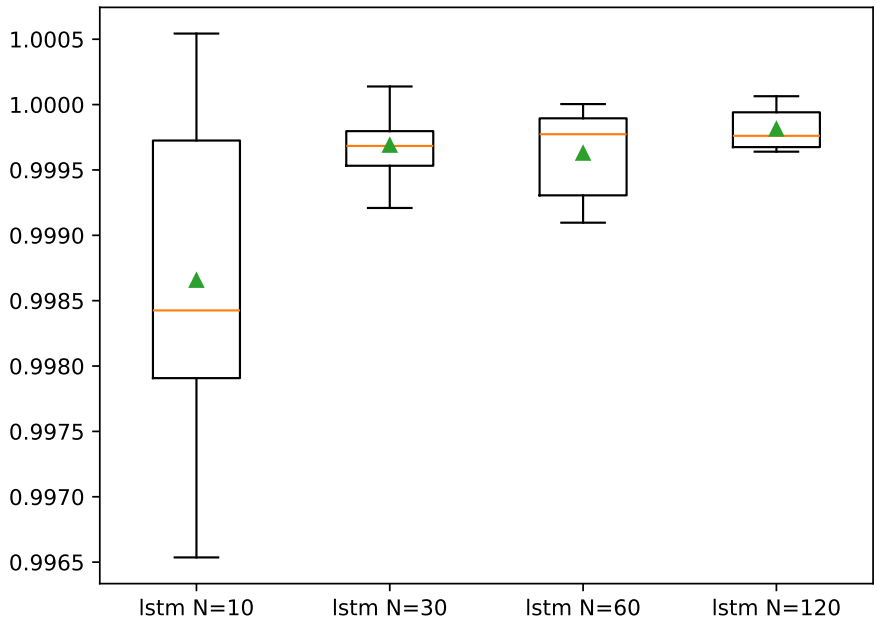

Figure 5.9: Look-Back Comparison for Real-Time LSTM Implementation

Similarly, after conducting multiple simulations, it was observed that a forecast horizon of $H = 30$ provides the best results for multi-step ahead (MSA) forecasting.

This horizon of $H = 30$ corresponds to an advisory window of 30 minutes, meaning the model forecasts 30 steps ahead.

All LSTM models were trained offline using the same dataset, with the operational data depicted in Fig. 5.8 being used for real-time inference. It can be observed that the system remains in its operational state, $\boldsymbol{s = 5}$, for the majority of the time, as expected. However, during a period centered around 60k on the *x-axis*, the system transitioned to different states, indicating a system failure.
The models were run over the whole range of the data as unseen data for the purpose to analyse the performance and rank the models for selection to test on embedded devices. The state transitions $A \rightarrow B$, $B \rightarrow C$, $C \rightarrow D$, $D \rightarrow E$ are of particular interest to verify whether the proposed models can predict a state change, or in practical terms, predict an imminent failure.
It can also be noted in Fig. 5.8, a collective anomaly type, as described in [91], [174] and [175], which is a collection of related data instances anomalous with respect to the entire data set. The individual data instances in a collective anomaly may not be anomalies by themselves, but their occurrence together as a collection is anomalous.

According to Fig. 5.7 and Table 5.6, the different methods implemented and tested are arranged in the following order:

A. three MSA LSTM models, as shown in Fig. 4.4, trained offline and deployed in real-time straight over the current data (a moving look-back window of $N = 30$ registers), with no intermediate steps or assessment of the features of the system; and
B. the one-step ahead LSTM models that, once deployed on the target device, run over a range of features data (a moving window of $N + H = 60$ registers), where N are the last look-back readings and H are forecasted by either AR or ARIMA. A new 60 registers set is provided every timestep (minute) for the LSTM model to compute the predicted target value. As shown, two different forecast strategies are used, Recursive and MIMO [176]; two different real-time features forecast are also used, Autoregression and ARIMA; and the pre-trained OSA LSTM models are stacked, bidirectional and CNN-LSTM, as mentioned before.

All combinations, 15 in total were implemented and tested.

5.2.3 Models and Optimization

The hyperparameters for the implementation of the three different LSTM models: stacked, bidirectional and ensembled CNN-LSTM are shown in Table 5.7. The hyperparameters have been optimised according to subsection 4.2.3 and were kept the same both for OSA and MSA outputs.

Table 5.7: Hyperparameters for all Models

Symbols	**Parameter**	**Stacked-LSTM**	**Bidi-LSTM**	**CNN-LSTM**
H_L	Hidden Layers	3	3	3
N	Neurons/Layer	50	64	64
t	Timesteps	30	30	30
d	Dropout Rate	0.0	0.5	0.5
D_L	Dense Layers	{1,30}	{1,30}	{1,30}
b	Batch Size	32	32	64
ϕ	Nonlinearity	ReLU	ReLU	ReLU
opt	Optimiser	Adam	Adam	Adam
$\{L_S(\theta)\}$	Loss Function	MSE	MSE	MSE
{-}	Metrics	Accuracy	Accuracy	Accuracy
η	Learning Rate	0.001	0.001	0.001
ϵ	Epsilon	1.00E-07	1.00E-07	1.00E-07
		CNN-LSTM Only		
–	Filter Size	-	64	-
K	Kernel Size	-	1x1	-
MP	Pool Size	-	2x2	-

5.2.4 Models Ranking

With the algorithm, metrics and models establish it is possible to evaluate the named methods and rank them. Table 5.8 shows the overall accuracy, metrics and ranking, according to the proposed KPI for each one of the 15 methods.

Table 5.8: Overall Accuracy and Ranking for all Models

Method	Accuracy %	sMAE	NRMSE	sMAPE	KPI	Ranking
#1	**70.77**	**0.45**	**0.51**	**0.48**	**0.46**	**5**
#2	66.92	0.38	0.42	0.40	0.46	7
#3	68.97	0.46	0.53	0.48	0.61	9
#4	77.69	0.59	0.71	0.57	0.57	8
#5	75.90	0.88	0.97	0.68	1.55	13
#6	69.74	0.98	1.05	0.65	3.12	15
#7	78.21	0.55	0.64	0.57	0.46	6
#8	**79.23**	**0.49**	**0.55**	**0.54**	**0.31**	**1**
#9	70.00	0.58	0.65	0.49	0.85	10
#10	77.95	0.86	0.97	0.65	1.26	11
#11	**79.49**	**0.49**	**0.56**	**0.59**	**0.35**	**3**
#12	71.03	1.09	1.17	0.71	3.08	14
#13	**78.21**	**0.54**	**0.63**	**0.56**	**0.44**	**4**
#14	**78.97**	**0.49**	**0.55**	**0.54**	**0.32**	**2**
#15	69.49	0.71	0.79	0.53	1.44	12

It can be observed that an overall accuracy of nearly 80% can be achieved in real-time for some of these methods. The lower the KPI, the better the proposed model fits the dataset, including during the collective anomaly period.

The best five methods are highlighted in bold in Table 5.8 and were implemented in the proposed hardware platforms. One should note that the Bidi-LSTM is absolutely predominant in the results. The strategies MIMO and ARIMA also

indicate a prevalence of these models. The ranking only take into consideration the model fitness to the data, not considering the execution times which are looked at the next subsection.

5.2.5 Real-time Inference

The five best ranked methods were implemented in two hardware platforms, standard testbeds available:

- **Raspberry Pi 4 (RP4)**: using a Broadcom BCM2711 SoC with a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor, with 1 MB shared L2 cache.
- **Ultra96-V2 (U96)**: running a quad-core 1.5 GHz ARM Cortex-A53, with 1MB L2 cache.

ARM processors are the most widely used architecture for embedded systems due to their low power consumption, high performance, and cost-effectiveness. The Ultra96-V2 (U96) is a MPSoC (multiprocessor SoC) and has a wealthy set of processors and a FPGA. The MPSoC is a good candidate to fulfill the requirements of a mixed-criticality system, as implemented in [21]. The algorithms were executed on the referred ARM processors. Fig. 5.10 shows the results for the selected methods, #1, #8, #11, #13 and #14. It can be depicted by the chart that the method #8 takes around 270 s to run on RP4 and 192 s on U96.

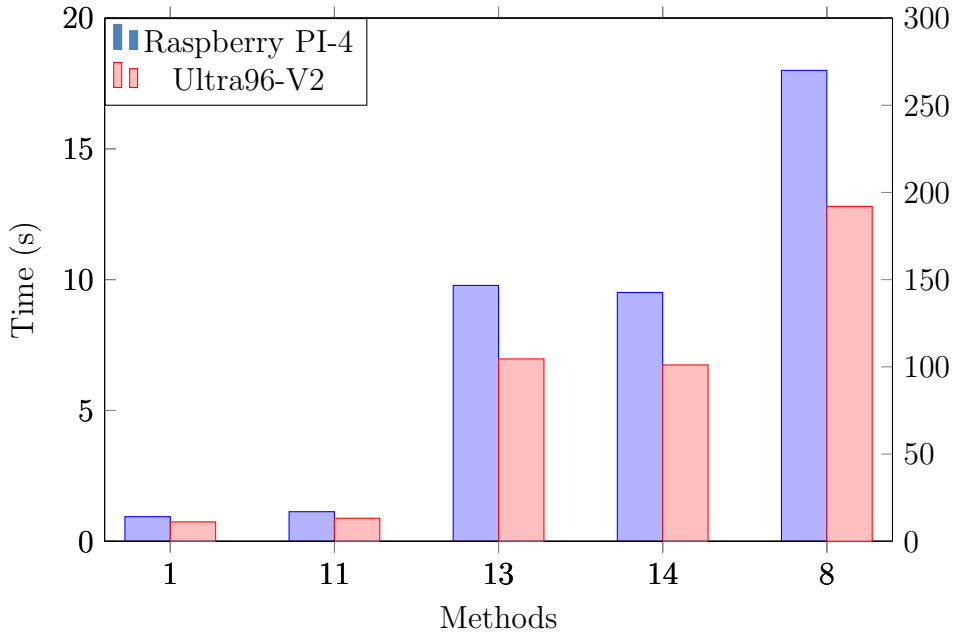

Figure 5.10: Execution times for the selected Methods. [Method #8 refers to the y-axis on the right]

5.2.6 The Novel Framework Algorithm

Besides method #8 ranks as first in Table 5.8, its implementation in real-time is not feasible, as shown in the Fig. 5.10, which would take nearly 200 s for each new estimate which exceeds by far the new data acquisition time of 60 s. The second best ranked method is then #14, which is a combination of {MIMO}{ARIMA}{Bidi-LSTM}.

Algorithm 1 shows then the preferred method implementation for real-time applications.
The method implements a {MIMO}{ARIMA} model that utilises the most recent $N = 30$ registers to forecast the horizon of the next $H = 30$ X's.
Once the new features set is available, a Bidi-LSTM network, with parameters and hyperparameters according to Table 5.7 predicts the target values of the next 30 operational states, providing therefore an advisory window.
It has also been demonstrated empirically that during the state transitions, the moving average of the target value changes towards the new status. Thereby, this measurement is used in conjunction with $\hat{y}_k$ to determine the system status.

Algorithm 1: Algorithm for real-time multi-step forecasting of embedded system operational state

Input : Load the Bidi-LSTM model $\hat{F}_\theta$, read the last $N = timesteps$ $X's$ available datapoints

Forecast the Horizon (H) of the Features ($X_{0...N}$);
for $i = 1$ **to** n **do**
    model $\leftarrow$ ARIMA(p, d, q).fit
    $\hat{X} \leftarrow$ model.predict$(N = timesteps)$
end
X.append$(\hat{X})$ #Adds $\hat{X}'s$ to the datapoints
$\hat{y} \leftarrow \hat{F}_\theta(X)$ #Calculates y_{pred} for the measured values and estimate it for $\hat{X}'s$
Calculate the weighted moving average (WMA) of last 10 cycles of $\hat{y}_{avg}$;
if $|WMA(\hat{y}_{avg})| <> y_t$ **then**
    return "$faulty$" $status$;
else
    return "$operational$" $status$;
end

Output : $status$

5.2.7 Results Analysis

The proposed framework is a multi-step ahead forecasting system that can be implemented on most embedded systems, where the system's status can be interpreted from the data collected by sensors and actuators. The main limitations are related to data availability and execution times.

The results in this thesis show that the bidirectional LSTM outperforms the other models, stacked LSTM and the ensemble CNN-LSTM. This finding is consistent with the results reported in [22], [123], [125] and [177].

The ARIMA method showed to be more suitable for the features real-time forecasting as well, as in Methods #8, #14 and #13, given its capacity to differencing of actual observations in order to make the time series stationary and lags of the forecast errors of the moving average to consider the trends [23], [162], instead of relying only on a relationship between the current observation and past observations, as the autoregressive method does.

For the MSA forecasting strategies, MIMO can be selected over Recursive for being much less computationally intensive, as shown by Fig. 5.10. Additionally, it learns one multiple-output F model from the time series $\{x_{t-n}, x_{t-n-1} \ldots x_t\}$ where the

forecasts are returned in one step by a multiple-output model $\hat{F}$ and it preserves between the predicted values, the stochastic dependency characterizing the time series [80]. Finally, it avoids the accumulation of errors found in the Recursive strategy.

The proposed algorithm aims not only to determine the correct system status during steady-state operation but also to flag any operational status changes as early as possible, providing an advisory window for the operator. It was empirically demonstrated that by calculating the weighted moving average of the last 10 cycles of the predictor, the algorithm can effectively determine the system status — whether faulty or operational — since the moving average of the target value shifts toward the new status.

For the execution times, among the short-listed methods, only Method #8 exhibits significantly higher runtime compared to the other methods, as shown in Fig. 5.10. As a result, it cannot be considered for implementation in real-time.

5.3 Chapter Summary

A valuable real-world case dataset was used in this chapter to implement the novel PHM framework proposed. The dataset spans over 2,000 hours of data collection, equivalent to more than 90 days. It was collected from an operational ECU that began experiencing multiple faults, leading to intermittent resets that prevented it from performing its expected functions. The novel PHM method was therefore employed to determine the ECU's operational state in real-time and to forecast its future state.

The data-driven PHM methodology proposed in Chapter 4 was applied to determine the ECU's operational state, incorporating feature selection and the optimization of seven supervised machine learning models: Decision Tree (DT), Support Vector Regression (SVR), Logistic Regression (LR), Long Short-Term Memory (LSTM), Artificial Neural Networks (ANN), and two stacked ensemble models. Overall, a model precision score exceeding 99.97% was achieved, demonstrating highly positive results, particularly for safety-critical applications. These findings confirm that, for this dataset, data-driven machine learning algorithms can be effectively utilized and trusted for PHM in high-criticality systems, addressing [**RQ1**].
It is also worth noting that explainable AI (xAI) techniques enable, the relevant features can be uncovered, favouringe fault diagnosis and root cause analysis, by shortlisting the main contributor for the system faults, addressing [**RQ2**].

For the multi-step-ahead (MSA) forecasting evaluation, 15 different strategy combinations were tested to determine the best fit for the dataset. To account for data non-uniformity caused by sudden state transitions, a set of scaled and normalized metrics was introduced. A combination of these metrics was proposed as a key performance indicator (KPI) for evaluating model performance. The most effective MSA hybrid approaches were selected to assess whether an advisory window could predict an impending reset event. Results showed that a forecasting horizon of 30 future timesteps could be achieved with nearly 80% accuracy.

Real-time requirements were also evaluated by measuring the execution times of all seven models across datasets of varying sizes. While neural network-based algorithms had the highest execution times, they remained viable with appropriate hardware selection. The top-performing multi-step ahead forecasting methods

were implemented on two embedded system platforms: Raspberry Pi 4 (RP4) and Ultra96-V2 (U96), both equipped with ARM processors. Most strategies met the real-time constraints, with the best performance achieved by a hybrid MSA model. This approach combined statistical methods — MIMO and ARIMA — to forecast explanatory variables, followed by an optimized bidirectional LSTM (Bidi-LSTM) for predicting the next 30 operational states.

Chapter 6

Experiment 2: Battery State-of-Charge Estimation

This chapter evaluates the novel PHM framework proposed in Chapter 4, validating and extending the results obtained in the previous experiment in Chapter 5, using a new dataset consisting of Lithium-ion (Li-ion) batteries.
The experiment follows the same PHM framework, incorporating noise reduction, feature selection and model optimization to enable real-time inference and multi-step ahead forecasting. This serves to validate and generalize the findings from the previous chapter, confirming that the Bidi-LSTM architecture, offline schema and real-time deployment are suitable for implementation on the same hardware platform. Additionally, it ensures that the results are reliable and can be reproduced effectively.

Initially, in this chapter, the battery is modelled using a second-order $2RC$ Thevenin model, and model-based state of charge estimators are proposed, Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF), as described in Section 3.2.1 and Section 3.2.2. Then, the novel PHM framework is employed to develop an alternative data-driven battery SOC estimator. This approach involves: applying noise reduction techniques to the original dataset, performing feature selection to

refine the dataset and utilising a specialized bidirectional long short-term memory (Bidi-LSTM) network for state-of-charge (SOC) estimation. A thorough comparison between the two model-based methods, EKF and UKF, and the novel PHM data-driven approach is conducted across ten (10) different datasets, each representing a distinct drive cycle that the battery set has been subjected to.
Finally, using the same Bidi-LSTM algorithm, optimized with the same hyperparameters, a battery SOC multi-step ahead forecast estimator is proposed. This estimator is capable of predicting future SOC values, supporting failure prediction and predictive maintenance. The chapter concludes with a summary of the main findings.

Electric vehicles (EVs) have been widely introduced into the transportation sector aiming to accelerate decarbonization and reduce its emissions. Most of EVs nowadays use lithium-ion (Li-ion) batteries, with their different chemistries [178].
Li-ion batteries state of charge (SOC) estimation accuracy is critical for the battery cells lifetime and its safe operation for EV applications [179].
Modelling a battery is a difficult task due to the complex nonlinearity and time-variability of the system and various factors that may affect the battery performance. Different models have been proposed: electrochemical model (EM) used mainly for life prediction and cell degradation purposes; equivalent circuit model (ECM) used for battery management systems (BMS); and electrochemical impedance model (EIM) mainly used as a non-destructive characterization technique to determine the electrical response of chemical systems [127].
In many works, the model-based methods, such as ECM, are used in association with adaptive filters and state estimation algorithms. The most well-known algorithms for battery' state estimation include Kalman filters and its variants [127].
Other methods include particle filter [180], H∞ filter [138], [181], sliding-mode observer [182], [183] and others.

An in-depth feasibility study of using Kalman filters (KF) for the state estimation of Li-ion batteries is presented in [98]. It implements a wide variety of KFs for a total number of 224 Panasonic NCR18650PF NCA cells with a nominal capacity (C_{nom}) of 2.85 Ah used to construct two battery modules.
A ternary Li-ion battery SOC was estimated using the unscented Kalman filter (UKF) in [101] with the maximum absolute errors below 3%.

A method using Fractional Unscented Kalman Filter (FUKF) algorithm is used

by [184] to estimate the battery SOC. This method was employed in both static and dynamic battery discharging experiments. The researchers took into account the fractional properties inherent in nonlinear systems. Fractional-order systems employ dynamic models with non-integer orders, which are based on fractional calculus involving differentiation or integration of non-integer order. The paper proposed a second-order RC model for the battery and then applied the unscented transformation technique, which approximates the probability distribution of the variable using sigma points. The results show an SOC estimate mean error below 3% for the FUKF.

Conversely, the emergence of big data and powerful computers has paved the way for the development of relatively new approaches in data-driven SOC estimation. These methods, also referred to as black box models, rely on empirical observations rather than extensive knowledge of the underlying processes [127]. One of this methods is the long short-term memory network (LSTM), which exhibits faster convergence to the true SOC compared to the unscented Kalman filter (UKF) when the initial SOC is inaccurate. The LSTM achieves a root mean square error (RMSE) and mean absolute error (MAE) of less than 2% and 1% respectively [185]. Moreover, the LSTM can accurately assess SOC by solely monitoring battery measurements such as current, voltage, and temperature. It does not rely on information regarding battery internal chemistry, complex reactions, or model parameter estimation [186].

The novel data-driven PHM method proposed in Chapter 4 is used herein for the batteries SOC estimation, combining a robust offline trained machine learning model, based on a bidirectional LSTM, with an online inference deployment, to forecast the SOC in real-time.

6.1 Battery Modelling

The SOC is the ratio between the battery available capacity $Q_{available}$ and its rated capacity Q_n:

$$SOC_k = \frac{Q^k_{available}}{Q^k_n} \tag{6.1}$$

Despite of the apparent simplicity of the equation above, $Q_{available}$ and Q_n are not trivial to quantify at every timestep k.

The widely used SOC estimation method is the Ah (Ampere/hour) Coulomb-counting method, as used in [184], and shown in (6.2).

$$SOC_k = SOC_{k-1} + \frac{\eta}{Q_n} \int_{k-1}^{k} \mathrm{I}_k \, \mathrm{d}k \tag{6.2}$$

where η is the charge-discharge efficiency, which is related to the battery working temperature, charge and discharge rate and other aspects. The instant current drawn from the battery at every timestep k is represented by I_k.

The Thevenin model is used as typical battery equivalent circuit model (ECM), which is designed using one or multiple RC (resistance-capacitance) groups, another resistance and a voltage source [127], as depicted in Fig. 6.1.

Compared with the first order $1RC$ model (with a single RC branch in the electrical circuit), the estimation error in the second-order $2RC$ model is smaller. This means that the $2RC$ model has better accuracy, stability, and robustness. The $2RC$ model outperforms the $1RC$ and the third-order $3RC$ models and reduces the computational cost [187].

Another battery ECM has been presented and compared to the second-order $2RC$ Thevenin model in [188]. For that model, developed by the National Renewable Energy Laboratory (NREL), the equivalent circuit has the components R_b, R_s, C_b, C_s and R_t. The terminal resistance R_t models the voltage drop when the battery has a load connected, R_s and C_s model the diffusion effects of the battery, and the bulk resistance R_b and bulk capacitor C_b represents the battery storage capacity.
The conclusion of the study in [188] is that the $2RC$ Thevenin model used herein offers a better performance.

Therefore the second-order $2RC$ is selected herein to model the battery.

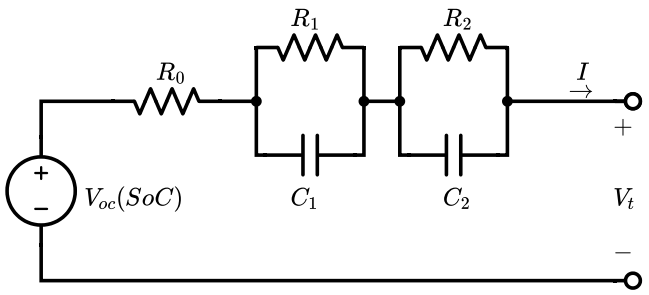

Figure 6.1: Second-Order RC Thevenin Battery Model

In the $2RC$ Thevenin model, R_0 represents the ohmic resistance, which includes the resistance of contacts, electrodes, as well as electrolytes. The double pair RC captures the transient battery dynamics such as the charge transfer kinetics, the Lithium-ion diffusion, and solid/electrolyte interface (SEI) dynamics. The voltage source VOC represents the OCV (open circuit voltage), which mainly depends on the battery SOC.

Considering Kirchhoff's voltage law across the full loop, and defining $\tau_n = R_n C_n$ as the RC circuit time constant, where $n \subset [1, 2]$, it comes:

$$Vt_k = OCV_k - R_0 I_k - V1_k - V2_k \tag{6.3}$$

$$V1_k = V1_{k-1} e^{-\Delta t/\tau_1} + R_1(1 - e^{-\Delta t/\tau_1}) I_k \tag{6.4}$$

$$V2_k = V2_{k-1} e^{-\Delta t/\tau_2} + R_2(1 - e^{-\Delta t/\tau_2}) I_k \tag{6.5}$$

Combining (6.2) to (6.5), it provides the matrix in the form of state-space (6.6):

$$\begin{bmatrix} SoC_k \\ V1_k \\ V2_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{-\Delta t/\tau_1} & 0 \\ 0 & 0 & e^{-\Delta t/\tau_2} \end{bmatrix} * \begin{bmatrix} SoC_{k-1} \\ V1_{k-1} \\ V2_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{\eta * k}{Q_n} \\ R_1(1 - e^{-\Delta t/\tau_1}) \\ R_2(1 - e^{-\Delta t/\tau_2}) \end{bmatrix} * \mathrm{I}_k \tag{6.6}$$

The state vector $\mathbf{x}$ contains the overpotentials of the RC terms and the SOC:

$$\mathbf{x}_k = [SOC_k \; V1_k \; V2_k]^T \tag{6.7}$$

6.2 Experimental Setup

A comprehensive review of publicly available batteries datasets is presented in [189]. The dataset selected herein is the one available in [190] (under 'CC BY 4.0'), which has been cited in multiple publications, such as in [191], [192] and [193].

This battery testing dataset covers four typical driving cycles, US06, HWFET, UDDS and LA92 and a mix of other cycles. It contains data from a single 2.9 Ah NCA (Lithium Nickel Cobalt Aluminium Oxide) Panasonic 18650PF cell. A brand new 2.9 Ah Panasonic 18650PF cell was tested in an 8 $cu.ft.$ thermal chamber with a 25 A, 18 V Digatron Firing Circuits Universal Battery Tester channel. The cell was cycled according to the above driving cycles and an additional "neural network driving cycle" systematically through a range of temperatures (25 °C, 10 °C, 0 °C, -10 °C, and -20 °C, in that order).

The dataset includes characterisation data from Hybrid Power Pulse Characterisation (HPPC) and Electrochemical impedance spectroscopy (EIS) tests, and in-cycle measurements from the driving cycles including voltage, current, capacity, energy and temperature. The data is presented in '.mat' and '.csv' files with a well structured format sorted by temperature, test type and drive cycle.
The HPPC profile is made of pulses of 10 s equivalent to 0.5C, 1C, 2C, 4C, 6C, where C is the battery nominal capacity, each pulse followed by a rest period of 20 minutes. A total of five pulse discharge HPPC tests were performed at 100, 95, 90, 80, 70,..., 30, 25, 20, 15, 10, 5, 0 % of SOC.

A series of nine drive cycles were performed in the following order: Cycle 1, Cycle 2, Cycle 3, Cycle 4, US06, HWFET, UDDS, LA92, Neural Network (NN). Cycles 1-4 consist of random mix of US06, HWFET, UDDS, LA92, and Neural Network drive cycles (these been emission test cycles regulated by American authorities [194]).
Neural Network drive cycle consists of combination of portions of US06 and LA92 drive cycle, and was designed to have some additional dynamics which may be useful for training neural networks.

The drive cycle power profile is calculated for an electric Ford F150 truck with a 35 kWh battery pack scaled down for a single 18650PF cell. The drive cycle tests are terminated when voltage first hits 2.5 V for 25 °C.

Additional information about the dataset used in this Chapter is found in the Appendix B.

6.2.1 Battery Model Estimation

The Hybrid Pulse Power Characterization (HPPC) test dataset at 25 °C provided in [190] was utilised to estimate the parameters of a second-order $2RC$ model. A second-order model has two RC networks, $R_1 \parallel C_1$ and $R_2 \parallel C_2$, according to Fig. 6.1.
The HPPC test consists of pulses separated by a rest period that allows the battery to reach a stable state and is used to obtain the Battery OCV versus SOC curve and the circuit components' values $R_1 \parallel C_1$ and $R_2 \parallel C_2$.

6.2.2 Battery SOC Estimation

Herein, the nine drive cycles test datasets at 25 °C only, provided in [190] were used to estimate the SOC of the proposed battery both for the model-based or data-driven approaches.
Each drive cycle has the SOC computed via EKF and UKF for the model-based system. In addition, the ECM terminal voltage V_t is also computed, since it is one of the states of the space-state modelling. For the data-driven PHM calculation, the SOC is computed from the dataset without any prior modelling of the system.

For the SOC, four estimations are made using:

- Coulomb-counting method according to (6.2), used as a benchmark for the proposed methods.
- Extended Kalman Filter (EKF) method as elucidated in Section 3.2.1.
- Unscented Kalman Filter (UKF) method as elucidated in Section 3.2.2.
- Data-driven PHM, as described in Chapter 4.

6.2.3 Hardware and Software

The SOC estimator using EFK and UKF is computed using Matlab and Simulink R2023b. For the battery model estimation, the Parameter Estimator toolbox [195] is used.
The offline training for the data-driven PHM was performed on a 1xQuadro RTX8000 - 48GB GDDR6 GPU (graphical card).
The data-driven PHM SOC estimator is computed using Keras [196] with a Tensorflow-backend, with Python version 3.9.7.

6.3 Model-Based Battery Parameters Estimation

This section provides initially the battery modelling established by an optimization process on the battery ECM. The parameters obtained then will be used in the SOC estimations can be compared across the different methods.

6.3.1 Battery Parameters Estimation

The parametrization is done using the HPPC dataset at 25 °C. Initial values are assigned for the parameters, V_{oc}, R_0, R_1, τ_1, R_2, τ_2 for each $SOC = [0 : 0.1 : 1]$, being 1 equivalent to a battery 100% charged.

The HPPC dataset is then loaded into the parameter estimator tool [195]. The used software formulates the parameter estimation as an optimization problem, aiming at minimizing a sum-squared error (SSE) cost function. It is a long and laborious process, since the dataset contains over 30 hours of data. Table 6.1 shows the results for the parameters for the $SOC = [0 : 0.1 : 1]$.

The results for the battery parameters optimization is depicted in Fig. 6.2.
The first plot shows the current (A) as detailed in the previous section, that is, pulses of 10 s equivalent to 0.5C, 1C, 2C, 4C, 6C, with each pulse set apart by 20 minutes rest. A total of five pulse discharge HPPC tests were performed at each $SOC = [100\% : 5\% : 0\%]$.

Both the terminal voltage and the SOC without the parameters optimization are shown in the two subsequent plots: it can be seen that the model running with non-optimised parameters does not reach the target values.
The voltage (ECM), modelled with non-optimised parameters: R_0, R_1, τ_1, R_2, τ_2 does not fit the Voltage (Actual) profile.
Similarly, the SOC (EKF), computed according to Section 3.2.1 does not follow the SOC (Coulomb) reported by the battery intrinsec modelling.
However, when the battery ECM is loaded with the values shown in Table 6.1, both the terminal voltage and the SOC meet the accuracy required and track the target values.

Table 6.1: Battery ECM Parameters Optimization for Different SOC levels

SOC (V)	VOC (V)	R_0 (Ω)	R_1 (Ω)	τ_1 (s)	R_2 (Ω)	τ_2 (s)
0.10	2.6216	0.0999	0.0098	1.1541	0.0097	12.7400
0.15	3.9563	0.1000	0.0099	66.8010	0.0097	48.8030
0.30	3.7915	0.0353	0.0100	49.5000	0.0097	52.7700
0.40	3.8678	0.0696	0.0100	73.2770	0.0097	58.1370
0.50	3.8172	0.04304	0.0099	47.4710	0.0097	51.9410
0.60	3.8620	0.0666	0.0099	72.8030	0.0097	57.7570
0.70	3.4726	0.0272	0.0006	3.5125	0.0042	173.9300
0.80	3.7527	0.0291	0.0076	12.7400	0.0094	123.7300
0.90	4.0085	0.02918	0.0070	8.6985	0.0100	37.9880
1.00	4.1717	0.0357	0.0100	11.9480	0.0075	79.9610

6.4 Data-Driven PHM Battery SOC Estimation

This section summarizes the data-driven approach to estimate the battery SOC using the methodology proposed in Chapter 4.
The same number of measured variables are present at different datasets representing the different drive cycles, which are HPPC, Cycle 1, Cycle 2, Cycle 3, Cycle 4, US06, HWFT_a, HWFT_b, UDDS, LA92 and Neural Network (NN).
For each of these datasets, the process described below has been used.

6.4.1 Noise Handling

Initially, a selected number of spans is chosen to enrich the original dataset with values of both EWMA and EWMS. An automated approach to selecte feasible values for the span s is investigated in [128], which is a purely data-driven method to

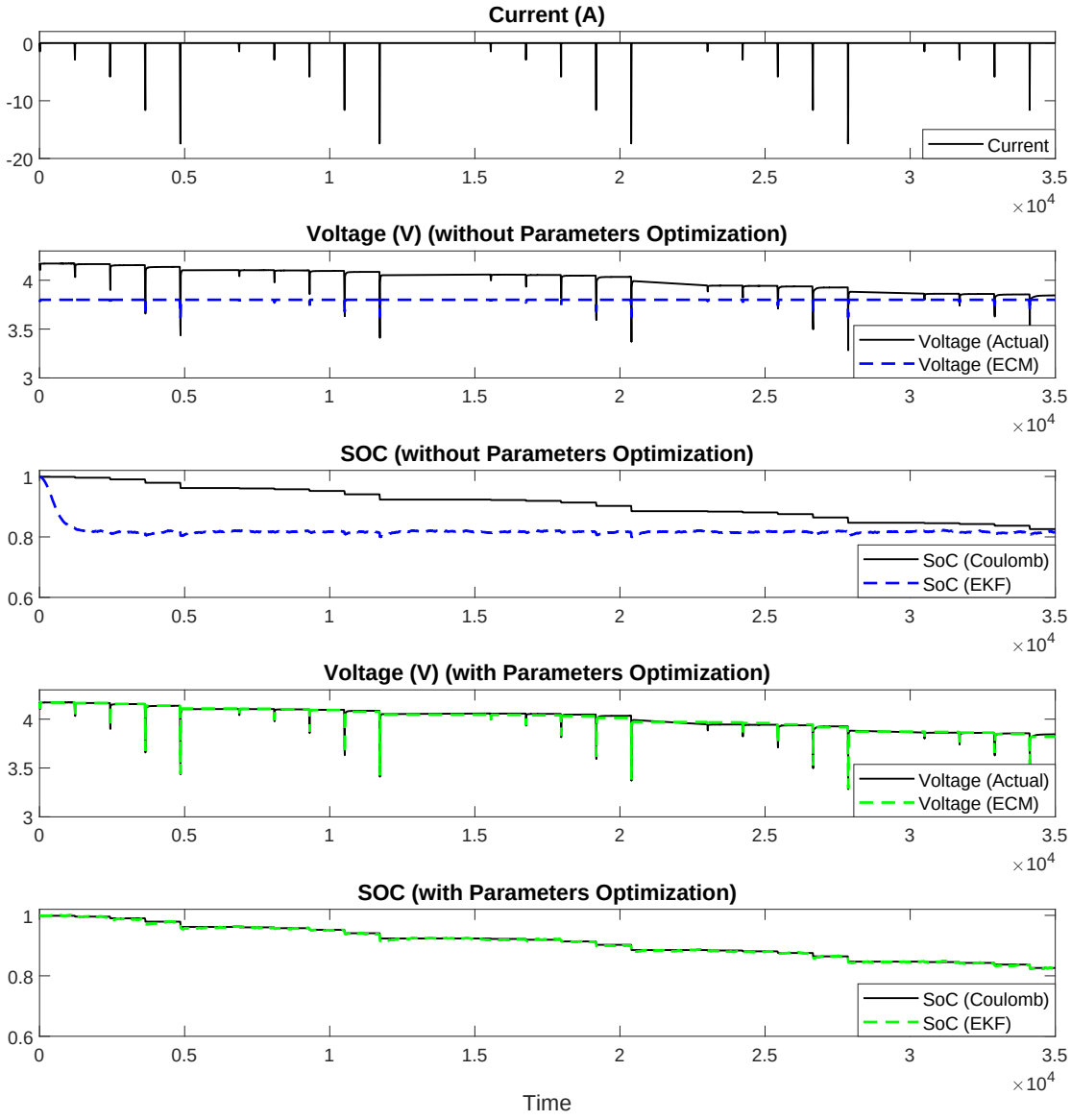

Figure 6.2: HPPC with and without Battery Parameters Optimization

establish the system dynamics. Herein we look at results in Table 6.1 to determine feasible span values, since they are available from the battery ECM parameters optimization.

We then, as in [128], explore only four possible values for the span in EWMA and EWMS calculation simultaneously in order to limit memory demand.
Ranging from a very short dynamics response to a longer time span, considering the battery ECM full charge or discharge (equivalent to a value approximately between

5τ's to 6τ's), the resulting selected span values as $s = \{10, 50, 100, 1000\}$. Applying these 4 span values for the existing dataset, made originally by 8 variables, it comes a synthetic dataset of 72 variables, considering both EWMA and EWMS.

6.4.2 Features Selection Outcome

Therefore, 72 variables including the original features and their measurements of EWMA and EWMS for the 4 selected spans are subjected to a tree SHAP algorithm as in [128] to identify which of these features' contributions are relevant. It was then noticed that out of total available features, only 10 (ten) had a clear average impact on the model output magnitude.

Fig. 6.3 shows an uniform distribution of absolute attributions across the time spans, and the most impactful features are consistently among the time, capacity and energy quantities, variables naturally needed for the SOC estimation. Nevertheless, the features attribution process was performed individually for each of the available drive cycle within this experiment. The SHAP graphic shown in Fig. 6.3 represents the UDDS drive cycle.

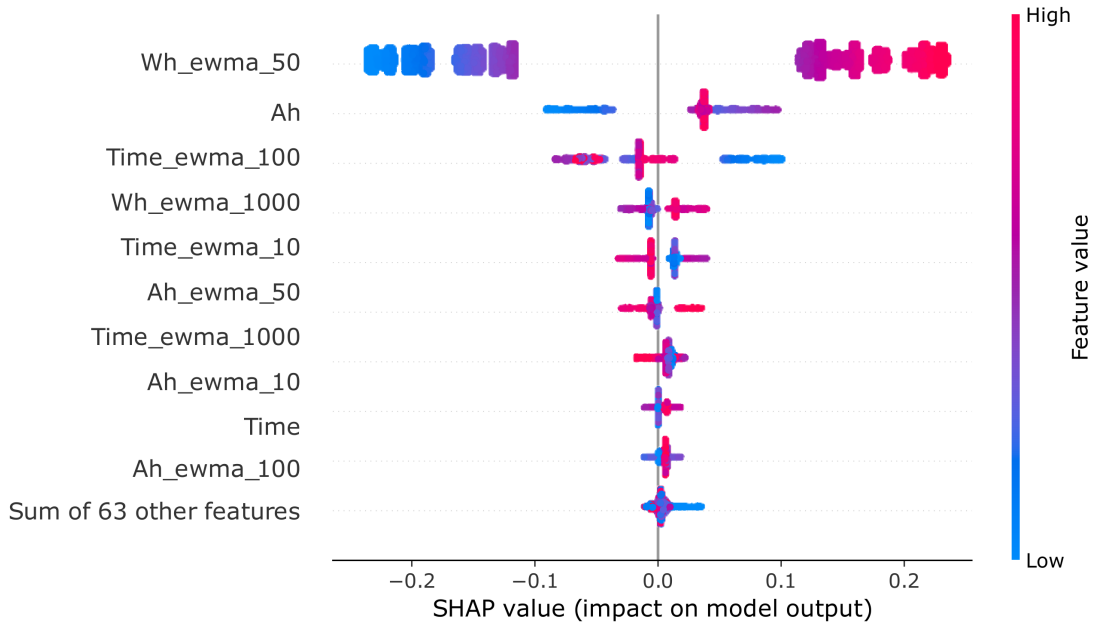

Figure 6.3: UDDS Optimal Input Attribution of the Most Impactful Features

It has also been noticed that consistently the EWMS quantities do not appear amongst the most impactful features for any drive cycles, therefore the EWMS quantities were no further considered for the battery SOC estimation.

6.4.3 The PHM Method for SOC

The same Bidi-LSTM model as presented in Chapter 4 and implemented in Chapter 5, Table 5.7 was used herein to model the data-driven PHM of the different battery's drive cycles.

6.4.4 Battery SOC Estimation

Out of the nine drive cycles tests dataset used during the data-driven PHM comparison with the state-of-the-art Kalman filters, the "HWFTa" (Highway Fuel Economy Test Cycle) and the "Cycle 3" datasets have their results presented graphically.
Two additional drive cycles, the "LA92" and the "US06" have their plots presented in Appendix B.
For each of the tested drive cycles, the main collected input time series data are the current drawn from the battery and the voltage measured across its terminals. The battery model considers the parameters optimization as shown in Table 6.1.

In the figures presented below and in Appendix B, the following structure is followed:

- The first graphic depicts the current drawn from the battery at every instant. The x-axis is shown in seconds (s), the current is given in Ampères (A).
- The second plot shows the input terminal voltage (V_t), as the actual measurement, compared with the second-order ECM voltage derived from the battery model. The x-axis is shown in seconds (s), and the y-axis shows the voltage in Volts (V)
- The third figure shows the SOC derived and computed according to Section 6.2.2: the device calculated Coulomb-counting value, the model-based Kalman Filters (EKF and UKF) and finally the proposed data-driven PHM SOC values. The SOC is an adimensional variable.

The results for the battery SOC estimation using the optimised parameters are depicted in Fig. 6.4 and Fig. 6.5.

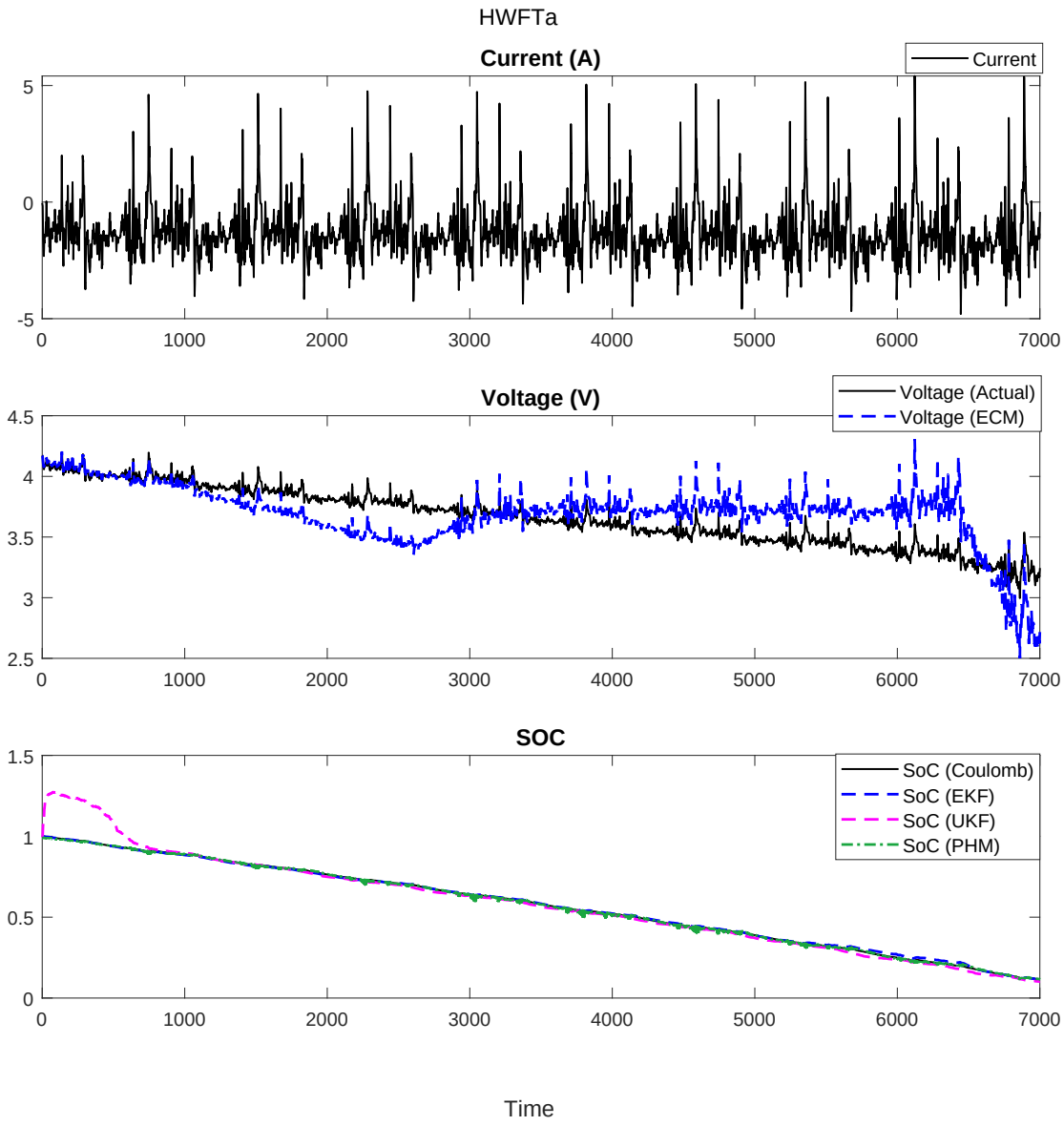

Figure 6.4: HWFTa Drive Cycle Estimation

It is seen that the battery ECM voltage is not able to perfectly match the real voltage values. Some reasons are related to the limitation of the optimization process to find suitable $R's$ and $C's$ to better represent the battery cell nonlinearity and its intrinsic behaviour. Besides other aspects such as the battery's temperature and ageing effects are difficult to model.

The graphical result for the SOC shows an overall good accuracy for all model with

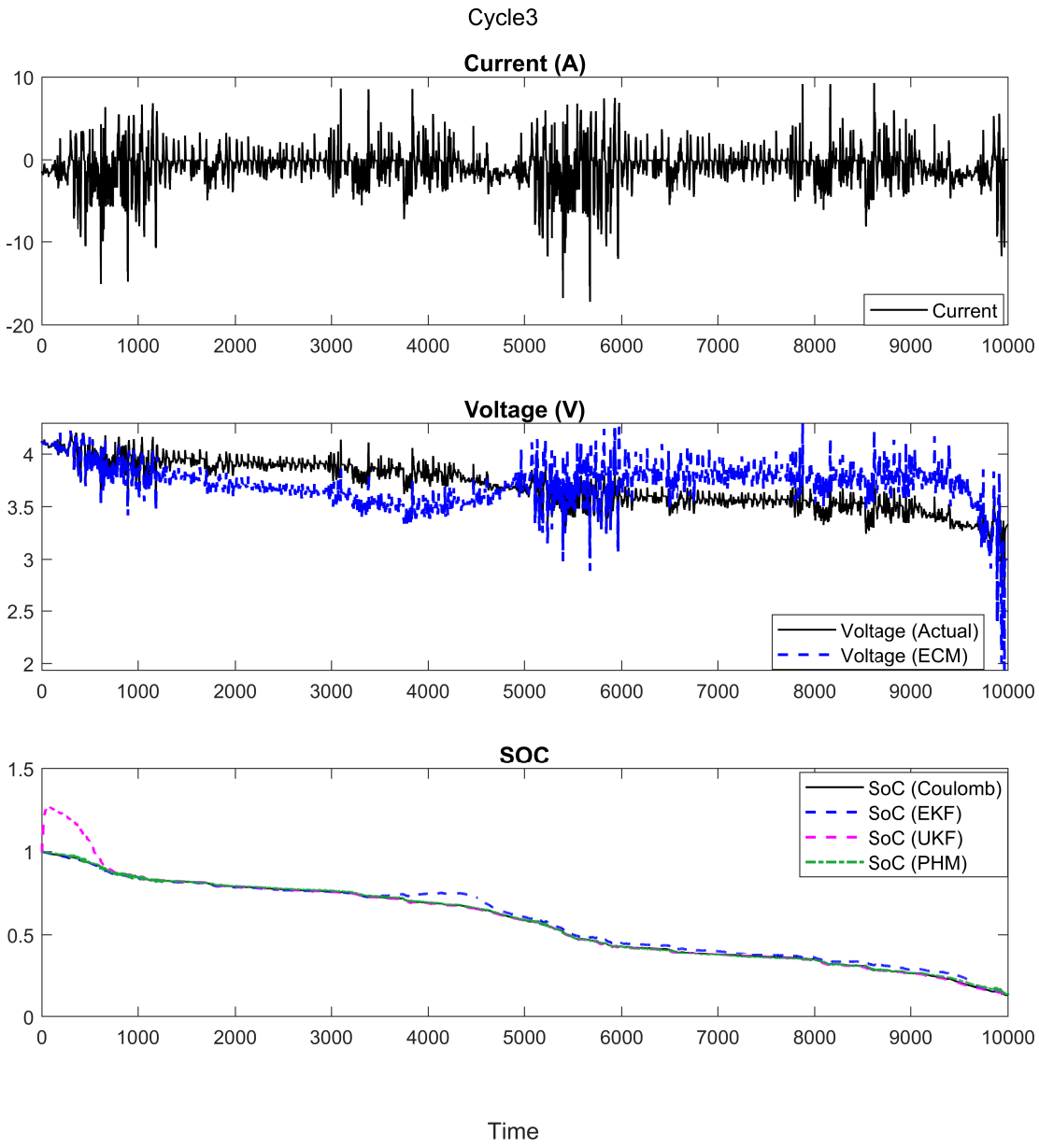

Figure 6.5: Cycle 3 Drive Cycle Estimation

regards to the Coulomb counting method, which shows low data variance for each model.
The only noticeable characteristic across the results is the overshoot seen at the SOC estimation with UKF. This behaviour during a transient period is previously reported, in case of innacurate initial SOC [127], [197].

The SOC estimation error for these two cases, HWFTa and Cycle 3, are presented in Table 6.2. The results show the maximum absolute error and the mean absolute

error (MAE) for each of the three estimators, PHM, UKF and EKF. The remaining drive cycles were also included in Table 6.2. The model significance and the full results are discussed in Section 6.4.5.

Table 6.2: SOC Estimation Error Comparison

Drive Cycles	Method	SOC Max Absolute Error	SOC Mean Absolute Error
UDDS	EKF	7.62%	0.81%
	UKF	8.28%	1.24%
	PHM	2.38%	0.08%
HWFTa	EKF	10.10%	0.66%
	UKF	13.14%	1.90%
	PHM	7.66%	0.40%
HWFTb	EKF	2.31%	0.59%
	UKF	13.12%	2.15%
	PHM	6.85%	0.69%
LA92	EKF	5.10%	0.64%
	UKF	1.80%	1.56%
	PHM	1.82%	1.04%
US06	EKF	7.55%	0.42%
	UKF	18.35%	1.84%
	PHM	6.60%	0.38%
Neural Network	EKF	5.28%	0.48%
	UKF	11.73%	1.28%
	PHM	2.44%	0.84%
Cycle 1	EKF	13.04%	0.81%
	UKF	10.97%	1.42%
	PHM	2.83%	0.42%
Cycle 2	EKF	10.23%	1.02%
	UKF	9.02%	1.29%
	PHM	6.14%	0.53%
Cycle 3	EKF	11.26%	1.58%
	UKF	10.22%	1.36%
	PHM	2.66%	0.47%
Cycle 4	EKF	7.55%	0.81%
	UKF	19.43%	1.76%
	PHM	3.13%	1.20%

The full comparison amongst the two model-based and the data-driven models is consolidated in Table 6.3 for the ten different drive cycles.

Table 6.3: Drive Cycles Estimation Comparison

Drive Cycles	Method	Correlation (corr)	Variance (var)	Bias	R^2	MSE
UDDS	EKF	0.9984	2.2e−4	8.6e−3	0.9967	3.0e−4
	UKF	0.9996	4.7e−5	3.3e−3	0.9991	5.6e−5
	PHM	0.9999	3.1e−5	8.1e−4	0.9999	3.2e−5
HWFTa	EKF	0.9997	5.7e−5	6.5e−3	0.9994	1.0e−4
	UKF	0.9997	4.2e−5	3.6e−3	0.9995	4.9e−5
	PHM	0.9998	3.2e−5	9.2e−4	0.9996	3.3e−5
HWFTb	EKF	0.9996	1.9e−3	7.6e−3	0.9992	1.9e−3
	UKF	0.9994	4.6e−5	6.4e−3	0.9989	7.5e−5
	PHM	0.9997	3.4e−5	6.9e−3	0.9994	8.2e−5
LA92	EKF	0.9994	6.3e−5	7.4e−3	0.9989	1.2e−4
	UKF	0.9995	5.6e−5	3.7e−3	0.9991	6.6e−5
	PHM	0.9999	2.4e−5	1.0e−3	0.9998	1.3e−4
US06	EKF	0.9998	1.4e−5	4.2e−3	0.9996	3.2e−5
	UKF	0.9999	5.8e−6	3.3e−3	0.9998	1.4e−5
	PHM	0.9999	6.9e−6	3.8e−3	0.9998	2.1e−5
Neural Network	EKF	0.9972	2.9e−5	6.1e−3	0.9944	6.6e−5
	UKF	0.9997	3.0e−5	3.9e−3	0.9994	4.3e−5
	PHM	0.9997	1.6e−5	1.1e−3	0.9994	5.6e−5
Cycle 1	EKF	0.9995	4.4e−5	8.1e−3	0.9990	1.1e−4
	UKF	0.9997	3.2e−5	3.9e−3	0.9994	4.3e−5
	PHM	0.9999	1.4e−5	5.7e−3	0.9998	2.0e−5
Cycle 2	EKF	0.9986	2.4e−4	1.0e−2	0.9972	3.4e−4
	UKF	0.9997	2.8e−5	3.8e−3	0.9994	3.9e−5
	PHM	0.9999	1.5e−5	4.9e−3	0.9998	1.7e−5
Cycle 3	EKF	0.9974	2.6e−4	1.6e−2	0.9948	5.1e−4
	UKF	0.9996	3.6e−5	3.9e−3	0.9992	4.8e−5
	PHM	0.9998	1.4e−5	4.7e−3	0.9996	3.6e−5
Cycle 4	EKF	0.9989	2.2e−4	8.1e−3	0.9979	2.9e−4
	UKF	0.9993	9.2e−5	6.2e−3	0.9986	1.2e−4
	PHM	0.9999	7.4e−5	1.2e−3	0.9998	2.2e−4

The criteria to assess the different methods across the drive cycles included the analysis of the correlation (corr), variance (var), the average bias, R-square (R^2) as in (4.8) and the MSE as in (5.2). As stated in the previous chapter, the lower the MSE, *var* and *bias* and the higher the R^2 the better the model fits the data.
The model significance and the full results are discussed in Section 6.4.5.

In general, the data-driven PHM method outperforms the Kalman filters by having a higher correlation with the real values, and by displaying a higher R^2 (near the unit value) and lower MSE across the different datasets.

6.4.5 Model-Based and Data-Driven Results Analysis

The results in Table 6.2 show the state-of-charge maximum absolute error (SOC MaxAE) and the SOC mean absolute error (SOC MAE) for each of the three estimators, EKF, UKF and PHM.
PHM results outperform the ones reported in the literature [101], [184], [197] with mean absolute errors (MAE) below 2% for all methods. The data-driven PHM reached 0.40% and 0.47% for the two reported drive cycles, HWFTa and Cycle 3, with the best MAE across all other methods.
The remaining drive cycles were also included in Table 6.2. Once more, the MAE for all cycles were kept below 2% outperforming the state-of-the-art found in the literature. It can also be deducted from the results that even the maximum absolute error (MaxAE) was lower than the ones computed for the EKF and UKF.

The full comparison amongst the two model-based and the data-driven models is consolidated in Table 6.3 for the ten different drive cycles.

The criteria to assess the different methods across the drive cycles included the analysis of the correlation (corr), variance (var), the average bias, R-square (R^2) as in (4.8) and the MSE as in (5.2).
In general, the data-driven PHM method outperforms the Kalman filters by having a higher correlation with the real values, and by displaying a higher R^2 (near the unit value) and lower MSE across the different datasets.
As in the previous experiment, Table 6.3 presents the quantitative evaluation of the prediction performance for the proposed model for each dataset, using five indicators, along with the test results in Table 6.2. The metrics include the MSE,

correlation, variance, bias of the estimator, and R^2 (R-squared). The objective was to minimize the variance of the regression model. A lower MSE, variance, and bias, along with a higher R^2 and correlation indicate a better fit of the model to the data.
The results from Table 6.3, when analyzed alongside Table 6.2, highlight the relative significance of each model.
For the HWFTa, for instance, depicted in Fig. 6.4, PHM algorithm stands superior results in all criteria: higher correlation and R^2 and lower MSE, variance and bias when compared with EKF and UKF in Table 6.3. This behaviour is reflected in Table 6.2, where the SOC max absolute error (MaxAE), which represents how well the model will fit across the spread of data and the SOC mean absolute error (MAE), which represents the average deviation of the model to the data, are also lower for the PHM compared to EKF and UKF. It is also worth noting that second best model is UKF and third EKF. This is an expected behaviour since UKF propagates the state variables through a nonlinear transformation, while EKF uses a first order Taylor-series expansion linearisation to propagate the state variables. Since the battery modelling is highly nonlinear, UKF provides a better state representation for the system.

This thesis compares the state-of-the-art SOC estimators based on Kalman filter variants, EKF and UKF, to the proposed data-driven PHM methodology. The latter outperforms the former ones consistently across ten different tested datasets, which reproduced different deployment scenarios and load conditions.
The proposed approach contributes with a solid framework to handle noisy measurements which are common to real-time systems limited to measurement devices capabilities. By enriching the original dataset with their weighted moving averages selected across different time spans, it was shown the SOC estimator can attain better results and eliminate inherent noise interferences.
Another important aspect of the framework is to look carefully at the relevant features in the model during the tuning phase and propose, by using additive features contributions techniques, those which are meaningful for the process. That has the benefit to reduce the computational cost of the system and to provide an insight of the system behaviour by displaying the explanatory variables more effective for the system under analysis.

The results in this thesis propose the use of an adapted bidirectional LSTM as

the most effective recursive neural network for this application, which agrees with [22], [123], [124], [125] and [177].

For the SOC estimation error comparison, both absolute and relative, the PHM framework surpasses the ones reported in the literature [101], [184], [197] with mean absolute errors (MAE) below 2% for all methods. The data-driven PHM reported maximum MAE of 1.20% and stood an average MAE of 0.60% across the different drive cycles.

It is seen in Table 6.3 that the proposed data-driven PHM framework outperforms the state-of-art methods using Kalman filters for the tested datasets, by having a higher correlation with the real values, and by displaying a higher R^2 (near the unit value) and lower MSE. For very stringent load profiles, as in the US06 and the Neural Network drive cycles, the UKF matches the PHM performance, but does not outperforms it.

6.5 Battery SOC Multi-Step Ahead Estimation

Herein, the Neural Networks (NN) drive cycle dataset at 25 °C provided in [190] is used to estimate multi-steps ahead of the SOC of the proposed methods.
The Coulomb-counting (CC) keeps tracking the Ampère-hour (Ah) and determines therefore the battery remaining capacity.
The Coulomb-counting a time consuming process and requires a high storage capacity. It is an error prone process, since if the initial value of Ampère-hour is given wrong, then all estimation tends to be incorrect. Hence, frequent calibration are often needed to prevent accumulated errors in charge integration [198].

In this section, correspondingly to the MSA forecast performed in the previous chapter, Section 5.2, the baseline for the SOC MSA forecasting, T the Coulomb-counting (6.2), is compared with three data-driven proposed methods:

- A statistical-based algorithm composed of a combination of ARIMA and polynomial regression, as explained below in subsection 6.5.1.
- A multi-step ahead Stacked-LSTM method as elucidated in Section 5.2 and [124]; and
- A multi-step ahead Bidi-LSTM, as equally described in Section 5.2 and [124].

6.5.1 ARIMA and Polynomial Regression

Based on the direct relationship between the battery instant capacity and its SOC, we propose utilising ARIMA to forecast different horizons H for the capacity, given in Ah.

Then the obtained horizon is combined with recent measurements to form the data for an estimator of a regression function that computes the SOC based on the forecasted capacity Ah.

The regression function has the form of (6.8) and it is obtained using the curve fitting tool from Simulink R2023b [199].

$$\hat{y}(k) = \beta_0 x(k) + \beta_1 + \varepsilon(k) \tag{6.8}$$

Where ε is a random error component. Additionally, β_0 and β_1 are coefficients that best fit the correlation SOC x Ah.

6.5.2 Forecast Evaluation

In the previous chapter, for the assessment of the ECU MSA forecast, the use of scaled errors metrics was proposed, to cope with the non-uniformity in the available dataset.
In this chapter though, for the evaluation of batteries SOC MSA forecast, we propose the use of both scale-dependent and percentage-based metrics, provided the uniformity of data and its dependency on the scale.
Four metrics are selected, which are the mean absolute error (MAE), the mean squared error (MSE) and its squared-root (RMSE), and the mean absolute percentage error (MAPE). The equations are as follows:
For datasets depending on the scale of their data, absolute and squared errors are used and known as scale-dependent measures. An option is using percentage-based measures to become more scale-independent. The use of scaled-independent data metrics is proposed in [89].

$$MAE = \frac{1}{n}\sum_{k=1}^{n} |e_k| \tag{6.9}$$

where n is the number of data points, y_k are the observed (or true) values and $\hat{y}_k$ the predicted ones. The scale-dependent error e_k is defined as:

$$e_k = y_k - \hat{y}_k \tag{6.10}$$

The MSE and RMSE are:

$$MSE = \frac{1}{n}\sum_{k=1}^{n} e_k^2 \tag{6.11}$$

$$RMSE = \sqrt{MSE} \tag{6.12}$$

Finally, the MAPE definition is given below:

$$MAPE = \frac{1}{n}\sum_{k=1}^{n}(\frac{|e_k|}{|y_k|}) \tag{6.13}$$

6.5.3 Battery SOC MSA Results and Analysis

This section summarizes the main results obtained in the evaluation and comparison of the different proposed methods. Initially the CC SOC is obtained directly from the application of (6.2) to the available dataset, and the data-driven methods are obtained according to the previous sections.
The prediction results are related to one single run of the battery data when the SOC was slightly above 90%. The selected dataset was the "Neural Network". For comparison, the different horizons were tested within $H = \{10, 20, 30\}$ timesteps ahead for the three different methods.

The results can be seen in Fig. 6.6. It is noticeable that both ARIMA and Bidi-LSTM have good fitting to the expected values. The stacked-LSTM, despite fitting within the close range of the expected values does not perform as the other methods when the time horizon increases. Overall, in the figure, it is seen that the ARIMA outperforms the other methods.
The quantitative analysis, using the metrics detailed in Section 6.5.2 are found in Table 6.4. All indicators show that statistical method ARIMA outperforms the machine learning ones, however the Bidi-LSTM has consistently better results than the stacked-LSTM. As the forecast window increases, the Bidi-LSTM falls under the same index scales of the ARIMA.
Despite of the best results and fitting to the proposed dataset, the statistical-based algorithm, a combination of ARIMA with polynomial regression, is based on the BMS Coulomb-counting, which as stated before, is more computationally intensive and requires frequent calibration to prevent accumulated errors in charge integration [198].

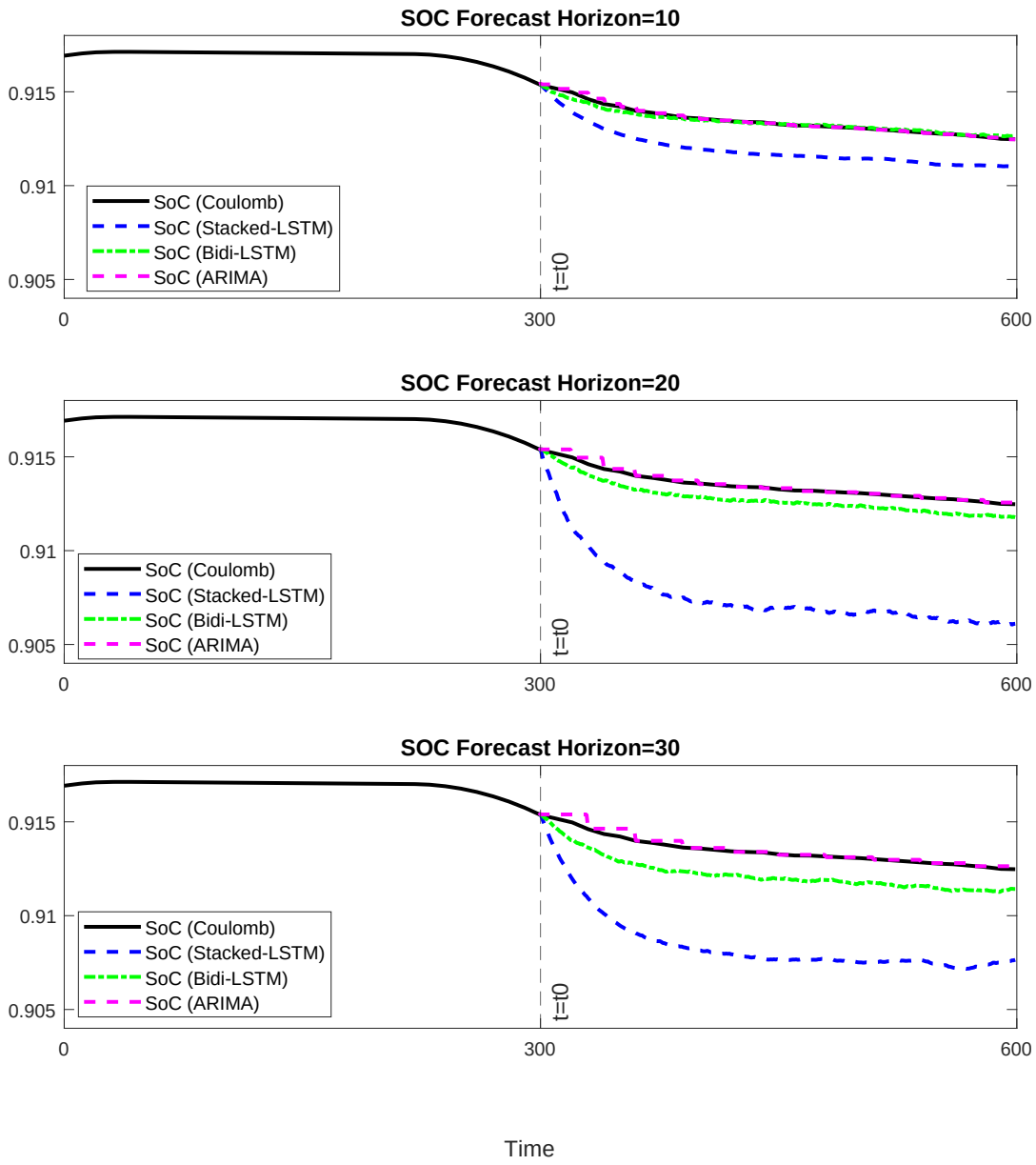

Figure 6.6: SOC Forecast for Different Horizons

Therefore, as presented in this thesis, the use of a bidirectional LSTM stands as an effective data-driven option for this application, which agrees with [124], [22], [123], [125] and [177].

Besides, the Bidi-LSTM requires to be trained only once and provides a MIMO response at each time sequence with excellent fitting to the dataset.

Table 6.4: SOC Forecast Estimation Comparison

Horizon	Method	MAE	MSE	RMSE	MAPE
10	ARIMA	1.27e−4	3.61e−8	1.90e−4	1.39e−4
	Bidi-LSTM	1.16e−3	1.76e−6	1.33e−3	1.27e−3
	Stacked-LSTM	1.94e−3	3.89e−6	1.97e−3	2.12e−4
20	ARIMA	6.62e−4	5.03e−7	7.10e−4	7.25e−4
	Bidi-LSTM	3.10e−3	1.15e−5	3.39e−3	3.41e−3
	Stacked-LSTM	5.76e−3	3.38e−5	5.81e−3	6.31e−3
30	ARIMA	1.17e−3	1.61e−6	1.27e−3	1.29e−3
	Bidi-LSTM	4.18e−3	2.14e−5	4.63e−3	4.61e−3
	Stacked-LSTM	6.77e−3	4.75e−5	6.89e−3	7.41e−3

6.5.4 Real-time Inference

The two machine-learning based methods, MSA Stacked-LSTM and Bidi-LSTM, were implemented in two hardware platforms, as described in Sections 4.3 and 5.2.5. The results for selected datasets, Neural Network and HWFTa for both MSA stacked-LSTM and Bidi-LSTM are shown in Fig. 6.7.

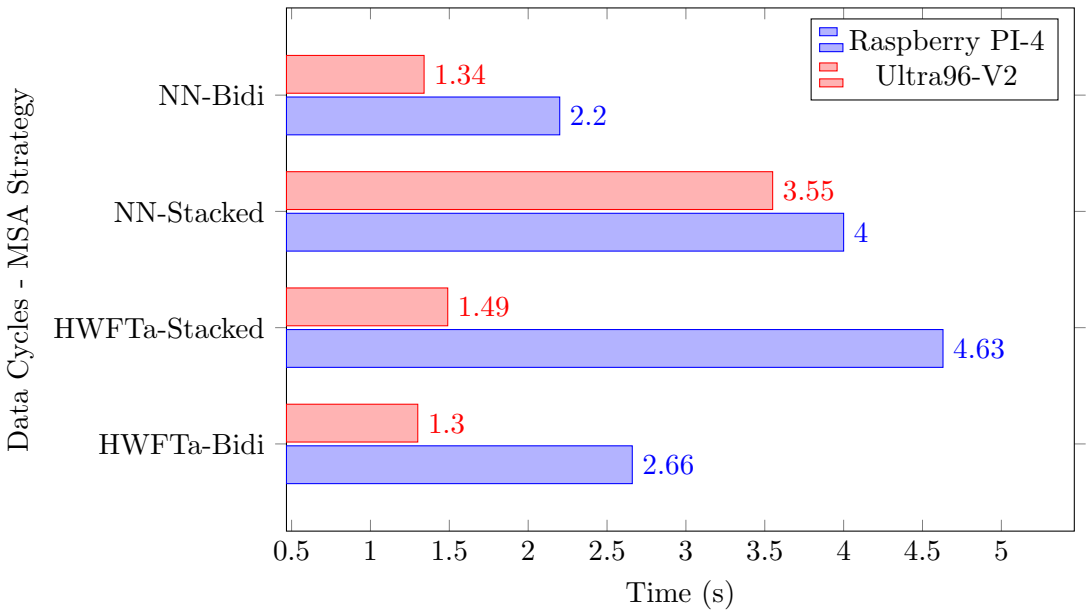

Figure 6.7: Execution times for the selected Methods.

It can be depicted by the chart that the MSA implementation using the method Bidi-LSTM is faster in all tested platforms and datasets than the Stacked-LSTM. The fastest real-time execution of the Bidi method takes around 1.3 s to run on U96 and 2.2 s on RP4.

6.6 Chapter Summary

This chapter validated the proposition in Chapter 4 and extended the results in Chapter 5. By using real battery datasets, it was possible to implement the whole PHM framework and verify the battery data-driven state-of-charge (SOC) determination, under different load conditions. Ten different datasets were implemented and tested according the novel method proposed in Chapter 4 and their results compared to the state-of-the-art Kalman Filters as proposed in Chapter 3. The results shown in Table 6.2 and Table 6.3 show that the proposed data-driven PHM methodology consistently outperforms the model-based Kalman filters implemented to determine the battery SOC.

The results obtained throughout the comparison of the proposed PHM framework with the two model-based methods were very encouraging. The data-driven PHM SOC method outperforms the state-of-the-art found in the literature for the mean absolute error (MAE) for all drive cycles tested. The PHM obtained results were kept near to the 1% threshold, which is considerably under the 2% found in the literature [101], [184], [197]. It was also shown that the results for the maximum absolute error (MaxAE) were lower than the ones computed for the EKF and UKF.

The PHM method implemented in this chapter uses the entire data pipeline proposed, including noise reduction, feature selection, hyperparameters optimisation and cross-validation. It also applies the same specialized bidirectional long short-term memory (LSTM) algorithm from the previous chapters, which is promising for generalisation purposes. These results address **[RQ1]** and **[RQ3]**, for implementation, reliability and generalisation of the proposed data-driven PHM framework.

For the multi-step ahead forecasting, an evaluation was performed, comparing one statistical-based algorithm, a combination of ARIMA with polynomial regression, with two data-driven machine-learning PHM methodologies for the determination of the multi-step ahead forecast of a battery SOC. The proposed statistical {ARIMA}-{polynomial regressor} presents the best fit the proposed horizons, however it is based on the BMS Coulomb-counting, which as stated before, is more computationally intensive and requires frequent calibration to prevent accumulated errors in charge integration [198]. Therefore, as presented in this paper, the use of a bidirectional LSTM stands as an effective data-driven option for the SOC MSA,

since it relies only on the explanatory variables, requires to be trained only once and provides a MIMO response at each time sequence with excellent fitting to the dataset. Finally, real-time requirements were also evaluated by executing the MSA methods above on the same platforms as mentioned in Chapter 5, with acceptable results for all methods.

Chapter 7

Discussions and Conclusions

7.1 Generalisation Feasibility

Modelling a process involves developing a mathematical model that accurately represents its dynamic input-output behavior. This is typically achieved by using input-output measurements and incorporating prior knowledge of the process [12]. The modelling process, when using state-space representation for instance, can be complex and challenging due to several factors, such as model complexity, order selection, parameter identification, nonlinear responses, time variability, and more. Since most real-world systems are nonlinear, certain assumptions must be made during modelling, such as linearisation or neglecting high-order derivatives for the EKF, as depicted in subsection 3.2.1. Consequently, the modeled system may not perfectly replicate the real system's behavior, as illustrated by the voltage plots using the ECM in Fig. 6.4 and Fig. 6.5.

Data-driven modelling for control and identification, also known as input-output models, offers an alternative to the often cumbersome model-based approach.
Data-driven methods, sometimes referred to as black box approaches, do not require explicit system models or intrinsic knowledge. Instead, they model the relationships between parameters directly and can handle complex, nonlinear systems. These methods are particularly effective in capturing the interactions between subsystems and the influence of exogenous parameters on system outcomes.

Furthermore, data-driven approaches are capable of modelling degradation characteristics using historical sensor data, enabling fault and failure prediction with minimal modelling. The correlations and causal relationships within the collected data can be uncovered, providing valuable insights for corrective and predictive maintenance.

However, data-driven approaches also have certain limitations. Some of these include:

- Data-driven methods rely heavily on the availability of training data. These approaches depend on historical system data (e.g., training datasets) to identify correlations, establish patterns, and analyze trends that can lead to failure predictions [13].
- A recursive neural network requires a laborious and computational intensive training process, with large amount of data required from the system-of-interest.
- The trained model is typically tailored to a specific system. Therefore, if the system dynamics, internal states, or input-output data profiles change, the model will likely need to be retrained to adapt to the new conditions [108].

One of the objectives of this thesis was to propose, implement, and evaluate the potential for generalizing a data-driven PHM methodology for mixed-criticality embedded systems.
Two different experiments were setup to investigate the architecture proposed in Chapter 4, where the three main stages were applied: offline processing, real-time inference and multi-step ahead forecasting.
There were differences between the two experiments, which provided valuable insights for assessing the generalisation of the approach. In the first experiment, the ECU dataset could be treated as either a regression or a classification problem. Continuous data readings from sensors and actuators determine the system's operational state. In this thesis, the problem was approached using a regression machine learning technique.
In both experiments, special attention was given to offline processing, which followed a similar approach for identifying useful features. This involved the SHAP feature attribution process, which was then used to generate a model with optimised hyperparameters. The resulting model is deployed for online condition monitoring and can also be used for MSA forecasting.

The key difference between the two experiments in the offline processing stage was related to noise reduction in the battery datasets, which were affected by the inherent noise in the data acquisition process. The noise reduction technique was not used for the ECU dataset since good results were obtained with the raw values from the control system. Therefore, the offline processing stage, as described in Fig. 4.1 can be applied for supervised machine learning tasks for continuous datasets' regression without major changes.

For online inference, the deployment of the model is straightforward, with no need for additional data preparation such as pre-padding or post-padding, as the model has been trained with the relevant explanatory variables. The only potential adjustment during real-time processing is the inclusion of the EWMA features, if noise reduction is required before making predictions.
For multi-step ahead forecasting, a hybrid data-driven approach is proposed, combining statistical method, an Autoregressive Integrated Moving Average (ARIMA) model for forecasting explanatory features within a defined horizon and the specialized bidirectional LSTM (Bidi-LSTM) for accurately predicting the system's outcome.

The entire methodology, as described in Chapter 4, and particularly the dataflow diagram shown in Fig. 4.1, was implemented in the two experiments. These experiments involved different applications and datasets, with minimal modifications between them. This demonstrates the potential of using the proposed framework for a wide range of applications, as long as time-series data is available to compute and generate a system-level health indicator.

7.2 Conclusions

The use of data-driven approaches for the PHM of embedded systems shows significant potential, as evidenced by both the literature review and the experiments conducted in this thesis.
Revisiting the research questions that motivated this study, several key observations can be made:

[**RQ1**]: *Can data-driven machine learning algorithms, particularly those based on recurrent neural networks, be effectively utilised and trusted for the PHM of electronic systems to predict catastrophic failures?*

This thesis proposed a system-level PHM, as defined in Table 2.2, to address an entire modular electronic system or product. Two experiments were carried out on two different systems: an electronic control unit (ECU) and a battery set, including a Battery Management System (BMS). Both experiments consist of two or more subsystems that collectively support the operation of the entire system.
The proposed system-level PHM consists of an adapted long short-term memory (LSTM) recurrent neural network (RNN) designed to identify trends, patterns, and relationships in long time-series data. The implementation is a multivariate bidirectional adapted LSTM, which can learn in both forward and reverse directions, offering higher accuracy than traditional LSTMs. It serves a dual purpose in real-time: system inference and system forecasting.
In Experiment #1, an overall accuracy of over 99.98% was achieved for the selected models in classifying the system's operational state. The model's precision score, including resets, reached 99.75% with only four inaccurate predictions. This demonstrates that the proposed data-driven PHM framework can effectively predict the operational state of the considered ECU. For multi-step ahead forecasting, used as an advisory window, an accuracy of over 79% was achieved for a 30-minute future window, providing system operators with sufficient time to act and prevent catastrophic failure.
In Experiment #2, a comprehensive comparison was made between two model-based methods — the Extended Kalman Filter and the Unscented Kalman Filter — and the proposed data-driven PHM approach to determine battery systems' state of charge (SOC). Both the model-based and data-driven methods yielded satisfactory results with the novel PHM frameword outperforming both EKF and UKF.

The PHM method achieved a mean absolute error within the 1% threshold, which is significantly lower than the 2% error found in the literature [101], [184], [197]. Similarly, a multi-step ahead forecasting evaluation was conducted to forecast the battery's SOC. For a forecast horizon of up to 30-timesteps, the bidirectional LSTM model proved to be an effective data-driven approach for SOC multi-step ahead forecasting. It relies solely on explanatory variables, requires training only once, and delivers a multi-input, multi-output (MIMO) response at each timestep with excellent fit to the dataset.

Regarding [**RQ1**], the highly positive results obtained in Experiments #1 and #2 corroborate that data-driven machine learning algorithms, particularly those utilizing specialized RNNs, can outperform state-of-the-art model-based algorithms. This finding highlights the capability of such architectures to accurately predict system behavior and play a crucial role in preventing catastrophic failures.

[**RQ2**]: *Are the results obtained from data-driven machine learning algorithms sufficiently explainable to be trusted and understood, particularly for reliability assessments and root cause analysis?*

This thesis made use of SHAP (SHapley Additive exPlanation) for feature selection, which in its turn supported the explainability of the used machine learning data-driven models, known as explainable AI (xAI) techniques. By applying additive feature attribution methods, as discussed in Chapter 4, Section 4.2.2, it was possible to establish a preliminary cause-and-effect relationship between the independent variables and the expected outcome.
In Experiment #1, a fully automated tree SHAP algorithm was used to reduce a dataset containing 47 features (independent variables) to 13 relevant ones. The algorithm identified features with a clear, average impact on the model's output magnitude. Redundant features, or those with little or no impact on the desired outcome, were excluded from the final model.
In this experiment, the electronic control unit (ECU) began exhibiting multiple faults, causing intermittent resets. Therefore, narrowing down the number of relevant features helped establishing a more precise cause-and-effect relationship between the system variables and the observed faults, aiding in the root-cause analysis.
In Experiment #2, due to the uncertainty in the raw values measurements, the noise handling technique proposed in Section 4.2.1 was used, enriching the entire dataset

to a total 72 features, considering the four different time spans proposed. The resulting dataset was then subjected to a tree SHAP algorithm with a significant reduction to ten (10) relevant features for the determination of the battery SOC for a particular data cycle.
The very positive results for the SOC estimation using the proposed strategy in Chapter 6 support the combination of noise handling and additive feature attribution methods as an effective approach for determining causality.

Regarding [**RQ2**], positive results were achieved in Experiments #1 and #2, demonstrating the ability to identify causality between inputs and outputs and helping to narrow down potential root causes for faults and failures. However, additional work is needed to formally integrate the proposed technique with suitable root-cause analysis (RCA) methods and to verify the results. This represents a promising avenue for future research in the field of explainable AI (xAI).

[**RQ3**]: *What are the most suitable machine learning techniques for the Prognostics and Health Management (PHM) of real-time embedded systems.*

As presented in Chapter 4, the multivariate bidirectional LSTM (Bidi-LSTM) is the chosen architecture in this thesis for the PHM of real-time embedded systems. As mentioned earlier, LSTM networks are highly effective when dealing with sequential data. The model replaces the traditional perceptron neuron with a memory block, allowing it to learn from long time-series data with time lags of over 1,000 discrete timesteps.

Several architectures were implemented and tested in Experiment #1 for real-time condition monitoring, according to Table 5.4: decision tree (DT), support vector regression (SVR), logistic regression (LR), long short-term memory (LSTM), artificial neural network (ANN), and two stacked ensemble models. Stack-1 uses a decision tree regressor as a base model and a linear regressor as a meta-model (or final estimator). Stack-2 uses two regressors as base models, decision tree regressor with support vector regressor, and a linear regressor stacked as a final estimator.
Additionally, several variations of LSTM were implemented for multi-step ahead estimation: Stacked-LSTM, GRU (Gated Recurrent Unit), CNN-LSTM, and Bidi-LSTM.
Each of these models had their hyperparameters optimised for the available dataset to establish a common baseline, and their performance was compared, with the

prominence of LSTM models. Furthermore, the Bidi-LSTM demonstrated the best results for multi-step ahead estimation. A multivariate bidirectional LSTM was then proposed due to its ability to learn in both forward and reverse directions, offering higher accuracy than the traditional LSTM. By using a bidirectional RNN like LSTM, the model can process input in both the forward (past) and reverse (future) directions.

In Experiment #2, the same Bidi-LSTM model presented in Chapter 4 and implemented in Chapter 5, Table 5.7, was used to model the data-driven PHM of different battery drive cycles. As mentioned above, the results were outstanding, outperforming model-based monitoring systems and achieving the best statistical results, even surpassing values reported in the state-of-the-art literature.

Therefore, for **[RQ3]**, for time-series data with long sequential dependencies, where past events are as important as more recent ones in predicting patterns, spotting trends, and identifying unexpected events, the multivariate bidirectional LSTM network was chosen as the preferred model for the experiments conducted.

Ultimately, as proposed in [4] and [124], the solutions presented here were implemented on two embedded system hardware platforms to verify their compliance with realtime constraints and resource consumption.
Both Experiments #1 and #2 were conducted on real hardware platforms to evaluate their real-time performance. The implementations on the Raspberry Pi 4 (RP4) and Ultra96-V2 (U96) platforms, both utilizing their respective ARM processors, demonstrated the feasibility of the proposed frameworks for real-time inference and forecasting applications. These platforms serve as standard testbeds, as ARM processors are the most widely used architecture for embedded systems due to their low power consumption, high performance, and cost-effectiveness.
In Experiment #1, the novel algorithm proposed in Section 5.2.6 was implemented on both the RP4 and U96 platforms for real-time multi-step forecasting of the embedded system's operational state. As shown in Fig. 5.10, four out of the five highest-ranked methods fall within the system's polling rate, with two of them achieving a refresh rate of one (1) second. This demonstrates the feasibility of implementing the algorithm on real-time embedded systems, particularly those for edge applications with limited hardware resources.
In Experiment #2, similarly, for the battery SOC multi-step ahead estimation, the fastest real-time method execution takes approximately 1.3 seconds on the U96 and

2.2 seconds on the RP4, as shown in Fig. 6.7.
Therefore, the implementation on the selected platforms demonstrated the real-time performance of the proposed framework. Systems with varying time constraints must choose their hardware accordingly to ensure that the execution time meets the system's requirements. Additionally, other platforms, such as CPUs, GPUs, and FPGAs, could be tested for further evaluation.

Finally, the novel data-driven PHM methodology introduced in Chapter 4 is designed for generalisation, and the experiments conducted in this thesis exhibit its potential for a wide range of applications. As long as time-series data is available, the methodology can be used to compute and generate real-time system health or condition monitoring indicators, provide a multi-step ahead advisory window, support root-cause analysis, and be executed on resource-constrained edge devices.

7.3 Major Findings

A summary of the major findings in this research is presented below:

- The highly favorable results from Experiments #1 and #2 confirm that data-driven machine learning algorithms, especially those leveraging specialized RNNs, can outperform state-of-the-art model-based approaches. This underscores the ability of these architectures to precisely predict system behavior and contribute significantly to preventing catastrophic failures, thereby validating research question [**RQ1**].

- Feature selection, supported by additive feature attribution methods, particularly SHAP (SHapley Additive exPlanations) in this thesis, yelded positive results in Experiments #1 and #2, effectively identifying causality between inputs and outputs and aiding in pinpointing potential root causes of faults and failures. Although further research is needed, the presented results represent a promising direction for future studies in the field of explainable AI (xAI), validating [**RQ2**],

- Bidirectional long short-term memory (Bidi-LSTM) architecture demonstrated exceptional performance in handling complex nonlinear relationships in time-series data with long sequential dependencies, where past events are as important as more recent ones in predicting patterns, identifying trends, and detecting anomalies. A multivariate bidirectional LSTM network was selected as the preferred model for the experiments conducted in this thesis, proving effective for both real-time inference and multi-step-ahead forecasting. Thus, the findings provide a positive answer to research question [**RQ3**].

- The proposed Prognostics and Health Management (PHM) framework demonstrated its effectiveness across different systems, including the electronic control unit in Chapter 5 and the battery pack in Chapter 6. The results confirm the framework's ability to generalize, showcasing its applicability to various embedded systems with minimal adjustments and successful deployment in real hardware environments.

- This thesis proposes a novel hybrid approach for multi-step-ahead forecasting in embedded systems, combining the reliability of the Autoregressive Integrated Moving Average (ARIMA) model for forecasting explanatory features within a defined horizon with the precision of the bidirectional LSTM (Bidi-LSTM) for predicting the system's operational state. The results from Experiments #1 and #2 confirm that this approach can effectively provide an advisory window for these complex systems.

- This thesis also introduces a novel approach for noise reduction in real-time embedded systems. By incorporating the exponentially weighted moving average (EWMA) and exponentially weighted moving standard deviation (EWMS) in real-time for the relevant features identified during the feature selection process, consistent results were achieved in Experiment #2 (Chapter 6). This method effectively reduces noise while preserving the original data structure, ensuring that no essential features are removed from the dataset.

- Despite its higher computational complexity compared to other tested algorithms, the long short-term memory (LSTM) architecture delivered consistent results in both reset classification and battery state-of-charge (SOC) estimation. Implementations on the Raspberry Pi 4 (RP4) and Ultra96-V2 (U96) platforms confirmed the feasibility of the proposed framework for real-time inference and forecasting applications.

7.4 Directions for Future Research

Several research directions can be derived from the work presented in this thesis. The following future areas of research are recommended:

Root-cause analysis using Explainable AI (xAI): formally integrating machine learning techniques with appropriate root-cause analysis (RCA) methods to identify causality and determine the root causes of problems. In this thesis, SHapley Additive exPlanation was used as an additive feature attribution method to identify potential root causes of faults and failures. However, other xAI methods and variants are available such as DeepLIFT and Layer-Wise Relevance Propagation, as discussed in [146], were not explored here but offer promising alternatives. These can be combined with formal RCA methods like Pareto charts, Failure Mode and Effects Analysis (FMEA), and Fault Tree Analysis for further investigation.

Safety-Critical Machine Learning Enabled Systems: the methodology proposed in this thesis demonstrated very high accuracy in determining fault conditions and outperformed state-of-the-art model-based methods. However, safety-critical systems require a conservative approach, along with extensive verification and validation (V&V) procedures. As stated in Section 2.2, embedded systems, an example of dependable systems, must comply with attributes of availability, reliability, maintainability, integrity and safety. Assurance approaches for safety-critical systems will include testing, formal verification methods, runtime verification, explainability, and others. Therefore, developing methods that address the challenges of safety-critical machine learning-enabled systems represents a promising avenue for future research.

Full Prognostics and Health Management AI-Enabled Systems: As discussed in Chapter 2, PHM not only monitors the current condition of a system but also predicts future damage progression and estimates the Remaining Useful Life (RUL). This thesis focused on condition monitoring, fault detection, performance degradation, and trend tracking, but did not address RUL determination. Future work can expand the system design to fully integrate PHM capabilities, including RUL estimation for embedded systems. RUL determination may require additional data, such as run-to-failure sensor data, and different models that combine parametric and non-parametric approaches, including survival models (lifetime), degenerate models (threshold), and/or similarity models (run-to-failure).

Appendix A

ECU PHM Supplementary Data

The complete ECU features profile, including all 47 features, for experiment #1 in Chapter 5 is presented in Table A.1 below.

Feature	Mean	std	min	25%	50%	75%	max
Feature 1	30.4	10.14	0	24.27	24.3	40.51	41.8
Feature 2	10.54	6.59	0	5.06	5.06	17.76	17.79
Feature 3	308.95	15.08	0	308	308	312	312
Feature 4	306.7	14.09	0	304	308	308	312
Feature 5	70.9	69.91	0	0	140	140	280
Feature 6	70.61	69.67	0	0	120	140	260
Feature 7	306.34	15.28	0	304	304	312	316
Feature 8	306.55	5.01	0	304	304	312	312
Feature 9	141.13	31.71	0	140	140	160	280
Feature 10	141.23	31.62	0	140	140	160	280
Feature 11	0	0	0	0	0	0	0
Feature 12	4.30e6	0.86e6	0	4.47e6	4.47e6	4.47e6	4.47e6

Feature	Mean	std	min	25%	50%	75%	max
Feature 13	3417.8	686.1	0	3555	3556	3556	3556
Feature 14	0	0	0	0	0	0	0
Feature 15	0	0	0	0	0	0	0
Feature 16	0	0	0	0	0	0	0
Feature 17	0	0	0	0	0	0	0
Feature 18	21.88	6.47	0	18	18	32	32
Feature 19	18.31	8.32	-1.2	10.73	21.43	25.73	35.28
Feature 20	18.27	6.58	10	10	20	25	28
Feature 21	16.51	3.2	13.99	14	18	18	36
Feature 22	17.55	7.08	10	10	20	24	32
Feature 23	17.55	7.14	10	10	20	24	32
Feature 24	38.43	17.78	-1.2	22.53	44.28	55.7	63.9
Feature 25	51.6	22.58	7	28	59	71	83
Feature 26	37.58	16.66	5	20	43	55	59
Feature 27	53.02	23.48	7.05	28.2	60.63	77.55	84.6
Feature 28	9	0	9	9	9	9	9
Feature 29	0	0	0	0	0	0	0
Feature 30	17.57	4.54	0	14	18	18	26
Feature 31	17.89	3.79	1.99	14	18	22	32
Feature 32	25.4	4.23	14	26	26	30	30
Feature 33	26.14	4.52	9.99	24	26	30	44
Feature 34	23.13	4.61	0	22	24	28	38
Feature 35	35.79	11.37	9.99	32	32	50	50
Feature 36	22.7	8.67	1.99	14	22	28	54
Feature 37	22.12	7.94	3.99	12	24	28	42
Feature 38	39.05	19.16	-41.68	30.98	49.53	52.4	54.01
Feature 39	36.89	17.67	-31.32	26.24	46.62	48.27	53.9
Feature 40	54.06	11.68	0	56.45	57.75	57.85	161.99
Feature 41	54.09	11.69	0	56.76	57.72	57.82	211.86

Feature	Mean	std	min	25%	50%	75%	max
Feature 42	53.85	11.71	0	56.6	57.4	57.66	300.84
Feature 43	3.54	6.68	-41.68	-3.01	6.65	10.47	12.4
Feature 44	7.47	2.08	-31.32	7.13	8.01	8.53	9.79
Feature 45	27.97	6.75	0	30	30	32	32
Feature 46	13.16	1.34	6	12	14	14	32
Feature 47	13.92	4.22	1.99	10	16	16	32
Operating Mode	4.56	1.29	0	5	5	5	5

Table A.1: ECU Dataset Profile

The 17 features selected during the feature attribution process, have their density distribution shown in Fig. A.1.

Table A.2 and Fig. A.2 represent the implementation proposed in subsection 4.4.2 and tested in the two experiments performed.

Table A.2: Multivariate Bidirectional LSTM Model Implementation

Layer	Layer Type	Output Shape	# Parameters
# 1	Bidirectional LSTM	(batch size, 30, 128)	41,984
# 2	Bidirectional LSTM	(batch size, 30, 128)	98,816
# 3	LSTM	(batch size, 64)	49,408
# 4	Dropout	(batch size, 64)	0
# 5	Dense	(batch size, 1)	65

The additional parameters for this implementation are according to the Table 5.7.

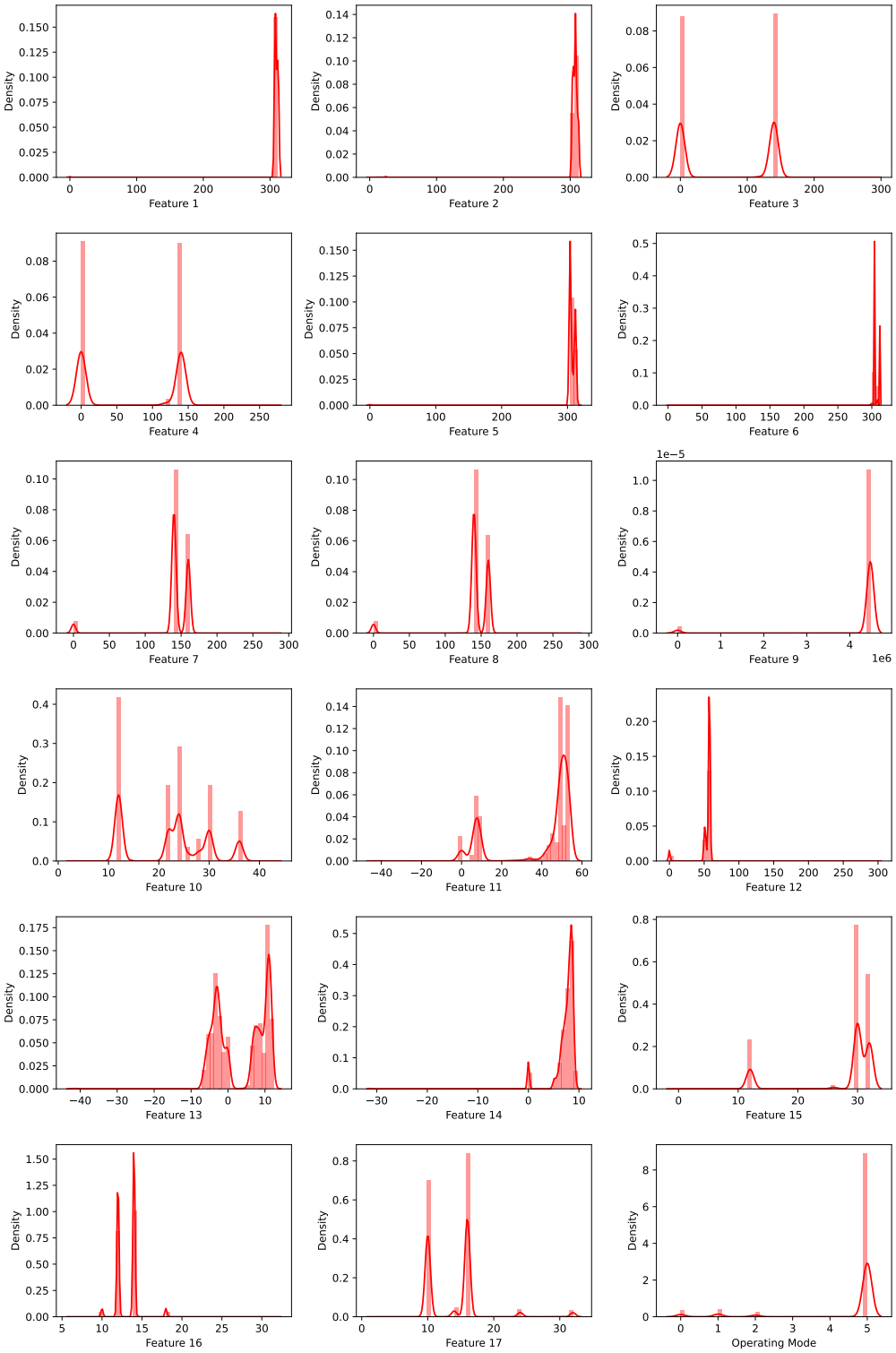

Figure A.1: Electronic Control Unit Features Distribution

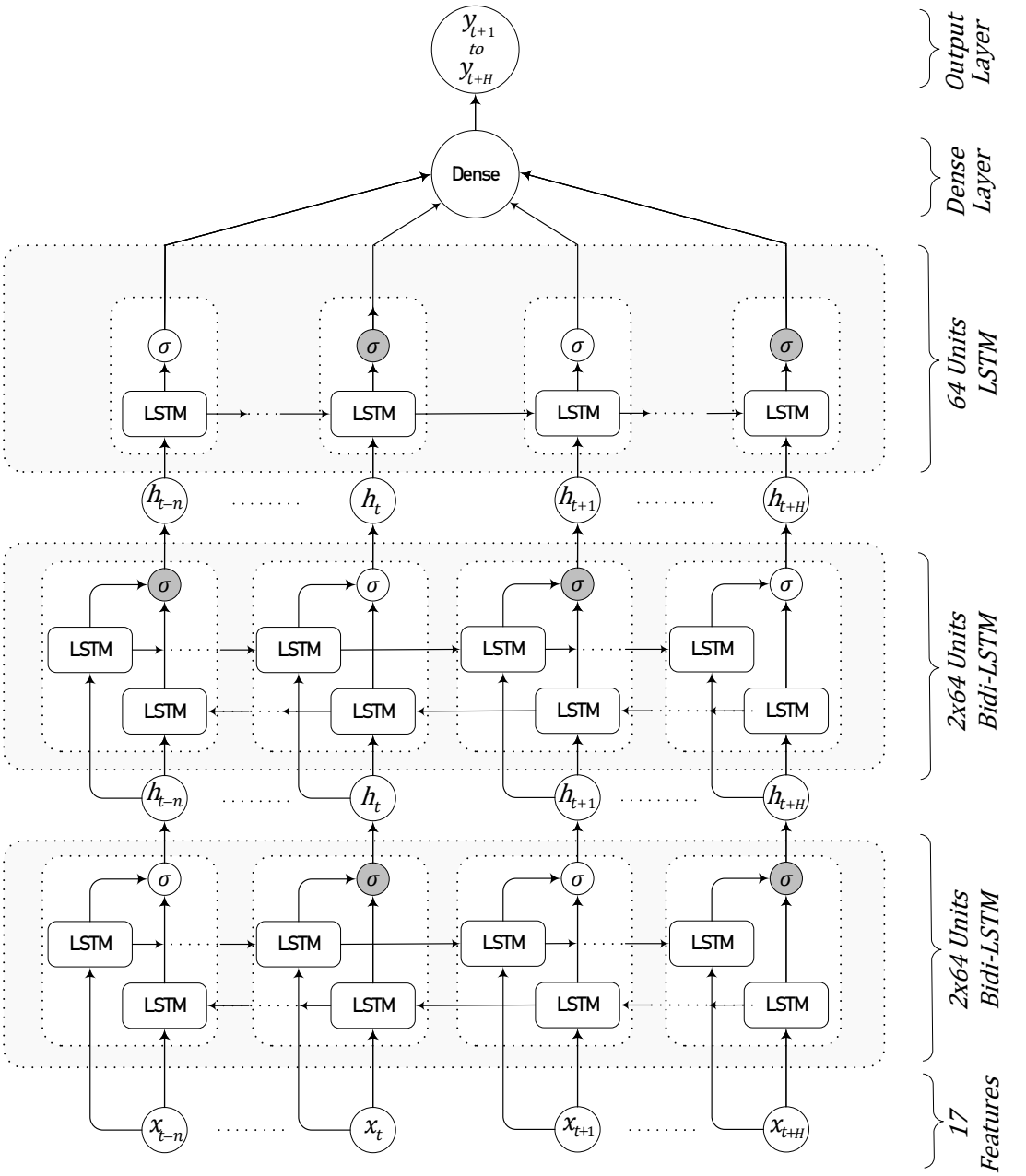

Figure A.2: MSA Bidi-LSTM Proposal with Hyperparameters Optimization

Appendix B

Battery SOC Supplementary Data

The battery features distribution for the Neural Network (NN) dataset at 25 °C is shown in Fig. B.1. The complete battery features profile for this dataset is then presented in Table B.1.

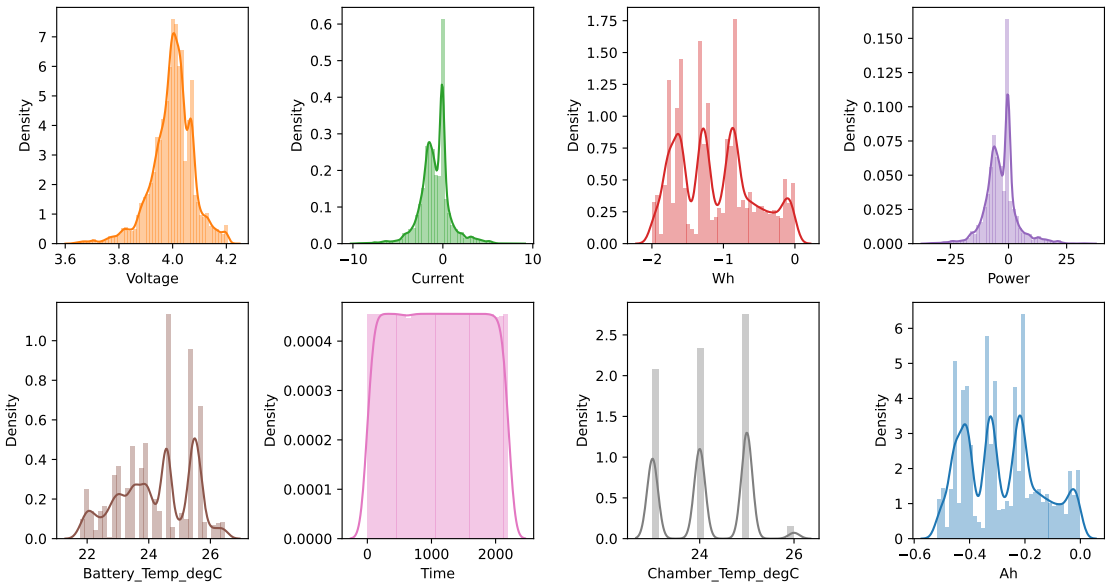

Figure B.1: Battery Features Distribution

Table B.1: Battery Dataset Profile

	Mean	**std**	**min**	**25%**	**50%**	**75%**	**max**	**Description**
Current	-0.78	2.65	-14.66	-1.59	-0.15	0.00	7.93	Current
Ah	-1.28	0.74	-2.55	-1.91	-1.24	-0.62	0.00	Capacity
Wh	-4.51	2.49	-8.64	-6.68	-4.49	-2.31	0.00	Watt-hour
Power	-2.66	9.48	-38.79	-5.67	-0.55	0.00	28.68	Power
Battery_Temp_degC	27.37	0.63	25.41	26.89	27.30	27.54	29.81	Cell Temperature
Time	5866.41	3387.53	0.00	2931.86	5865.74	8799.73	11733.23	Time
Chamber_Temp_degC	25.00	0.00	25.00	25.00	25.00	25.00	25.00	Chamber Temperature

The results for the battery SOC estimation for the drive cycles “LA92” and “US06” using the optimised parameters are presented in Fig. B.2 and Fig. B.3.

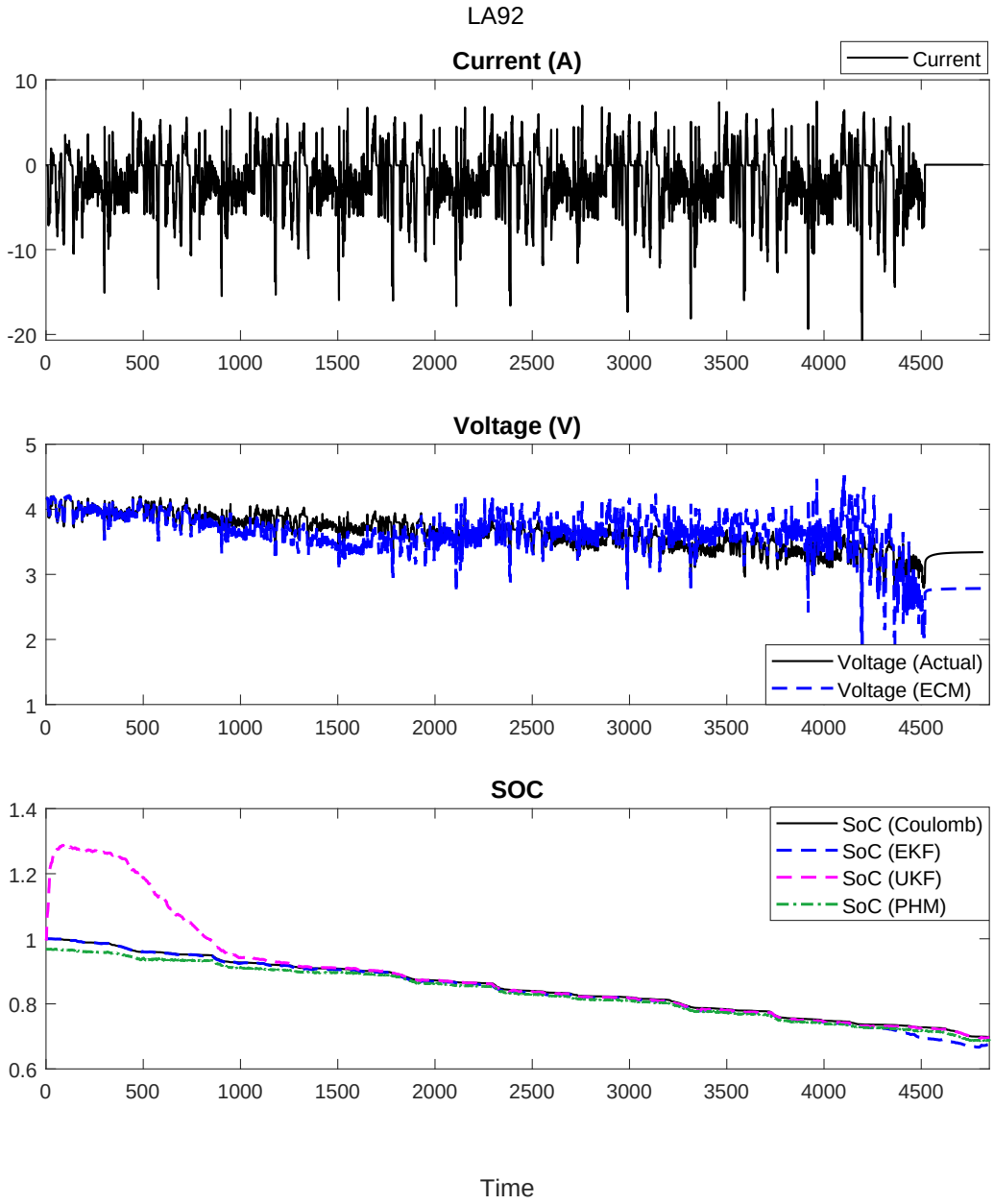

Figure B.2: LA92 Drive Cycle Estimation

The SOC estimation error for these two drive cycles, “LA92” and “US06”, are presented in Table 6.2, and the full comparison amongst the two model-based and the data-driven models is consolidated in Table 6.3.

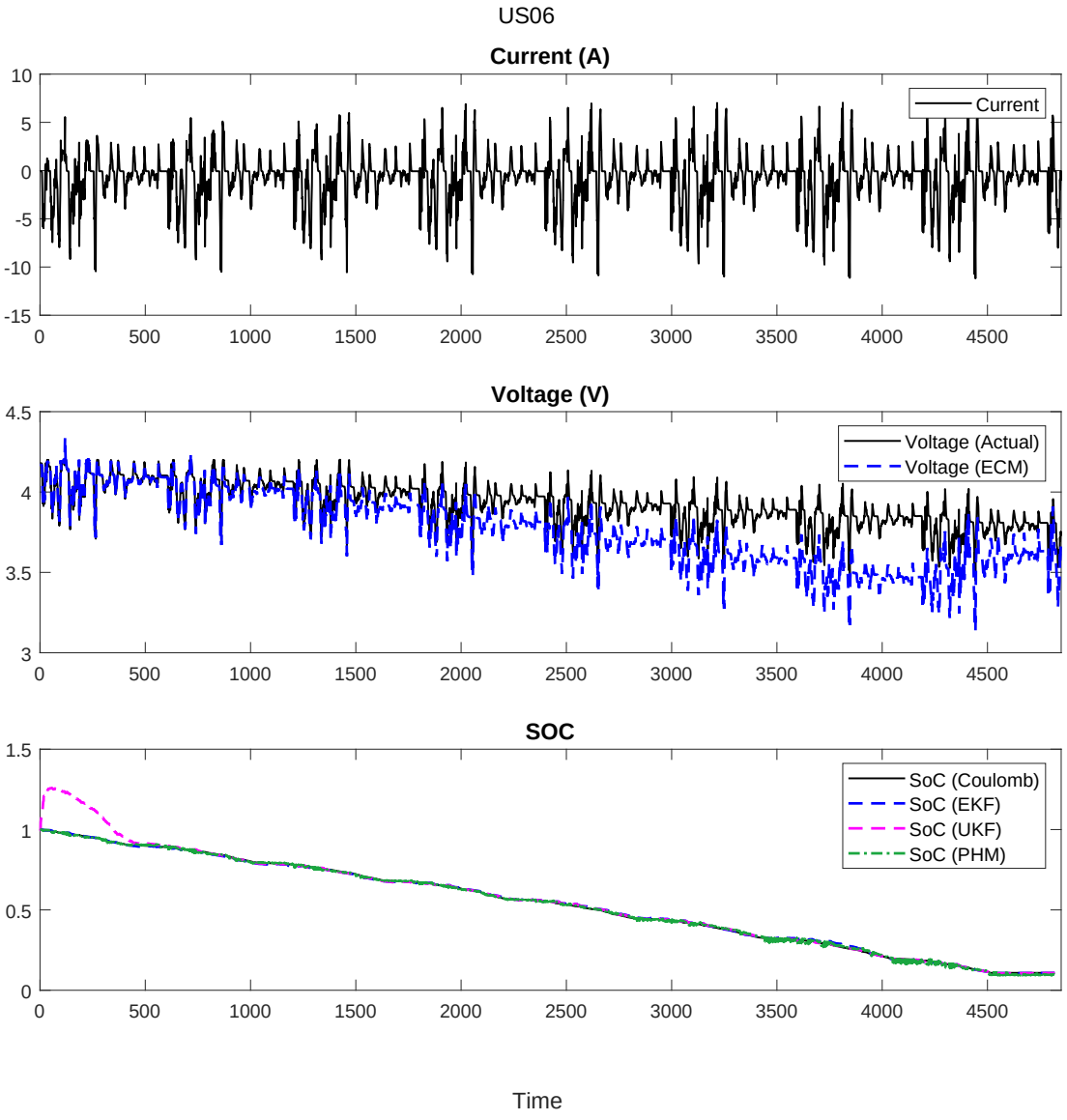

Figure B.3: US06 Drive Cycle Estimation

Bibliography

[1] D. Bhat, S. Muench, and M. Roellig, "Application of machine learning algorithms in prognostics and health monitoring of electronic systems: A review," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 4, p. 100166, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S277267112300061X

[2] A. Inamdar, W. D. van Driel, and G. Zhang, "Digital twin technology—a review and its application model for prognostics and health management of microelectronics," *Electronics*, vol. 13, no. 16, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/16/3255

[3] M. G. Pecht and M. Kang, *"A PHM Roadmap for Electronics-Rich Systems" in Prognostics and Health Management of Electronics: Fundamentals, Machine Learning, and the Internet of Things*. Wiley-IEEE Press, 2019, pp. 649–689.

[4] J. Pimentel, A. A. McEwan, and H. Q. Yu, "A comprehensive machine learning methodology for embedded systems phm," in *2023 IEEE 22nd International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2023, pp. 1954–1959.

[5] "Ieee standard framework for prognostics and health management of electronic systems." *IEEE Std 1856-2017*, pp. 1 – 31, 2017.

[6] S. H. Aldaajeh, S. Harous, and S. Alrabaee, "Fault-detection tactics for optimized embedded systems efficiency," *IEEE Access*, vol. 9, pp. 91 328–91 340, 2021.

[7] J. Henkel, J. Henkel, and N. Dutt, *Dependable embedded systems*, 1st ed., ser. Embedded systems (Springer (Firm)). Cham: Springer Nature, 2021.

[8] S. Ochella, M. Shafiee, and F. Dinmohammadi, "Artificial intelligence in prognostics and health management of engineering systems," *Engineering Applications of Artificial Intelligence*, vol. 108, p. 104552, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197621003961

[9] J. Guo, Z. Li, and M. Li, "A review on prognostics methods for engineering systems," *IEEE Transactions on Reliability*, vol. 69, no. 3, pp. 1110–1129, 2020.

[10] A. Murgia, C. Harsha, E. Tsiporkova, C. Nawghane, and B. Vandevelde, "A hybrid model for prognostic and health management of electronic devices," *Electronics*, vol. 13, no. 3, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/3/642

[11] Y. Wang, "A new concept using lstm neural networks for dynamic system identification," in *2017 American Control Conference (ACC)*, 2017, pp. 5324–5329.

[12] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*, 2nd ed. Cambridge University Press, 2022.

[13] X. Li, Q. Ding, and J.-Q. Sun, "Remaining useful life estimation in prognostics using deep convolution neural networks," *Reliability Engineering & System Safety*, vol. 172, pp. 1–11, 2018.

[14] M. Han and W. Baek, "Herti: A reinforcement learning-augmented system for efficient real-time inference on heterogeneous embedded systems," in *2021 30th International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2021, pp. 90–102.

[15] V. Nguyen, M. Kefalas, K. Yang, A. Apostolidis, M. Olhofer, S. Limmer, and T. Bäck, "A review: Prognostics and health management in automotive and aerospace," *International Journal of Prognostics and Health Management*, vol. 10, no. 2, 2019.

[16] V. Atamuradov, K. Medjaher, P. Dersin, B. Lamoureux, and N. Zerhouni, "Prognostics and health management for maintenance practitioners - review,

implementation and tools evaluation," *International Journal of Prognostics and Health Management*, vol. 8, no. 060, p. 1–31, Dec 2017.

[17] J. Pimentel, A. A. McEwan, and H. Q. Yu, "Towards a real-time smart prognostics and health management (phm) of safety critical embedded systems," in *2022 25th Euromicro Conference on Digital System Design (DSD)*, 2022, pp. 696–703.

[18] O. Fink *et al.*, "Potential, challenges and future directions for deep learning in phm applications," *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103678, 2020.

[19] C. Bhargava, P. K. Sharma, M. Senthilkumar, S. Padmanaban, V. K. Ramachandaramurthy, Z. Leonowicz, F. Blaabjerg, and M. Mitolo, "Review of health prognostics and condition monitoring of electronic components," *Ieee Access*, vol. 8, pp. 75 163–75 183, 2020.

[20] T. Zonta, C. A. Da Costa, R. da Rosa Righi, M. J. de Lima, E. S. Da Trindade, and G. P. Li, "Predictive maintenance in the industry 4.0: A systematic literature review," *Computers & Industrial Engineering*, vol. 150, p. 106889, 2020.

[21] J. Pimentel and T. Vladimirova, "Towards mpsoc enabled subsea embedded systems for fault tolerant applications," in *2019 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, 2019, pp. 1–8.

[22] R. Chandra, S. Goyal, and R. Gupta, "Evaluation of deep learning models for multi-step ahead time series prediction," *IEEE Access*, vol. 9, pp. 83 105–83 123, 2021.

[23] S. Masum, Y. Liu, and J. Chiverton, "Multi-step time series forecasting of electric load using machine learning models," in *Artificial Intelligence and Soft Computing: 17th International Conference, ICAISC 2018, Zakopane, Poland, June 3-7, 2018, Proceedings, Part I.* Berlin, Heidelberg: Springer-Verlag, 2018, p. 148–159. [Online]. Available: https://doi.org/10.1007/978-3-319-91253-0_15

[24] A. Khalid, A. Sundararajan, and A. I. Sarwat, "A multi-step predictive model to estimate li-ion state of charge for higher c-rates," in *2019 IEEE*

International Conference on Environment and Electrical Engineering and 2019 IEEE Industrial and Commercial Power Systems Europe (EEEIC/I&CPS Europe). IEEE, 2019, pp. 1–6.

[25] R. Razavi-Far, S. Chakrabarti, and M. Saif, "Multi-step-ahead prediction techniques for lithium-ion batteries condition prognosis," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2016, pp. 004 675–004 680.

[26] J. Hong, Z. Wang, W. Chen, L.-Y. Wang, and C. Qu, "Online joint-prediction of multi-forward-step battery soc using lstm neural networks and multiple linear regression for real-world electric vehicles," *Journal of Energy Storage*, vol. 30, p. 101459, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352152X20300396

[27] L. Yunpeng, H. Di, B. Junpeng, and Q. Yong, "Multi-step ahead time series forecasting for different data patterns based on lstm recurrent neural network," in *2017 14th Web Information Systems and Applications Conference (WISA)*, 2017, pp. 305–310.

[28] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off." *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 32, pp. 15 849 – 15 854, 2019.

[29] J. Pimentel and J. Arif, "Communication network optimization for subsea processing fields development," in *2019 Petroleum and Chemical Industry Conference Europe (PCIC EUROPE)*, 2019, pp. 1–8.

[30] G. J. Vachtsevanos, F. Lewis, M. Roemer, A. Hess, and B. Wu, *Intelligent fault diagnosis and prognosis for engineering systems*. Wiley Hoboken, 2006, vol. 456.

[31] C. L. Gan, "Prognostics and health management of electronics: Fundamentals, machine learning, and the internet of things: John wiley & sons ltd.(2018). pp. 731, isbn: 9781119515326 (print), 9781119515326 (online)," *Life Cycle Reliability and Safety Engineering*, vol. 9, no. 2, pp. 225–226, 2020.

[32] O. Fink, Q. Wang, M. Svensén, P. Dersin, W.-J. Lee, and M. Ducoffe, "Potential, challenges and future directions for deep learning in prognostics and health management applications," *Engineering Applications of Artificial Intelligence*, vol. 92, p. 103678, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197620301184

[33] Z. Zhao, J. Wu, T. Li, C. Sun, R. Yan, and X. Chen, "Challenges and opportunities of ai-enabled monitoring, diagnosis & prognosis: a review," *Chinese Journal of Mechanical Engineering*, vol. 34, no. 1, pp. 1–29, 2021.

[34] L. Polverino, R. Abbate, P. Manco, D. Perfetto, F. Caputo, R. Macchiaroli, and M. Caterino, "Machine learning for prognostics and health management of industrial mechanical systems and equipment: A systematic literature review," *International Journal of Engineering Business Management*, vol. 15, p. 18479790231186848, 2023. [Online]. Available: https://doi.org/10.1177/18479790231186848

[35] J. Sheppard, M. Kaufman, and T. Wilmer, "Ieee standards for prognostics and health management." *IEEE Aerospace and Electronic Systems Magazine, Aerospace and Electronic Systems Magazine, IEEE, IEEE Aerosp. Electron. Syst. Mag*, vol. 24, no. 9, pp. 34 – 41, 2009.

[36] S. Xu, C. Chen, Z. Lin, X. Zhang, J. Dai, and L. Liu, "Review and prospect of maintenance technology for traction system of high-speed train," *Transportation Safety and Environment*, vol. 3, no. 3, p. tdab017, 2021.

[37] J. Sharma, M. L. Mittal, and G. Soni, "Condition-based maintenance using machine learning and role of interpretability: a review," *International Journal of System Assurance Engineering and Management*, pp. 1–16, 2022.

[38] H. Kim, J. T. Kim, and G. Heo, "Prognostics for integrity of steam generator tubes using the general path model," *Nuclear Engineering and Technology*, vol. 50, no. 1, pp. 88–96, 2018.

[39] D. Kwon, M. R. Hodkiewicz, J. Fan, T. Shibutani, and M. G. Pecht, "Iot-based prognostics and systems health management for industrial applications," *IEEE Access*, vol. 4, pp. 3659–3670, 2016.

[40] D. An, N. H. Kim, and J.-H. Choi, "Practical options for selecting data-driven or physics-based prognostics algorithms with reviews," *Reliability Engineering & System Safety*, vol. 133, pp. 223–236, 2015.

[41] ISO, "Condition monitoring and diagnostics of machines — prognostics — part 1: General guidelines," International Organization for Standardization, Geneva, CH, Standard ISO 13381-1:2015, 2015.

[42] D. An, N. H. Kim, and J.-H. Choi, "Practical options for selecting data-driven or physics-based prognostics algorithms with reviews," *Reliability Engineering and System Safety*, vol. 133, no. C, pp. 223–236, 2015.

[43] A. L. Ellefsen *et al.*, "A comprehensive survey of prognostics and health management based on deep learning for autonomous ships." *IEEE TRANSACTIONS ON RELIABILITY*, vol. 68, no. 2, pp. 720 – 740, 2019.

[44] A. Elsheikh, S. Yacout, and M.-S. Ouali, "Bidirectional handshaking lstm for remaining useful life prediction." *Neurocomputing*, vol. 323, pp. 148 – 156, 2019.

[45] T. Sutharssan, S. Stoyanov, C. Bailey, and C. Yin, "Prognostic and health management for engineering systems: a review of the data-driven approach and algorithms," *The Journal of engineering*, vol. 2015, no. 7, pp. 215–222, 2015.

[46] J. Gu, N. Vichare, T. Tracy, and M. Pecht, "Prognostics implementation methods for electronics," in *2007 annual reliability and maintainability symposium*. IEEE, 2007, pp. 101–106.

[47] S. Kim, J.-H. Choi, and N. H. Kim, "Challenges and opportunities of system-level prognostics," *Sensors*, vol. 21, no. 22, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/22/7655

[48] Pallasch *et al.*, "Edge powered industrial control: Concept for combining cloud and automation technologies," in *2018 IEEE International Conference on Edge Computing (EDGE)*, 2018, pp. 130–134.

[49] M. Soleimani, F. Campean, and D. Neagu, "Diagnostics and prognostics for complex systems: A review of methods and challenges," *Quality and Reliability Engineering International*, vol. 37, no. 8, pp. 3746–3778, 2021.

[50] I. Crnkovic, "Component-based software engineering for embedded systems," in *Proceedings of the 27th international conference on Software engineering*, 2005, pp. 712–713.

[51] R. kamal Kaur, B. Pandey, and L. K. Singh, "Dependability analysis of safety critical systems: Issues and challenges," *Annals of nuclear energy*, vol. 120, pp. 127–154, 2018.

[52] J. Knight, "Safety critical systems: challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering. ICSE 2002*, 2002, pp. 547–550.

[53] D. J. Smith and K. G. Simpson, *Safety Critical Systems Handbook*, 3rd ed. Elsevier, 2011, p. 268.

[54] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, p. 11–33, Jan 2004.

[55] IEC, "Functional international electrotechnical vocabulary (iev) – Part 192: Dependability," International Electrotechnical Comission, Geneva, CH, Standard IEC 60050:2015, 2015.

[56] B. Mahesh *et al.*, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR)*, vol. 9, no. 1, pp. 381–386, 2020.

[57] S. Qiu, X. Cui, Z. Ping, N. Shan, Z. Li, X. Bao, and X. Xu, "Deep learning techniques in intelligent fault diagnosis and prognosis for industrial systems: A review," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1305

[58] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN computer science*, vol. 2, no. 3, p. 160, 2021.

[59] R. H. Jhaveri, A. Revathi, K. Ramana, R. Raut, and R. K. Dhanaraj, "A review on machine learning strategies for real-world engineering applications," *Mobile Information Systems*, vol. 2022, no. 1, p. 1833507, 2022.

[60] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019.

[61] K. T. P. Nguyen, K. Medjaher, and D. T. Tran, "A review of artificial intelligence methods for engineering prognostics and health management with implementation guidelines," *The Artificial intelligence review*, vol. 56, no. 4, pp. 3659–3709, 2023.

[62] H. Su and J. Lee, "Machine learning approaches for diagnostics and prognostics of industrial systems using open source data from phm data challenges: A review," *International journal of prognostics and health management*, vol. 15, no. 2, 2024.

[63] B. Rezaeianjouybari and Y. Shang, "Deep learning for prognostics and health management: State of the art, challenges, and opportunities," *Measurement*, vol. 163, p. 107929, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S026322412030467X

[64] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[65] R. Lin, Y. Yu, H. Wang, C. Che, and X. Ni, "Remaining useful life prediction in prognostics using multi-scale sequence and long short-term memory network." *Journal of Computational Science*, vol. 57, 2022.

[66] S. Fu and N. P. Avdelidis, "Prognostic and health management of critical aircraft systems and components: An overview," *Sensors*, vol. 23, no. 19, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/19/8124

[67] K. T. Nguyen and K. Medjaher, "A new dynamic predictive maintenance framework using deep learning for failure prognostics." *Reliability Engineering and System Safety*, vol. 188, pp. 251 – 262, 2019.

[68] J.-F. Toubeau *et al.*, "Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1203–1215, 2019.

[69] G. Ren, N. Yang, G. Zhang, and J. Wang, "Research on fault diagnosis based on transfer learning and fault sensitivity analysis," in *2021 Global Reliability and Prognostics and Health Management (PHM-Nanjing)*, 2021, pp. 1–4.

[70] S. Mangalathu, S.-H. Hwang, and J.-S. Jeon, "Failure mode and effects analysis of rc members based on machine-learning-based shapley additive

explanations (shap) approach," *Engineering Structures*, vol. 219, p. 110927, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141029620307513

[71] E. Plischke, G. Rabitti, and E. Borgonovo, "Computing shapley effects for sensitivity analysis." 2020.

[72] A. Farahani, B. Pourshojae, K. Rasheed, and H. R. Arabnia, "A concise review of transfer learning." 2021.

[73] Z. Wan, R. Yang, M. Huang, N. Zeng, and X. Liu, "A review on transfer learning in eeg signal analysis." *Neurocomputing*, vol. 421, pp. 1 – 14, 2021.

[74] O. Amin, B. Brown, B. Stephen, and S. McArthur, "A case-study led investigation of explainable ai (xai) to support deployment of prognostics in industry," in *Proceedings of the European Conference Of The PHM Society 2022*, 2022, pp. 9–20.

[75] D. Solís-Martín, J. Galán-Páez, and J. Borrego-Díaz, "On the soundness of xai in prognostics and health management (phm)," *Information*, vol. 14, no. 5, 2023. [Online]. Available: https://www.mdpi.com/2078-2489/14/5/256

[76] A. K. B. M. Nor *et al.*, "Explainable ai (xai) for phm of industrial asset: A state-of-the-art, prisma-compliant systematic review." 2021.

[77] S. Gawde, S. Patil, S. Kumar, P. Kamat, K. Kotecha, and S. Alfarhood, "Explainable predictive maintenance of rotating machines using lime, shap, pdp, ice," *IEEE Access*, vol. 12, pp. 29 345–29 361, 2024.

[78] N. Ahmad Kamal Mohd *et al.*, "Overview of explainable artificial intelligence for prognostic and health management," *Sensors*, vol. 21, no. 8020, p. 8020, 2021.

[79] I. E. Livieris and P. Pintelas, "A novel multi-step forecasting strategy for enhancing deep learning models' performance," *Neural Comput. Appl.*, vol. 34, no. 22, p. 19453–19470, nov 2022.

[80] S. Ben Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[81] D. Kaur, S. N. Islam, and M. A. Mahmud, "A bayesian probabilistic technique for multi-step ahead renewable generation forecasting," in *2021 IEEE 2nd International Conference on Smart Technologies for Power, Energy and Control (STPEC)*, 2021, pp. 1–6.

[82] Y. Fu, W. Hu, M. Tang, R. Yu, and B. Liu, "Multi-step ahead wind power forecasting based on recurrent neural networks," in *2018 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, 2018, pp. 217–222.

[83] P. Koukaras, N. Bezas, P. Gkaidatzis, D. Ioannidis, D. Tzovaras, and C. Tjortjis, "Introducing a novel approach in one-step ahead energy load forecasting," *Sustainable Computing: Informatics and Systems*, vol. 32, p. 100616, 2021.

[84] M. Cococcioni, E. D'Andrea, and B. Lazzerini, "24-hour-ahead forecasting of energy production in solar pv systems," in *2011 11th International Conference on Intelligent Systems Design and Applications*. IEEE, 2011, pp. 1276–1281.

[85] S. Ben Taieb, A. Sorjamaa, and G. Bontempi, "Multiple-output modeling for multi-step-ahead time series forecasting," *Neurocomputing*, vol. 73, no. 10, pp. 1950–1957, 2010, subspace Learning / Selected papers from the European Symposium on Time Series Prediction. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231210001013

[86] G. Bontempi, "Long term time series prediction with multi-input multi-output local learning," *Proc. 2nd ESTSP*, pp. 145–154, 2008.

[87] G. Chevillon, "Multistep forecasting in the presence of location shifts," *International Journal of Forecasting*, vol. 32, no. 1, pp. 121–137, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169207015000801

[88] I. Fox, L. Ang, M. Jaiswal, R. Pop-Busui, and J. Wiens, "Deep multi-output forecasting: Learning to accurately predict blood glucose trajectories," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1387–1395. [Online]. Available: https://doi.org/10.1145/3219819.3220102

[89] E. Strøm, "Evaluation of multi-step forecasting models - an empirical deep learning study," NTNU Open Access, MSc Thesis, Aug. 21, 2021. [Online]. Available: https://hdl.handle.net/11250/2831558

[90] Y. Wang, S. Zhu, and C. Li, "Research on multistep time series prediction based on lstm," in *2019 3rd International Conference on Electronic Information Technology and Computer Engineering (EITCE)*. IEEE, 2019, pp. 1155–1159.

[91] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, "A survey on anomaly detection for technical systems using lstm networks," *Computers in Industry*, vol. 131, p. 103498, 2021.

[92] Y. Zhao *et al.*, "Multi-step ahead forecasting for electric power load using an ensemble model," *Expert Systems with Applications*, vol. 211, p. 118649, 2023.

[93] Z. Deng, B. Wang, Y. Xu, T. Xu, C. Liu, and Z. Zhu, "Multi-scale convolutional neural network with time-cognition for multi-step short-term load forecasting," *IEEE Access*, vol. 7, pp. 88 058–88 071, 2019.

[94] N. K. ChikkaKrishna, P. Rachakonda, and T. Tallam, "Short - term traffic prediction using fb-prophet and neural-prophet," in *2022 IEEE Delhi Section Conference (DELCON)*, 2022, pp. 1–4.

[95] K. Ogata, *Modern control engineering*, 5th ed. Boston, [Mass.] ;: Pearson, 2010.

[96] A. Fekih, H. Xu, and F. N. Chowdhury, "Neural networks based system identification techniques for model based fault detection of nonlinear systems," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 5, pp. 1073–1085, 2007.

[97] M. Baptista, M. Mishra, E. Henriques, and H. Prendinger, "Using explainable artificial intelligence to interpret remaininguseful life estimation with gated recurrent unit," 2020.

[98] C. Campestrini, "Practical feasibility of kalman filters for the state estimation of lithium-ion batteries," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:146099614

[99] G. L. Plett, "Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 1. background," *Journal of Power Sources*, vol. 134, no. 2, pp. 252–261, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775304003593

[100] ——, "Sigma-point kalman filtering for battery management systems of lipb-based hev battery packs: Part 1: Introduction and state estimation," *Journal of Power Sources*, vol. 161, no. 2, pp. 1356–1368, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378775306011414

[101] X. Wu, X. Li, and J. Du, "State of charge estimation of lithium-ion batteries over wide temperature range using unscented kalman filter," *IEEE Access*, vol. 6, pp. 41 993–42 003, 2018.

[102] E. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[103] G. A. Terejanu, "Unscented kalman filter tutorial," *University at Buffalo, Buffalo*, 2011.

[104] W. Liu and T. Yairi, "Online fault detection for industrial processes through kalman filter," in *PHM Society Asia-Pacific Conference*, vol. 4, no. 1, 2023.

[105] Q. Han, X. Shen, B. Wu, R. Zhu, D. Wang, and B. Ruan, "Fault diagnose of the rolling bearings vibration signals based on kalman filter method," in *2021 Global Reliability and Prognostics and Health Management (PHM-Nanjing)*, 2021, pp. 1–6.

[106] G. Li, J. Wei, J. He, H. Yang, and F. Meng, "Implicit kalman filtering method for remaining useful life prediction of rolling bearing with adaptive detection of degradation stage transition point," *Reliability Engineering & System Safety*, vol. 235, p. 109269, 2023.

[107] P. Lall, J. Wei, and K. Goebel, "Comparison of lalman-filter and extended kalman-filter for prognostics health management of electronics," in *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, 2012, pp. 1281–1291.

[108] K. Niu, M. Zhou, C. T. Abdallah, and M. Hayajneh, "Deep transfer learning for system identification using long short-term memory neural networks," 2022.

[109] J. Gonzalez and W. Yu, "Non-linear system modeling using lstm neural networks," *IFAC-PapersOnLine*, vol. 51, no. 13, pp. 485–489, 2018, 2nd IFAC Conference on Modelling, Identification and Control of Nonlinear Systems MICNON 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318310814

[110] J. Benitez, J. Castro, and I. Requena, "Are artificial neural networks black boxes?" *IEEE transactions on neural networks*, vol. 8, no. 5, pp. 1156–1164, 1997.

[111] G. Cybenko, "Approximation by superpositions of a sigmoidal function." *Math. Control. Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989. [Online]. Available: http://dblp.uni-trier.de/db/journals/mcss/mcss2.html#Cybenko89

[112] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[113] F. Voigtlaender, "The universal approximation theorem for complex-valued neural networks," 2022.

[114] A. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930–945, 1993.

[115] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," 2017. [Online]. Available: https://arxiv.org/abs/1709.02540

[116] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," 2014.

[117] A. Graves, N. Jaitly, and A.-r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, 2013, pp. 273–278.

[118] R. C. Staudemeyer and E. R. Morris, "Understanding lstm – a tutorial into long short-term memory recurrent neural networks," 2019.

[119] J. Li, J.-h. Cheng, J.-y. Shi, and F. Huang, "Brief introduction of back propagation (bp) neural network algorithm and its improvement," in *Advances in Computer Science and Information Engineering*, D. Jin and S. Lin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 553–558.

[120] R. Rojas, *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

[121] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[122] E. López, C. Valle, H. Allende, E. Gil, and H. Madsen, "Wind power forecasting based on echo state networks and long short-term memory," *Energies*, vol. 11, no. 3, 2018. [Online]. Available: https://www.mdpi.com/1996-1073/11/3/526

[123] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Deep bidirectional and unidirectional lstm recurrent neural network for network-wide traffic speed prediction," 2019.

[124] J. Pimentel, A. A. McEwan, and H. Q. Yu, "A novel real-time framework for embedded systems health monitoring," in *2023 26th Euromicro Conference on Digital System Design (DSD)*, 2023, pp. 309–316.

[125] C. W. Hong, C. Lee, K. Lee, M.-S. Ko, D. E. Kim, and K. Hur, "Remaining useful life prognosis for turbofan engine using explainable deep neural networks with dimensionality reduction," *Sensors*, vol. 20, no. 22, 2020.

[126] R. Zhao, R. Yan, J. Wang, and K. Mao, "Learning to monitor machine health with convolutional bi-directional lstm networks," *Sensors*, vol. 17, no. 2, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/2/273

[127] D. N. T. How, M. A. Hannan, M. S. Hossain Lipu, and P. J. Ker, "State of charge estimation for lithium-ion batteries using model-based and data-driven methods: A review," *IEEE Access*, vol. 7, pp. 136 116–136 136, 2019.

[128] W. Kirchgässner, O. Wallscheid, and J. Böcker, "Empirical evaluation of exponentially weighted moving averages for simple linear thermal modeling of permanent magnet synchronous machines," in *2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, 2019, pp. 318–323.

[129] B. Rozemberczki, L. Watson, P. Bayer, H.-T. Yang, O. Kiss, S. Nilsson, and R. Sarkar, "The shapley value in machine learning." 2022.

[130] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*. Springer, 2019, pp. 3–33.

[131] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *International conference on machine learning*. PMLR, 2013, pp. 115–123.

[132] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection." *Statistics Surveys*, vol. 4, pp. 40–79, 2010.

[133] M. Hassan, "Heterogeneous mpsocs for mixed criticality systems: Challenges and opportunities," *arXiv preprint arXiv:1706.07429*, 2017.

[134] G. Bill, W. Him, and B. Elly, "Benchmarking machine learning algorithms: A framework for evaluating efficiency in ai applications," 12 2024.

[135] L. Biikes, E. Matthew, D. White, and B. Elly, "Real-time data processing: Enhancing machine learning algorithm efficiency for streaming applications," 12 2024.

[136] K. Shan, S. Wang, L. Xiao, and Y. Sun, "Sensitivity and uncertainty analysis of measurements in outdoor airflow control strategies," *HVAC Research*, vol. 19, 05 2013.

[137] B. Porr, S. Daryanavard, L. M. Bohollo, H. Cowan, and R. Dahiya, "Real-time noise cancellation with deep learning," *Plos one*, vol. 17, no. 11, p. e0277974, 2022.

[138] S. Zhang, Y. Wan, J. Ding, and Y. Da, "State of charge (soc) estimation based on extended exponential weighted moving average h-infinite filtering," *Energies*, vol. 14, no. 6, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/6/1655

[139] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[140] W. Kirchgässner, O. Wallscheid, and J. Böcker, "Estimating electric motor temperatures with deep residual machine learning," *IEEE Transactions on Power Electronics*, vol. 36, no. 7, pp. 7480–7488, 2021.

[141] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," 2014. [Online]. Available: https://arxiv.org/abs/1403.2877

[142] K. Daniel, H. Jessica, and B. Enrico, "A survey of domain knowledge elicitation in applied machine learning." *Multimodal Technologies and Interaction*, vol. 5, no. 73, p. 73, 2021.

[143] C.-M. Forke and M. Tropmann-Frick, "Feature engineering techniques and spatio-temporal data processing." *Datenbank-Spektrum*, vol. 21, no. 3, p. 237, 2021.

[144] P. Mellodge, "Chapter 1 - introduction: What is a dynamical system?" in *A Practical Approach to Dynamical Systems for Engineers*, P. Mellodge, Ed. Woodhead Publishing, 2016, pp. 1–16.

[145] S. Khan, S. Tsutsumi, T. Yairi, and S. Nakasuka, "Robustness of ai-based prognostic and systems health management," *Annual Reviews in Control*, vol. 51, pp. 130–152, 2021.

[146] S. M. Lundberg and S. Lee, "A unified approach to interpreting model predictions," *CoRR*, vol. abs/1705.07874, 2017. [Online]. Available: http://arxiv.org/abs/1705.07874

[147] L. Merrick and A. Taly, "The explanation game: Explaining machine learning models with cooperative game theory," *CoRR*, vol. abs/1909.08128, 2019. [Online]. Available: http://arxiv.org/abs/1909.08128

[148] D. Fryer, I. Strumke, and H. Nguyen, "Shapley values for feature selection: The good, the bad, and the axioms." *IEEE ACCESS*, vol. 9, pp. 144 352 – 144 360, 2021.

[149] L. Thi-Thu-Huong, K. Haeyoung, K. Hyoeun, and K. Howon, "Classification and explanation for intrusion detection system based on ensemble trees and shap method." *Sensors*, vol. 22, no. 1154, p. 1154, 2022.

[150] S. M. Lundberg, G. G. Erion, and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles." 2018.

[151] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H. Lei, and S.-H. Deng, "Hyperparameter optimization for machine learning models based on bayesian optimization," *Journal of Electronic Science and Technology*, vol. 17, no. 1, pp. 26–40, 2019.

[152] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.

[153] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," *arXiv preprint arXiv:2003.05689*, 2020.

[154] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," 2019. [Online]. Available: https://arxiv.org/abs/1902.07638

[155] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of machine learning research*, vol. 13, no. 2, 2012.

[156] I. Tougui, A. Jilbab, and J. E. Mhamdi, "Impact of the choice of cross-validation techniques on the results of machine learning-based diagnostic applications." *Healthcare informatics research*, vol. 27, no. 3, pp. 189 – 199, 2021.

[157] J. Liu *et al.*, "Performance analysis and characterization of training deep learning models on mobile device." *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 506 – 515, 2019.

[158] M. Capra *et al.*, "Hardware and software optimizations for accelerating deep neural networks: Survey of current trends, challenges, and the road ahead." *IEEE Access, Access, IEEE*, vol. 8, pp. 225 134 – 225 180, 2020.

[159] L. Li, G. Wang, G. Wu, and Q. Zhang, "An experimental evaluation of extreme learning machines on several hardware devices," *Neural Computing and Applications*, vol. 32, no. 18, pp. 14 385–14 397, 2020.

[160] J. Pimentel, A. McEwan, and Y. Hong, "Multi-step ahead battery soc estimation using data-driven prognostics and health management," *13th International Conference on Software and Information Engineering (ICSIE 2024)*, 2024.

[161] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482–493, 2014.

[162] S. Siami-Namini, N. Tavakoli, and A. Siami Namin, "A comparison of arima and lstm in forecasting time series," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 1394–1401.

[163] C.-H. Lu and K.-T. Lai, "Dynamic offloading on a hybrid edge-cloud architecture for multiobject tracking," *IEEE systems journal*, vol. 16, no. 4, pp. 1–11, 2022.

[164] A. Javed, J. Robert, K. Heljanko, and K. Främling, "Iotef: A federated edge-cloud architecture for fault-tolerant iot applications," *Journal of grid computing*, vol. 18, no. 1, pp. 57–80, 2020.

[165] Z. Yang, B. Liang, and W. Ji, "An intelligent end-edge-cloud architecture for visual iot-assisted healthcare systems," *IEEE internet of things journal*, vol. 8, no. 23, pp. 16 779–16 786, 2021.

[166] C. Yang, S. Lan, L. Wang, W. Shen, and G. G. Q. Huang, "Big data driven edge-cloud collaboration architecture for cloud manufacturing: A software defined perspective," *IEEE access*, vol. 8, pp. 45 938–45 950, 2020.

[167] T. Dash, S. Chitlangia, A. Ahuja, and A. Srinivasan, "Incorporating domain knowledge into deep neural networks." 2021.

[168] B. Pavlyshenko, "Using stacking approaches for machine learning models," in *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, 2018, pp. 255–258.

[169] F. Schwenker, "Ensemble methods, foundations and algorithms," *IEEE Computational Intelligence Magazine*, vol. 8, no. 1, pp. 77–79, 2013.

[170] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[171] R. J. Hyndman and A. B. Koehler, "Another look at measures of forecast accuracy," *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006.

[172] C. Chen, J. Twycross, and J. M. Garibaldi, "A new accuracy measure based on bounded relative error for time series forecasting," *PLOS ONE*, vol. 12, no. 3, pp. 1–23, 03 2017.

[173] H. Hewamalage, K. Ackermann, and C. Bergmeir, "Forecast evaluation for data scientists: Common pitfalls and best practices," 2022. [Online]. Available: https://arxiv.org/abs/2203.10716

[174] T.-Y. Kim and S.-B. Cho, "Web traffic anomaly detection using c-lstm neural networks," *Expert Systems with Applications*, vol. 106, pp. 66–76, 2018.

[175] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[176] N. H. An and D. T. Anh, "Comparison of strategies for multi-step-ahead prediction of time series using neural network," in *2015 International Conference on Advanced Computing and Applications (ACOMP)*, 2015, pp. 142–149.

[177] Z. Cui *et al.*, "Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values," *Transportation Research Part C: Emerging Technologies*, vol. 118, p. 102674, 2020.

[178] J. Zhao, X. Feng, Q. Pang, J. Wang, Y. Lian, M. Ouyang, and A. F. Burke, "Battery prognostics and health management from a machine learning perspective," *Journal of Power Sources*, vol. 581, p. 233474, Oct. 2023.

[179] M. A. Hannan, D. N. T. How, M. S. H. Lipu, M. Mansor, P. J. Ker, Z. Y. Dong, K. S. M. Sahari, S. K. Tiong, K. M. Muttaqi, T. M. I. Mahlia, and F. Blaabjerg, "Deep learning approach towards accurate state of charge

estimation for lithium-ion batteries using self-supervised transformer model," *Scientific reports*, vol. 11, no. 1, pp. 19 541–19 541, 2021.

[180] X. Liu, X. Fan, L. Wang, and J. Wu, "State of charge estimation for power battery base on improved particle filter," *World Electric Vehicle Journal*, vol. 14, no. 1, 2023. [Online]. Available: https://www.mdpi.com/2032-6653/14/1/8

[181] C. Chen, R. Xiong, and W. Shen, "A lithium-ion battery-in-the-loop approach to test and validate multiscale dual h infinity filters for state-of-charge and capacity estimation," *IEEE Transactions on Power Electronics*, vol. 33, no. 1, pp. 332–342, 2018.

[182] Q. Chen, J. Jiang, H. Ruan, and C. Zhang, "Simply designed and universal sliding mode observer for the soc estimation of lithium-ion batteries," *IET Power Electronics*, vol. 10, no. 6, pp. 697–705, 2017. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-pel.2016.0095

[183] B. Ning, J. Xu, B. Cao, B. Wang, and G. Xu, "A sliding mode observer soc estimation method based on parameter adaptive battery model," *Energy Procedia*, vol. 88, pp. 619–626, 2016, cUE 2015 - Applied Energy Symposium and Summit 2015: Low carbon cities and urban energy systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610216301527

[184] Y. Chen, D. Huang, Q. Zhu, W. Liu, C. Liu, and N. Xiong, "A new state of charge estimation algorithm for lithium-ion batteries based on the fractional unscented kalman filter," *Energies*, vol. 10, no. 9, 2017. [Online]. Available: https://www.mdpi.com/1996-1073/10/9/1313

[185] F. Yang, X. Song, F. Xu, and K.-L. Tsui, "State-of-charge estimation of lithium-ion batteries via long short-term memory network," *IEEE access*, vol. 7, pp. 53 792–53 799, 2019.

[186] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, and A. Emadi, "Long short-term memory networks for accurate state-of-charge estimation of li-ion batteries," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 8, pp. 6730–6739, 2018.

[187] B. V. Rajanna and M. Kumar, "Comparison of one and two time constant models for lithium ion battery," *International Journal of Electrical and Computer Engineering*, vol. 10, pp. 670–680, 02 2020.

[188] G. Monsalve, A. Cardenas, and W. Martinez, "Analysis of two equivalent circuit models for state of charge estimation using kalman filters," in *2022 IEEE 31st International Symposium on Industrial Electronics (ISIE)*, 2022, pp. 347–353.

[189] G. dos Reis, C. Strange, M. Yadav, and S. Li, "Lithium-ion battery data and where to find it," *Energy and AI*, vol. 5, pp. 100 081–, 2021.

[190] P. Kollmeyer, "Panasonic 18650pf li-ion battery data," Jun 2018. [Online]. Available: https://data.mendeley.com/datasets/wykht8y7tg/1

[191] E. Chemali, P. J. Kollmeyer, M. Preindl, R. Ahmed, and A. Emadi, "Long short-term memory networks for accurate state-of-charge estimation of li-ion batteries," *IEEE transactions on industrial electronics (1982)*, vol. 65, no. 8, pp. 6730–6739, 2018.

[192] C. Li, F. Xiao, and Y. Fan, "An approach to state of charge estimation of lithium-ion batteries based on recurrent neural networks with gated recurrent unit," *Energies*, vol. 12, no. 9, 2019. [Online]. Available: https://www.mdpi.com/1996-1073/12/9/1592

[193] K. L. Wong, M. Bosello, R. Tse, C. Falcomer, C. Rossi, and G. Pau, "Li-ion batteries state-of-charge estimation using deep lstm at various battery specifications and discharge cycles," in *Proceedings of the Conference on Information Technology for Social Good*, ser. GoodIT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 85–90. [Online]. Available: https://doi.org/10.1145/3462203.3475878

[194] S. Singh, M. J. Kulshrestha, N. Rani, K. Kumar, C. Sharma, and D. Aswal, "An overview of vehicular emission standards," *Mapan*, vol. 38, no. 1, pp. 241–263, 2023.

[195] The MathWorks Inc., "Optimization toolbox version: R2023b," 2023. [Online]. Available: https://www.mathworks.com

[196] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[197] F. Yang, X. Song, F. Xu, and K.-L. Tsui, "State-of-charge estimation of lithium-ion batteries via long short-term memory network," *IEEE Access*, vol. 7, pp. 53 792–53 799, 2019.

[198] A. Basia, Z. Simeu-Abazi, E. Gascard, and P. Zwolinski, "Review on state of health estimation methodologies for lithium-ion batteries in the context of circular economy," *CIRP Journal of Manufacturing Science and Technology*, vol. 32, pp. 517–528, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1755581721000249

[199] The MathWorks Inc., *Curve Fitting Toolbox*, Natick, Massachusetts, United State, 2023. [Online]. Available: https://www.mathworks.com/products/curvefitting.html