Università degli Studi di Cassino e del Lazio Meridionale
Corso di Dottorato in Metodi, Modelli e Tecnologie per l'Ingegneria
Dipartimento di Ingegneria Elettrica e dell'Informazione



Artificial Intelligence "at the Edge" for Resource-Constrained Devices Based on SENSIPLUS Technology

Michele Vitelli

michele.vitelli@unicas.it

In Partial Fulfillment of the Requirements for the Degree of
PHILOSOPHIAE DOCTOR in
Electrical and Information Engineering

18/12/2024

TUTOR
Prof. Mario Molinara

COORDINATOR
Prof. Fabrizio Marignetti

UNIVERSITÀ DEGLI STUDI DI CASSINO E DEL LAZIO MERIDIONALE

CORSO DI DOTTORATO IN METODI, MODELLI E TECNOLOGIE PER L'INGEGNERIA

Date: **18/12/2024**

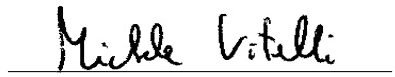
Author: **Michele Vitelli**

Title: **Artificial Intelligence "at the Edge"for Resource-Constrained Devices Based on SENSIPLUS Technology**

Department: **DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE**

Degree: **PHILOSOPHIAE DOCTOR**

Permission is herewith granted to university to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.



Signature of Author

THE AUTHOR RESERVES OTHER PUBLICATION RIGHTS, AND NEITHER THE THESIS NOR EXTENSIVE EXTRACTS FROM IT MAY BE PRINTED OR OTHERWISE REPRODUCED WITHOUT THE AUTHOR'S WRITTEN PERMISSION.

THE AUTHOR ATTESTS THAT PERMISSION HAS BEEN OBTAINED FOR THE USE OF ANY COPYRIGHTED MATERIAL APPEARING IN THIS THESIS (OTHER THAN BRIEF EXCERPTS REQUIRING ONLY PROPER ACKNOWLEDGEMENT IN SCHOLARLY WRITING) AND THAT ALL SUCH USE IS CLEARLY ACKNOWLEDGED.

Contents

List of figures **vi**

List of tables **x**

Glossary **xii**

Acronyms **xiv**

Abstract **1**

1 Introduction **3**

2 The SENSIPLUS Platform **9**

2.1 Sensichips S.r.l. 11

2.2 Applications and Sensing Platforms 11

2.2.1 Pollution Monitoring 11

2.2.2 Smart Batteries 14

2.3 Software Infrastructure 16

2.3.1 Embedded API 17

3 Open-Source Tiny Machine Learning Toolchain **21**

3.1 Related works 22

3.2 Architecture 23

3.2.1 Module for Code Generation 25
3.2.2 Module for Firmware Generation and Upload 27
3.3 Inference Performance Evaluation 29
3.3.1 Numerical precision and inference time analysis 29
3.3.2 Analysis of inference accuracy through a real use case 30
3.4 Conclusions 32

4 In-Water Environmental Monitoring 37
4.1 State of the Art 38
4.2 A dataset for detecting pollutants in wastewater using low-cost, real-time sensors 40
4.2.1 Measurements setup and adopted sensors 40
4.2.2 The Dataset 41
4.3 An IoT-based platform for real-time wastewater pollutants spilling recognition 44
4.3.1 Metodology 45
4.3.2 Classification 50
4.3.3 Experimental Results 58
4.3.4 Field tests 63
4.3.5 Discussion 65
4.3.6 Conclusions and Future Directions 68
4.4 Natural Language Generation pollution detection with low-cost sensor platform 69
4.4.1 System at a glance 69
4.4.2 Data 74
4.4.3 Experiments 75
4.4.4 Results 81
4.4.5 Discussion and Conclusion 86

5 Smart Batteries 89

5.1 Statistical Approach to LFP Battery State of Charge Analysis . . . 90
5.1.1 Experimental campaign . . . 91
5.1.2 Description of the considered approach . . . 94
5.1.3 Preliminary results . . . 96
5.1.4 Conclusions . . . 98
5.2 Battery Equivalent Series Resistance for Shuntless Coulomb Counting and SOC Estimation . . . 100
5.2.1 Proposed Shuntless SOC Measurement Method . . . 102
5.2.2 Experimental SOC Estimation Results . . . 105
5.2.3 Discussion and Conclusions . . . 110
5.3 New method for EIS-based SoC estimation optimizes accuracy and time. . . . 113
5.3.1 The proposed method . . . 116
5.3.2 The proposed method: characterization of batteries under test 118
5.3.3 The proposed method: Classifier Selection . . . 122
5.3.4 In–depth analysis . . . 124
5.3.5 The proposed method: feature selection . . . 125
5.3.6 Performance evaluation . . . 131
5.3.7 Case study A: Accuracy target . . . 132
5.3.8 Case study B: Accuracy target and minimum measurement time . . . 135
5.3.9 Conclusions . . . 135

Summary and conclusions **139**

Bibliography **142**

List of Figures

2.1 The SENSIPLUS Chip. 10
2.2 Smart Cable Air (SCA). 12
2.3 Smart Cable Water. 14
2.4 Battery Cell Management Unit. 16
2.5 Embedded API Architecture. 18

3.1 Toolchain Architecture. 24
3.2 Workflow for the Code Generator module. 25
3.3 A Smart Cable Water (SCW) connected to the MCU through the SENSIBUS, a proprietary one-wire bus adopted by SPC. On the SCW are visible the IDEs in different metals and the SPC. 26
3.4 Directory structure. 28
3.5 The execution model of GenericTarget class. 29
3.6 Evaluation of inference time for different MLP configurations. The number of neurons in the input and hidden levels varies while the output is always fixed at six. 33
3.7 Evaluation of the inference time for different MLP configurations. The number of neurons in the input and hidden layers varies in the first subplots, while the output is always fixed at six. In the second subplot, the performance of the model used in the real use case is highlighted with the number of neurons per layer fixed. 33
3.8 Summary confusion matrix, obtained from 10 folds, with mean value and standard deviation. 34

4.1 Measurement setup 41
4.2 Example of acquired signals. 44
4.3 The overall detection system deployed 46
4.4 Smart Cable Water 46
4.5 Measurement Set-Up for dataset acquisition 48

4.6 Randles at different frequencies 49
4.7 Data Set Structure. Exp 0, Exp 1, ..., Exp 9 means respectively acquisition 0, 1, ..., 9 of all substances 51
4.8 An overall view of the system 51
4.9 Finite state machine 52
4.10 Finite State Machine Flow Chart 54
4.11 Entire System Flow Chart 56
4.12 Multiclass Classifier Results 62
4.13 Entire System Results 62
4.14 Borgopiave. Green circle represents the sensing manhole, while red circles represent the spiking manhole positioned at 60m from the sensing manhole. 63
4.15 Castelbottaccio. The green circle represents the sensing manhole, while red circles represent the spiking manholes respectively positioned at 50m, 75m, and 150m from the sensing manhole 64
4.16 The flotation system for the immersion of the SCW. In red are highlighted some dirt. 64
4.17 Measurement system prototype 65
4.18 The hardware acquisition chain at a glance. 69
4.19 Smart Cable Water with its Inter Digitated sensors 70
4.20 The software chain at a glance. 72
4.21 Finite State Machine. 73
4.22 Example of acquired signals. The figure shows a subset of the signals present in the dataset. In detail, there are two features collected on four different substances. For each substance, there are nine experiments. 74
4.23 Average effectiveness metrics computed on the test sets for $F1_W$. The dotted line represents the best baseline method. 83
4.24 Distribution of the real and predicted values for the T5 (100-samples) model. 84
4.25 Average effectiveness metrics were obtained when varying the number of timestamps t_b fed to the T5 model. 85

5.1 Schematic diagram of the developed measurement set-up. 92
5.2 Implemented measurement procedure. 94
5.3 Nyquist diagram for a battery and a subset of measured SoCs. . . . 95
5.4 Structure of the obtained experimental dataset. 96
5.5 Example of the pdf estimated by the KDE technique. 97
5.6 Output of the KDE technique applied to the a) real part and b) imaginary part of the EIS measurements. 98

5.7 Output of the KDE technique applied to the EIS measurements: a) trend of the Peak value of the pdf with respect to the SoC for the real part, b) trend of the Width value of the pdf with respect to the SoC for the real part, c) trend of the Peak value of the pdf with respect to the SoC for the imaginary part, d) trend of the Width value of the pdf with respect to the SoC for the imaginary part. . . . 99
5.8 Operation of the proposed method, divided into two phases: calibration phase where the LUTs are built, and operational phase, where the LUTs are used together with voltage measurements to estimate the current and SOC. 103
5.9 The pseudo-code for the Algorithm proposed. 104
5.10 WLTP current profile used to validate the proposed method in the operational phase. 107
5.11 Experimental data: OCV as a function of SOC. The curves were obtained in the calibration phase for different values of the discharging current. 108
5.12 Experimental data: ESR as a function of SOC. The curves were obtained in the calibration phase for different values of the discharging current. 109
5.13 An example of the best-fit line computed during the operational phase to obtain an estimate of OCV from an estimate of the current. The fit was performed on four data samples from the LUTs acquired at different currents: 250 mA, 500 mA, 750 mA, and 1000 mA. . . . 110
5.14 Experimental results in the operational phase. (a): Current estimated by the proposed shuntless method with a WLTP profile; (b): magnification of the curve in (a), showing a single repetition of the WLTP profile; (c): current estimation error, computed as the difference between the estimated and reference value of the current. . . . 111
5.15 Experimental results in the operational phase: (a) SOC estimated by the proposed shuntless method; (b) estimation error. 112
5.16 The proposed method workflow. 118
5.17 Implemented experimental set–up. 119
5.18 Adopted protocol for batteries characterization. 122
5.19 Obtained Nyquist plot of a single battery at different State of Charge. 123
5.20 Structure of the entire obtained experimental dataset. 123
5.21 Obtained confusion matrix of the Support Vector Machine model, with mean value (top) and standard deviation (bottom) for each class. 127
5.22 Genetic Algorithm (left) and Particle Swarm Optimization (right) flow charts. 128

5.23 Analysis of Accuracy and measurement time as a function of alpha weighting coefficient for different configurations of search algorithms and fitness functions. 133
5.24 Performance analysis and distribution of selected features using different combinations of search algorithms and fitness functions for SoC estimation in the two scenarios A and B, considering the constraints imposed. 134

List of Tables

2.1 Specifications of the Smart Cable Air (SCA) 13
2.2 Specifications of the Smart Cable Water (SCW) 15
2.3 Battery Cell Management Unit 17

3.1 MLP tested configurations. 29
3.2 Selected features (sensors). 31
3.3 Details of the chemical composition of substances. In particular, for each analyte, 10 ml has been inserted in 100 ml of wastewater, the composition of this 10 ml is described in the analyte column. 35

4.1 Details of chemical composition of substances 42
4.2 Aquired physical quantities. 43
4.3 Synthetic waste water chemical composition. 48
4.4 Anomaly Detection Models Parameters 57
4.5 Multiclass Classification Model Parameters 57
4.6 Best Results Anomaly Detection 61
4.7 CrossValidation0 Results 61
4.8 Sensors used in the experiments. 71
4.9 Substances used in the experiments. 75
4.10 Number of samples by substance. For each acquisition, 600 samples are collected in Wastewater and 1000 samples after a substance injection. Each fold's training set comprises 9 acquisitions for each substance, while the test set comprises 1 acquisition. 76
4.11 The selected hyper-parameters for the various ML algorithms employed. 78
4.12 Average effectiveness metrics computed over the test sets. metric_{m} denotes the micro-averaged metric, metric_{M} the macro-averaged metric, and metric_{W} the weighted metric. 82

5.1 Settings used for the experimental characterization. 93
5.2 Main characteristics of the considered batteries. 120
5.3 Adopted conditions for the experimental characterization. 121
5.4 Average values of the considered evaluation metrics and main best obtained hyperparameters for the ML models. 125
5.5 Average values and standard deviation of the considered evaluation metrics examined, where Train and Test datasets are divided by batteries. 126
5.6 Genetic Algorithm parameters. 128
5.7 Particle Swarm Optimization parameters. 130
5.8 Comparison of the analyzed scenarios based on some application examples. The first scenario (A) emphasizes accuracy, neglecting measurement time, while the second (B) achieves a use–case–dependent trade–off. 136

Glossary

AI on the Edge	"AI on the Edge" refers to deploying artificial intelligence algorithms directly on edge devices, such as sensors, cameras, or microcontrollers, allowing data processing and decision-making to occur locally, near the data source, rather than relying on centralized cloud computing. This approach reduces latency, enhances privacy, and improves the efficiency of real-time applications.
BLE	Bluetooth Low Energy (BLE) is a wireless communication technology designed for short-range data exchange with minimal power consumption. It's widely used in applications like wearable devices, smart home systems, and IoT sensors.
Electrochemical Impedance Spectroscopy	Electrochemical Impedance Spectroscopy (EIS) is a powerful analytical technique used to study the electrical properties of materials and systems, especially those involving electrochemical processes such as batteries, fuel cells, corrosion, and electroplating.
I^3C	Improved Inter Integrated Circuit (I^3C) is a specification that enables communication between computer chips by defining the electrical connection between the chips and the signaling patterns to be used. Evolution of the I^2C protocol.

Internet of Things	Internet of Things (IoT) is a network of devices that can sense, process, and exchange data with other devices and systems over the Internet.
SENSIBUS	is a proprietary single data line digital serial bus designed for the cabled networking of sensors and actuators.
Sensichips	Sensichips S.r.l. is an Italian company specializing in the development of smart sensor systems and microelectronic chips for environmental, industrial, and health monitoring. Their technology integrates sensors, signal processing, and machine learning to enable real-time data analysis and decision-making.
SENSIPLUS	Microelectronics platform designed to integrate multiple heterogeneous sensors on a single chip.
Smart Cable Air	Smart Cable Air (SCA) is a compact multi-sensor system designed for air quality monitoring, developed by Sensichips.
Smart Cable Water	Smart Cable Water (SCW) is a multi-sensor system designed for the monitoring of water quality, developed by Sensichips.
SPI	Serial Peripheral Interface (SPI) is a standard for synchronous serial communication used primarily in embedded systems for short-distance wired communication between integrated circuits.
Wi-Fi	Wi-Fi is a wireless networking technology that allows devices to connect to the internet and communicate with each other over a local area network (LAN) using radio waves. It provides high-speed data transfer and is commonly used in homes, offices, and public spaces to enable internet access on devices like smartphones, laptops, and smart appliances..
ZigBee	Zigbee is a low-power, wireless communication protocol designed for IoT devices, home automation, and sensor networks.

Acronyms

AI	Artificial Intelligence.
API	Application Programming Interface.
APIs	Application Programming Interfaces.
BEVs	Battery Electric Vehicles.
BLE	Bluetooth Low Energy.
BMS	Battery Management System.
CMOS	Complementary Metal-Oxide Semiconductor.
CMU	Battery Cell Management Unit.
DL	Deep Learning.
ECM	Equivalent Circuit Model.
EIS	Electrochemical Impedance Spectroscopy.
ESR	Equivalent Series Resistance.
EV	Electric Vehicle.
EVs	Electric Vehicles.
I^2C	Inter Integrated Circuit.
I^3C	Improved Inter Integrated Circuit.
IDE	InterDigitated Electrode.
IDEs	InterDigitated Electrodes.
IoT	Internet of Things.
KDE	Kernel Density Estimation.
LFP	Lithium Iron Phosphate.
LUT	Look-Up Table.

MCU	Micro Controller Unit.
MCUs	Micro Controller Units.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
MSM	Multi Sensor Microsystem.
NLP	Natural Language Processing.
OCV	Open Circuit Voltage.
PC	Personal Computer.
PDF	Probability Density Function.
RVFL	Random Vector Functional Link.
SCA	Smart Cable Air.
SCW	Smart Cable Water.
SENSIPLUS-LM	SENSIPLUS Learning Machine.
SoC	State of Charge.
SoH	State of Health.
SPC	SENSIPLUS Chip.
SPI	Serial Peripheral Interface.
SPIFFS	Serial Peripheral Interface Flash File System.
TF	Transfer Learning.
TIC	Toxic Industrial Chemical.
TICs	Toxic Industrial Chemicals.
TinyML	Tiny Machine Learning.
VOCs	Volatile Organic Compounds.
WW	WasteWater.

Abstract

Artificial Intelligence (AI) and the proliferation of edge devices have advanced 'AI at the edge', allowing AI models to be deployed directly on devices closer to data sources or end users. Unlike traditional cloud-based approaches that rely on remote data centers, edge AI processes data locally on devices such as smartphones, cameras, and IoT sensors. This local processing reduces latency, lowers bandwidth requirements, improves privacy, and increases reliability. It is valuable for applications with stringent real-time requirements or intermittent connectivity, such as autonomous vehicles, industrial automation, and smart city systems. The rise of edge AI is closely linked to advances in the IoT and Industry 4.0, emphasizing interconnected, autonomous systems. By processing data locally, edge AI reduces network load, allowing IoT devices to perform tasks such as predictive maintenance, real-time monitoring, and autonomous decision-making without cloud delays. However, deploying AI on edge devices is challenging due to limited processing power, memory, and battery life, especially in mobile or resource-constrained environments. Complex AI models require significant computational resources, making optimization techniques such as model compression and quantization essential. Battery-powered devices face additional challenges due to the power requirements of local AI processing, requiring energy-efficient algorithms and specialized hardware. In addition, model updates are required to adapt to dynamic real-world data, creating logistical challenges for consistent updates across devices. Maintaining data quality in an environment of noise is also critical, as poor-quality data can degrade AI performance. The research in this thesis focuses on environmental monitoring and intelligent battery management applications for edge AI. It enables real-time sensor data analysis for rapid environmental response and optimizes battery parameters for energy efficiency in electric vehicles and renewable energy systems. Using Sensichips s.r.l.'s SENSIPLUS sensor platform, these works demonstrate the potential of edge AI in scalable, real-time applications across multiple industries, and highlight the need for high-quality sensors and efficient algorithms in robust edge AI systems. This thesis describes the development and application of Edge

Abstract

AI methods for environmental monitoring and smart battery systems. In environmental monitoring, AI algorithms are used to analyze real-time data from sensors to detect environmental changes. This enabled timely responses, such as triggering alarms for air quality hazards or adjusting system operations. Research also demonstrated the effective detection of pollutants in water for environmental protection. In smart battery systems, the research applied Edge AI to optimize battery performance. This improved battery life prediction and energy consumption. Overall, the research demonstrated that Edge AI can provide robust real-time processing capabilities in environmental and energy-related applications.

Chapter 1

Introduction

AI at the Edge refers to the deployment of Artificial Intelligence (AI) algorithms and models directly on edge devices, which are closer to the data source or end user. This approach differs from the cloud-based AI model, where data is sent to centralized servers for processing. In the cloud model, data must travel to a remote data center, be processed, and then returned to the edge device with the results. In contrast, AI at the Edge brings computation and intelligence directly to devices such as smartphones, cameras, or Internet of Things (IoT) sensors, enabling local data processing and decision-making. The key distinction between edge AI and centralized cloud AI lies in the location of data processing. In edge AI, data is processed locally on the device or within proximity (e.g., on local gateways or edge servers). This localized processing offers several advantages:

- *Reduced Latency*: Edge AI enables faster decision-making by processing data near the source. In real-time applications, such as autonomous vehicles, drones, or industrial automation, even milliseconds can be critical. Centralized cloud processing may introduce delays due to the time taken to transmit data to the cloud and back.
- *Lower Bandwidth Requirements:* With edge AI, only relevant data (or processed insights) need to be transmitted to the cloud, reducing the amount of data that travels over networks. This is crucial in bandwidth-constrained environments like remote industrial sites or areas with poor connectivity.
- *Enhanced Privacy:* Since sensitive data can be processed locally on the device, fewer data transfers reduce the risk of exposure, which is beneficial for applications requiring strong data privacy, such as healthcare or surveillance.

- *Reliability:* Edge AI systems are less reliant on continuous internet connectivity. In environments where network access is intermittent or unreliable, such as rural areas or manufacturing plants, edge devices can continue to operate independently.

Edge AI is closely tied to the proliferation of the Internet of Things (IoT) and the vision of Industry 4.0, the next phase of industrial development characterized by interconnected, autonomous systems. IoT devices generate massive amounts of data from a wide range of sensors. If all this data were sent to the cloud for processing, it would overwhelm network infrastructure and introduce latency. By integrating AI at the edge, IoT devices can process data locally, perform real-time analytics, and send only the most relevant insights to centralized systems. This decentralization increases the efficiency and scalability of IoT networks, particularly in applications such as smart homes, predictive maintenance, and environmental monitoring. In manufacturing and industrial environments, Industry 4.0 envisions factories and supply chains where machines are equipped with sensors and AI to perform tasks autonomously. Here, edge AI enables machines to analyze sensor data in real-time, enabling faster problem detection, predictive maintenance, and optimization of production processes. This results in reduced downtime, improved product quality, and increased operational efficiency. Edge AI helps these systems achieve the low latency and high reliability needed to operate autonomously without relying on external cloud services. These approaches are perfect for real-time scenarios where rapid decision-making is paramount, and Edge AI delivers the speed and responsiveness required. For example, autonomous vehicles rely on instant interpretation of sensor data to make split-second decisions. Routing data through a distant cloud could result in dangerous delays. Similarly, industrial robots on a factory floor need to analyze data instantly to avoid production bottlenecks or safety hazards. Real-time AI is also critical in smart cities, where surveillance systems, traffic management, and energy grids need to adapt quickly to changing conditions. Edge AI ensures that these systems can respond in real-time, improving efficiency and safety.

There are several key issues and challenges associated with deploying AI on edge devices, stemming from the unique constraints of these systems compared to cloud-based AI. These challenges must be addressed to ensure optimal performance, reliability, and scalability:

- These devices typically have limited processing power, memory, and storage compared to centralized cloud servers. Running complex AI models, especially those that require significant computational power (e.g., deep learning

models), can be resource-intensive, leading to potential performance bottlenecks. This requires optimization techniques such as model compression, pruning, or quantization to reduce the computational load while maintaining accuracy.

- Many edge devices are battery-powered or have limited power supplies, especially in remote or mobile applications (e.g., drones, autonomous vehicles, or IoT devices). Running AI models locally can consume significant power, reducing battery life and impacting the operational lifetime of the device. Developing energy-efficient AI algorithms and leveraging low-power hardware (such as specialized AI chips or accelerators) is critical to overcoming this limitation.

- AI models must be frequently updated or retrained to maintain their accuracy and relevance, especially in dynamic environments where data distribution can change over time (known as model drift). Deploying and updating models across thousands or millions of distributed edge devices is a significant logistical challenge. Efficient over-the-air (OTA) updates, remote monitoring, and model version control systems are essential for maintaining AI systems at the edge.

- Edge devices often operate in environments where data quality can vary due to sensor inaccuracies, noise, or environmental interference. Poor quality data can negatively impact the performance of AI models, resulting in incorrect predictions or decisions. Managing data collection, filtering out noise, and ensuring the integrity of sensor data are critical for reliable AI results. In addition, efficient data management practices must balance local storage constraints with the need for comprehensive data analysis.

Overcoming these challenges is critical to unlocking the full potential of AI at the edge and enabling robust, real-time, and scalable AI applications in areas such as healthcare, autonomous systems, smart cities, and industrial automation.

Although applicable regardless of the final context, the research activities carried out and the methodologies shown in this thesis focused mainly on the environmental monitoring field and the smart batteries context. In environmental monitoring, edge AI analyzes data from sensors that track variables such as temperature, humidity, air quality, and pollution levels in real-time. This enables rapid responses to environmental changes, such as triggering alarms for air quality hazards or adjusting systems for optimal energy use. Similarly, in battery-powered systems, AI at the edge plays a critical role in estimating battery parameters by

continuously analyzing factors such as state of charge, temperature, and degradation patterns. This helps optimize energy consumption and extend battery life in electric vehicles, IoT devices, and renewable energy storage systems. In both areas, sensors are fundamental, serving as the primary source of data and providing real-time input to the AI models. The accuracy, sensitivity, and integration of these sensors directly affect the performance and reliability of edge AI systems, making them key components in the overall architecture. This document is divided into several chapters organized by topic. First, chapter 2 describes in detail the SENSIPLUS sensor platform developed by Sensichips s.r.l. It introduces the company and the key role of the SENSIPLUS chip as the main component of the platform. The chapter discusses various applications of sensors, emphasizing the importance of combining sensors and Artificial Intelligence algorithms to extract meaningful information from the collected data. Several prototype sensor platforms are presented, and the chapter concludes with a description of the software infrastructure, in particular the embedded API that allows developers to interact with the SENSIPLUS chip. The chapter 3 focuses on the development of an open source toolchain for the SENSIPLUS platform, specifically designed for implementing Machine Learning (ML) models on microcontrollers. The toolchain simplifies the process of generating firmware and deploying ML models on edge devices by enabling local data processing. The architecture of the toolchain is described, highlighting its main components and workflow. A practical case study is also presented, in which an Multi-Layer Perceptron (MLP) model is converted to C and tested on a target device. The performance of the model in terms of numerical accuracy and inference time is analyzed in detail, demonstrating the effectiveness of the toolchain for Edge AI applications. The next chapter (chapter 4) discusses the importance of environmental monitoring, particularly for water quality. The role of sensors and artificial intelligence in detecting pollutants and ensuring the sustainability of ecosystems is highlighted. The chapter focuses on the development of IoT-based pollutant detection systems using the SENSIPLUS chip and an Micro Controller Unit (MCU) to create a miniaturized, low-power, IoT-enabled sensor platform. Two specific projects will be described in detail:

- Internet of Things (IoT) Platform for Real-Time Detection of Pollutants in Wastewater: This project focuses on the early detection of contaminant spills in wastewater using sensors, machine learning algorithms, and a finite state machine.
- Pollution detection with natural language generation using a low-cost sensor platform: This project explores the use of Natural Language Processing (NLP) to describe and predict the presence of pollutants.

Both sections provide a detailed description of the system, the methodology used, the results obtained, and possible future research directions. The chapter 5 focuses on the importance of smart batteries, equipped with sophisticated Battery Management System (BMS), in the transition to cleaner energy sources. The key role of batteries in applications ranging from Electric Vehicle (EV) to renewable energy storage systems is highlighted. The chapter addresses the challenges associated with State of Charge (SoC) estimation and battery diagnostics and proposes innovative solutions based on Electrochemical Impedance Spectroscopy (EIS) and Machine Learning (ML) algorithms. Three different sections are presented:

- Statistical approach to Lithium Iron Phosphate (LFP) battery SoC analysis: This section explores the use of the Kernel Density Estimation (KDE) technique to estimate the state of charge (SoC) of LFP batteries.
- Equivalent Battery Series Resistance for Shuntless Coulomb Counting and SoC Estimation: This section proposes a shuntless SoC estimation method that exploits the knowledge of the internal resistance of the battery.
- Novel method for EIS-based SoC estimation optimizes accuracy and time: This section presents a novel method for EIS-based SoC estimation that aims to minimize measurement time while maintaining high accuracy.

Each section provides an overview of the problem, the methodology employed, experimental results, and conclusions. Finally, the conclusions of the research projects are presented.

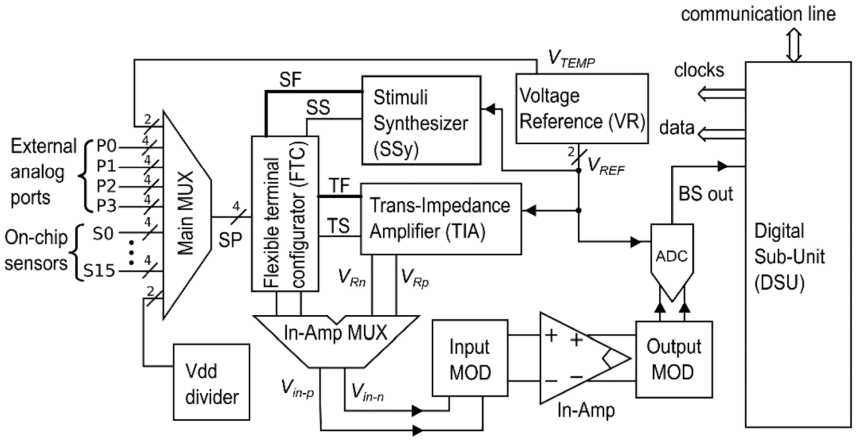
Chapter 2

The SENSIPLUS Platform

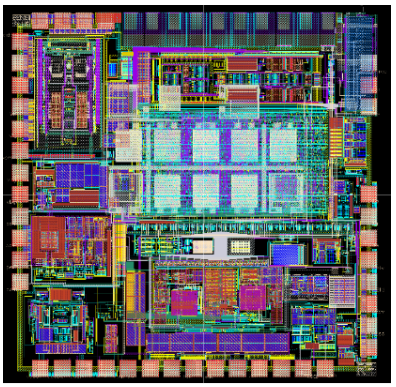
The SENSIPLUS microelectronics platform (Figure 2.1) has been developed to integrate a multitude of heterogeneous sensors into a single chip or compact Multi Sensor Microsystem (MSM). The chip aims to merge a variety of functional materials with Complementary Metal-Oxide Semiconductor (CMOS) microelectronics to create a single-chip microsensor. These devices address various environmental sensing requirements, and the platform is sufficiently versatile to be applied to other fields. At the core of the SENSIPLUS platform is a highly precise and adaptable analytical microchip incorporating a wideband electrical impedance spectrometer based on a frequency-programmable lock-in amplifier and a potentiostat. This is all housed within a compact 2x2 mm chip that draws only 2 mW of power when fully operational. Upgrades to these measurement instruments facilitate compatibility with various chemical, optical, magnetic, and acoustic sensors. Further integration is achieved by creating sensors with additional functional and sensitive materials (such as metal oxide) using proprietary post-CMOS and MSM manufacturing techniques. Artificial Intelligence and analytical algorithms facilitate the cooperative operation of multiple sensors, with hardware support provided by SENSIPLUS. There are not many sensor systems comparable to this chip, especially in terms of miniaturization. From the point of view of functionality, the device that shares some of its functionality is certainly the CN0510 circuit produced by Analog Devices [1], which is an Electrochemical Impedance Spectroscopy (EIS) measurement system, but the lack of features that are fundamental to its intended objectives do not make it a viable choice.

The SENSIPLUS chip can be deployed on smart cables via a proprietary single data wire cable bus, which supports synchronous operation and efficient data acquisition from distributed sensor arrays. Includes standard Inter Integrated Circuit (I^2C)/Improved Inter Integrated Circuit (I^3C) and Serial Peripheral Interface (SPI)

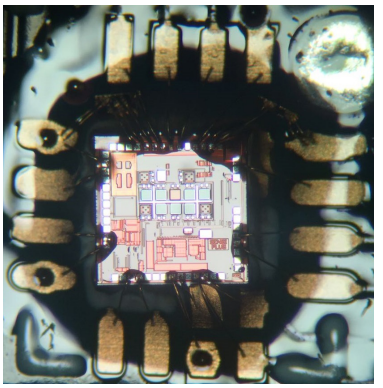
serial digital interfaces and analog output. To effectively address the hardware complexity, a software architecture was developed to facilitate the management of the chip's functionality by exposing its capabilities through a system of Application Programming Interfaces (APIs).



(a) Simplified block diagram of the SENSIPLUS chip.



(b) Design layout.



(c) Picture taken with a microscope.

Figure 2.1: The SENSIPLUS Chip.

2.1 Sensichips S.r.l.

Sensichips is the company that engineered and developed the SENSIPLUS chip. It is a relatively small company that is involved in the field of learning microsensors. The company is headquartered in Anzio, Italy. The company was established in 2011 by experienced entrepreneurs from the semiconductor industry and Arescosmo SpA, a company active in the aerospace and defense sectors. The technology developed by Sensichips can set new performance standards and enable new applications in several fields involving sensors, including robotics, electrification, wearable technology, healthcare automation, and the Internet of Things. The team comprises engineers and researchers with expertise in microelectronics, firmware, artificial intelligence, material science, and electrochemistry. The company houses a chemical laboratory, application development, administration, and management. Research and development facilities are also located at the University of Pisa and the University of Cassino and Southern Lazio.

2.2 Applications and Sensing Platforms

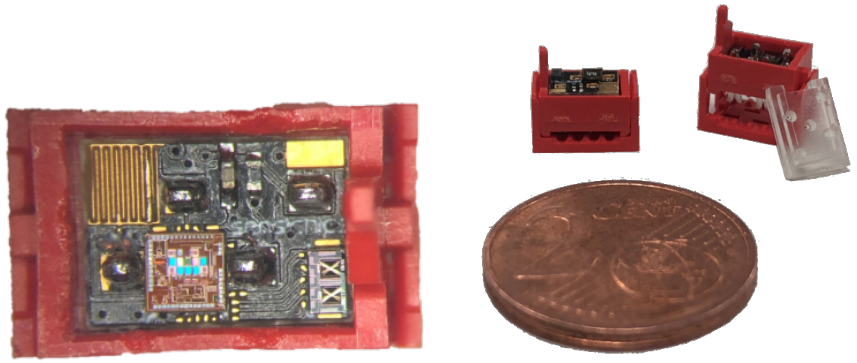
Based on the SENSIPLUS, several boards are produced for specific applications that complement the chip's capabilities by adding external sensors, mainly electrochemical sensors. Most of these sensors are analyzed using the technique of Electrochemical Impedance Spectroscopy(EIS), which correlates different physical phenomena with changes in the associated impedance. For example, in the context of the automotive industry, it is possible to correlate the impedance of the energy cell with its state of health. The same principles can be applied to other areas of interest, such as pollution detection or biomedical applications. Artificial intelligence algorithms can enhance such information from different sources to relate the variation of these quantities to physical phenomena. In addition, the goal of bringing this intelligence close to the sensors seems to be the beginning of interesting scenarios where the sensor directly provides qualified data and simplifies and streamlines communication infrastructures. This section presents prototypes of various Sensing Platforms that can be used as the basis for acquisition sensing systems in various applications, as described in the following chapters.

2.2.1 Pollution Monitoring

Air and water pollution monitoring has become a critical global challenge due to increasing pollution from industrial activities, urbanization and agricultural practices. Air pollution, characterized by particulate matter, nitrogen oxides, and volatile organic compounds, poses significant health risks, including respiratory

and cardiovascular diseases. Similarly, water pollution, often caused by the discharge of industrial effluent, agricultural runoff, and untreated sewage, leads to the degradation of aquatic ecosystems and threatens human health by contaminating drinking water sources. These pollutants' complexity and widespread impact require effective monitoring systems to detect and quantify pollutants in real time. Accurate pollution monitoring is essential for assessing environmental quality, enforcing regulations, and implementing pollution mitigation measures. However, challenges such as the need for sensitive detection technologies, the diverse nature of pollutants, and the variability of environmental conditions make this task complex and require innovative solutions and advances in sensor technology, data analysis, and regulatory frameworks.

Smart Cable Air



(a) Top view. (b) SCA dimension.

Figure 2.2: Smart Cable Air (SCA).

Sensichips has developed the Smart Cable Air(SCA) as a compact and low-power solution for air quality monitoring, with a size of approximately 8x9 mm (see Figure 2.2). The SCA is a Multi Sensor Microsystem (MSM) primarily designed to detect the presence of Toxic Industrial Chemicals (TICs), pollutants, Volatile Organic Compounds (VOCs), and combustible gases in the atmosphere. At the heart of the card is the SENSIPLUS platform, the measurements are primarily based on Electrochemical Impedance Spectroscopy (EIS). This measurement technique utilizes sensitive films' chemisorption and redox dynamics to facilitate gas discrimination. In addition, EIS allows the derivation of electrical equivalent circuit parameters for the sensor, separating components that exhibit drift from those that directly correspond to the gas response. Multiple SCAs can be deployed

along the same cable to monitor a large area continuously. Combined with the versatile analytical capabilities of the onboard InterDigitated Electrodes (IDEs), the SCA serves as an excellent experimental platform. Table 2.1 outlines the key specifications associated with the SCA.

ELECTRICAL	
Supply Voltage	1.5-3.6V
Max Current	1.4mA continuous when reading on-chip sensors with EIS
Size	5.5x3.9mm
Interface	I3C or SENSIBUS, single data wire multidrop sensor array cable interface, 1.5-3.6V
Unique Identifier	OTP 48bits Unique Device Identifier, 16bits User Defined
ELECTRICAL IMPEDANCE SPECTROSCOPY	
Frequency	3.1mHz to 1.2MHz
Vpp output sinewave	156mV to 2.8Vpp
Coherent demodulation	1st, 2nd, or 3rd harmonic
Output	Reciprocal of real or imaginary component
Wide Measurement Range	From ohms to 100MΩ
TEMPERATURE	
Range	-40°C to 125°C
Accuracy	±0.1°C
POLYIMIDE	
Detects	Relative Humidity, Vapors
Range	0% to 100%
Accuracy	±0.5%
ALUMINUM OXIDE	
Detects	Ionized Particles and Vapors, i.e., Acids, Ammonia dissolved in water
Limit of detection	1ppm Ammonium
HEATED TIN OXIDE	
Detects	Air contaminants, Flammables, Cigarette smoke
Limit of detection	100ppb H_2

Table 2.1: Specifications of the Smart Cable Air (SCA)

Smart Cable Water

The Smart Cable Water(SCW) is a Multi Sensor Microsystem (MSM)) designed for the monitoring of Toxic Industrial Chemical (TIC), pollutants, hydrocarbons, and organic compounds in aqueous environments (Figure 2.3). At the core of the SCW system is the SENSIPLUS platform, which represents Sensichips' cutting-edge microsensor technology. The platform is equipped with a versatile and precise Electrochemical Impedance Spectroscopy (EIS) and potentiostat, capable of interrogating both on-chip and off-chip sensors. The EIS enables the exploitation of redox dynamics in catalytic noble metals, facilitating enhanced chemical discrimination



Figure 2.3: Smart Cable Water.

and the measurement of conductivity and permittivity spectra. Furthermore, the on-chip potentiostat facilitates a range of voltammetric and amperometric measurements, enabling real-time discrimination of pollutants. To prevent or mitigate biofilm formation, the EIS employs electrode cycling with overvoltage, making it a comprehensive and multiparametric microsystem for water analysis. Due to its versatile analytical capabilities and the inclusion of onboard catalytic interdigitated electrodes, the SCW also serves as an excellent experimental platform for discriminative measurement techniques. Table 2.2 outlines the key specifications associated with the SCW.

2.2.2 Smart Batteries

Batteries are critical in several major industries, particularly in the automotive and smart energy sectors. The shift to Battery Electric Vehicles (BEVs) in the automotive industry has driven significant advances in battery technology, with lithium-ion batteries becoming the cornerstone of this transformation. These batteries provide the energy needed to power BEVs, offering a cleaner alternative to fossil fuels and reducing greenhouse gas emissions. In the smart energy sector, batteries are an integral part of energy storage systems that support renewable energy sources such as solar and wind power. By storing excess energy generated during peak production times, these batteries help balance supply and demand, ensuring a reliable and consistent energy supply even when renewable sources are not actively generating power. This capability is essential for creating more resilient and sustainable energy grids, paving the way for a low-carbon future. Batteries are critical to enabling the transition to more sustainable practices in these industries, driving innovation, and supporting the global push for cleaner energy solutions. As a result, battery characterization is a critical process for understanding and optimizing battery performance, safety, and longevity in various applications. Among

ELECTRICAL	
Supply Voltage	1.5-3.6V
Max Current	0.4mA continuous when reading on-chip sensors with EIS
Size	12x15mm, 3mm thickness
Interface	I^2C or SENSIBUS, single data wire multidrop sensor array cable interface, 1.5-3.6V
Unique Identifier	OTP 48bits Unique Device Identifier, 16bits User Defined
ELECTRICAL IMPEDANCE SPECTROSCOPY	
Frequency	3.1mHz to 1.2MHz
Vpp output sinewave	156mV to 2.8Vpp
Coherent demodulation	1st, 2nd, or 3rd harmonic
Output	Reciprocal of real or imaginary component
Wide Measurement Range	From ohms to 100MΩ
TEMPERATURE	
Range	-40°C to 125°C
Accuracy	±0.1°C
Thermodynamics	Calorimetry, enthalpy, exo/endothermic
ELECTROCHEMICAL METHODS	
pH	From 3 to 14, potential of Platinum vs Clads-Platinum
ORP	Total oxidation and reduction potentials
RedOx	Reduction/oxidation activity (free chlorine, hardness)
Voltammetry	Specific reduction or oxidation potentials
Anodic Stripping Voltammetry	Measures heavy metals
Electro-catalysis	Noble metal IDEs measure current specifically
IMPEDANCE METHODS	
Conductivity spectroscopy	Resistivity, salinity, EC, TDS, absorption dynamics
Dielectric spectroscopy	Turbidity, SS, biomass total/active, hydrocarbons detection

Table 2.2: Specifications of the Smart Cable Water (SCW)

the battery characterization techniques, Electrochemical Impedance Spectroscopy (EIS) stands out for its ability to provide detailed insights into the internal processes of a battery, enabling advancements in battery technology and helping to ensure the reliability and efficiency of energy storage systems in various industries.

Battery Cell Management Unit

The Battery Cell Management Unit (CMU) has been developed with the specific intention of providing battery engineers and researchers with a tool that will facilitate a range of measurements on batteries, supercapacitors, or fuel cells during operation (Figure 2.4). The device is constructed around the SENSIPLUS microchip, which incorporates an electrochemical impedance spectrometer with a lock-in amplifier. This enables the measurement of the first, second, and third harmonics and a potentiostat/galvanostat and on-chip sensors for temperature, relative humidity, and gas emissions. The CMU includes positive and negative tab electrodes,

which enable four-contact measurements of cell internal impedance and temperature. Furthermore, Coulomb counting can be conducted without a shunt resistor and calibrated to account for aging effects. The CMU also incorporates an off-chip interface that facilitates the integration of external sensors for strain, pressure, electrolyte leakage, and moisture. It is compatible with diverse chemistries, including Li-ion and LiFePO4 battery packs, and is also suitable for use with the third generation of LNMO cells, supercapacitors, and fuel cell combinations. Table 2.3 outlines the key specifications associated with the CMU.

2.3 Software Infrastructure

On top of the hardware is a complex software infrastructure designed to simplify the use of SENSIPLUS systems by allowing data to be collected, stored, processed, and Machine Learning (ML) algorithms to be applied. Regardless of the end use, each of these components relies on an Application Programming Interfaces (APIs) system specifically designed to access and expose the components and capabilities of the available hardware. As a member of the Sensichips team, part of the activities involved the development and maintenance of these APIs, with a focus on supporting resource-constrained devices, since one of the most interesting use cases is to build AI on the Edge systems based on commercial microcontrollers, capable of being independent of external processing systems and then directly providing qualified data (associated to physical quantities), by exploiting Artificial Intelligence algorithms specifically designed and executed on board these devices. An API designed for embedded devices provides a powerful and flexible interface for managing the SENSIPLUS chip, allowing streamlined access to the chip's sophisticated features, including Electrochemical Impedance Spectroscopy (EIS), potentiostat/-galvanostat capabilities, and integrated environmental sensors, enabling accurate and efficient real-time management of energy storage systems.

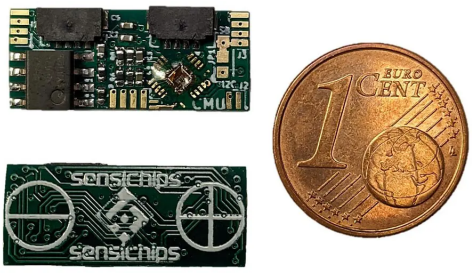


Figure 2.4: Battery Cell Management Unit.

ELECTRICAL	
Supply Voltage	2.7-4.2V directly powered by the battery cell
Max Current	35mA when reading EIS at best accuracy
Size	20x8.5mm, 2.3mm thickness
Interface	Isolated (UL1577 rating) I3C, I2C or SENSIBUS, single data wire bus
Unique Identifier	OTP 48 bits Unique Device/Cell Identifier, 16 bits User Defined
ELECTRICAL IMPEDANCE SPECTROSCOPY	
EIS	4 contacts EIS measurement of internal Cell Impedance
Frequency	100mHz to 10kHz spectroscopy
Ipp Output Sinewave	0 to 50mApp
Coherent Demodulation	1st, 2nd, or 3rd harmonic
Output	Real and Imaginary components
Measurement Resolution	150µΩ with 10µΩ accuracy at 1kHz
INTEGRATED CMU SENSORS	
Battery Voltage	±1mV accuracy in range 2.5-4.2V @ 50Hz conversion rate
Environmental Temperature	±0.1°C accuracy in range -40°C to 125°C
Environmental Humidity	Relative Humidity 0% to 100% with ±0.2% accuracy
Gas Sensing	VOCs, Vapors
Anode, Cathode Temperature	NTC thermistor 100kΩ ±1%, B value 4700K ±1%
OPTIONAL CMU SENSORS	
Strain Gauge	Cell swelling measurement
Moisture Sensor	Electrolyte leakage or moisture sensing
Body Temperature	NTC thermistor 100kΩ ±1%, B value 4700K ±1%
Extra Sensor Port	i.e., pressure sensor or additional thermistor
HIGH VOLTAGE ISOLATION	
Protection Level	Basic
Surge Test Voltage	$5000V_{PK}$
Isolation Rating	$3000V_{RMS}$
Working Voltage	$450V_{RMS}$ / $637V_{PK}$

Table 2.3: Battery Cell Management Unit

2.3.1 Embedded API

The SENSIPLUS API is structured as a C-based library designed to interact with the SENSIPLUS microsensor platform. It is organized into a two-level architecture that facilitates communication between the host system (such as a PC or other microcontrollers) and the chip, enabling various analytical measurements and data processing. Figure 2.5 shows the general structure of the architecture.

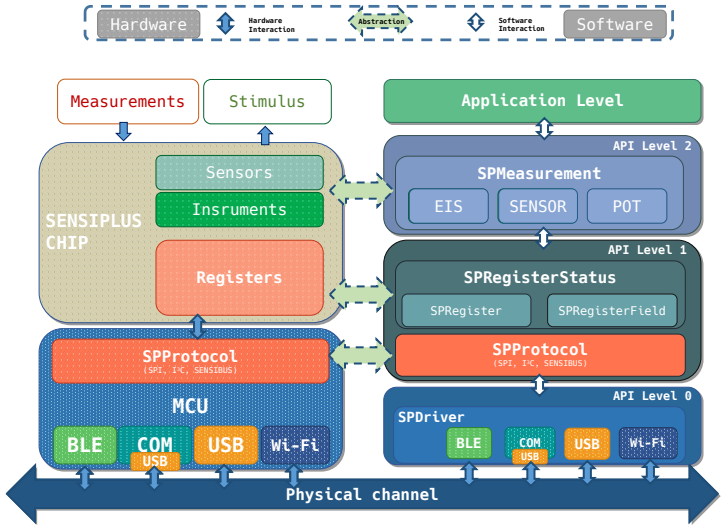


Figure 2.5: Embedded API Architecture.

1. Low-Level API (Level 0 and Level 1)

Level 0: Interface Driver Layer This foundational layer provides the interface between the host system and the chip. It acts as an abstraction layer, allowing communication through various hardware interfaces like I^3C, SPI, or SENSIBUS. The layer abstracts the specific hardware details, enabling seamless data transmission regardless of the connection type. It can also be configured to support different connectivity standards, such as BLE, ZigBee, or Wi-Fi, by replacing the bridge interface.

A key feature of this level is the synchronization of read operations, even when using asynchronous communication channels like USB with Android. This synchronization ensures that data access is consistent and reliable, with the option to include timeout notifications if operations exceed a user-defined duration.

Level 1: Data Decoding Layer This layer translates low-level calls into byte sequences sent directly to the chip. It decodes these instructions and ensures that the correct commands are issued for the intended operations on the chip.

2. High-Level API

The high-level API is designed to provide a user-friendly interface for executing various types of analytical measurements using the SENSIPLUS chip. It maintains

the complete state of the current measurement process and provides methods to set and retrieve measurement types.

Measurement Process: The API includes methods such as *setMeasureType* and *getMeasureType* for each measurement type.

- ***setMeasureType*:** This method initializes the chip's state and internal parameters based on the specified measurement type. It also validates input parameters and computes necessary values (e.g., determining available frequencies).
- ***getMeasureType*:** This method interacts with the chip to perform the actual measurements, updating the API module state with each call. Multiple *getMeasureType* calls can be made after setting the measurement type, allowing for continuous data collection and analysis.

Example Measurements:

- **Electrical Impedance Spectroscopy (EIS):**
  - *setEIS* configures the chip registers for EIS measurements, preparing both the stimulus and measurement sections.
  - *getEIS* performs the measurements in quadrature and phase, using the results to compute Resistance, Capacitance, and Inductance.
- **Potentiostat (POT):**
  - *setPOT* initializes the chip for potentiostatic measurements, configuring parameters such as the type of voltage ramp.
  - Subsequent *getPOT* calls apply the stimulus and read sensor data based on the user's timing requirements.

Chapter 3

Open-Source Tiny Machine Learning Toolchain

The advent of the Internet of Things has profoundly impacted today's society, as small, intelligent devices are increasingly integrated into various sectors ranging from healthcare to industrial applications. These IoT devices have led to significant advances in improving people's lives and contributed to the growth of the technology and communications sectors of the economy [2]. However, simply collecting vast amounts of data from these devices without using them to improve human lives would be of little value. This is where the power of Machine Learning (ML) becomes critical, as it imbues these devices with decision-making capabilities, enabling them to learn from their environment and act accordingly. Traditionally, ML processes have been designed as computationally intensive routines that require large data sets and powerful machines to operate effectively. In the context of ubiquitous IoT devices, such an approach would require transferring massive amounts of data to cloud infrastructures for processing. This approach has several drawbacks, including slower processing times, reduced responsiveness, and the risk that decisions will be made too late to solve the intended problems. A pertinent example of this is monitoring water and air quality in smart cities [3–5]. In these environments, numerous sensors generate large amounts of data at multiple measurement points. Processing this data locally, making timely decisions, and sending only refined information to the cloud can provide multiple benefits. These include reduced energy costs associated with communications, the elimination of the need to encrypt raw data before transmission, and the ability to make immediate decisions in the field. This paradigm, known as edge computing, has received considerable attention in the academic community [6–8]. Despite its benefits, edge

computing presents several challenges, such as limited storage capacity, low processing power, and the need for custom code on specific platforms. An integrated multi-sensor platform called SENSIPLUS Learning Machine (SENSIPLUS-LM) has been developed to address these challenges. This platform consists of both hardware and software components, with the hardware based on an off-the-shelf Micro Controller Unit (MCU) and the SENSIPLUS Chip (SPC). The SPC, developed by Sensichips S.r.l., is a proprietary system capable of performing Electrochemical Impedance Spectroscopy (EIS) and acting as a multi-parameter sensor. The software component of the SENSIPLUS-LM [9] platform includes a toolchain that can generate customized firmware for the MCU. By toolchain, is meant a set of utilities, compilers, and libraries devoted to generating a firmware suitable for the MCU of the C code that implements an ML algorithm. This firmware includes a control library for the SPC as well as ML algorithms designed to bring intelligence to the edge. The capabilities of the SPC have been previously explored in the literature [10, 11]. The toolchain optimizes and automates the processes required to produce AI-based firmware ready for deployment on edge devices, specifically tailored for the SENSIPLUS and the selected MCU platform. It is important to note that not all ML models are suitable for implementation on embedded devices due to their limited computational resources. For example, low-cost MCUs are typically unable to support deep neural networks or other algorithms that require significant memory or processing power. However, the current version of the toolchain includes a selection of algorithms available for C code generation. As discussed later, the modular design of the toolchain's software architecture makes it easy to add new ML models by extending a Java class. While the algorithms in the current version focus on classification tasks, the toolchain can be extended to include regression or anomaly detection algorithms in the future [12]. To demonstrate the effectiveness of the proposed system, the entire toolchain was tested in a surveillance application using a simple Multi-Layer Perceptron (MLP) network. The system's ability to classify pollutants in challenging environments, such as wastewater, was successfully demonstrated, demonstrating the platform's potential for real-world applications.

3.1 Related works

The AI on the Edge paradigm, often referred to as Tiny Machine Learning (TinyML), addresses many of the limitations inherent in cloud-based approaches by providing a complementary solution [13]. TinyML is a rapidly evolving field with numerous challenges that require ongoing research and development [14]. Significant progress has been made by both the academic and industrial communities in adapting ma-

chine learning algorithms to operate on low-cost, low-power devices. This progress has led to developing of several software libraries specifically tailored for TinyML applications [15, 16]. One of the most prominent frameworks in this area is TensorFlow Lite for microcontrollers, developed by Google [17]. TensorFlow Lite consists of two main modules: the converter, which optimizes the model by reducing its size and preparing it for deployment, and the interpreter, which executes the model on embedded devices [18]. This framework has been applied in various contexts, including industrial applications [19] and smart city initiatives [20]. Another essential tool in this area is uTensor, a free integrated environment to facilitate rapid prototyping and implementation of TinyML systems [21]. uTensor includes an inference engine, a data collection tool, and a graphical data processing interface, streamlining the development of edge AI solutions. NanoEdge AI Studio [22] provides a platform for selecting and testing machine learning libraries before final implementation, using an emulator to improve the development process for edge AI applications. Similarly, Edge Impulse provides a service for building cloud-based machine learning models specifically designed for edge devices, with support for AutoML tailored to edge platforms [23]. STM32Cube.AI is another notable tool that supports code generation and optimization for machine learning tasks on STM32 ARM Cortex M-based boards [24]. In addition, the toolkit presented in [25] enables efficient execution of neural network routines on low-cost hardware architectures such as ARM Cortex-M or RISC-V. The literature also highlights open source frameworks for IoT applications that provide tools for high-level data analysis and automatic code generation tailored to specific hardware [26]. These frameworks are critical in driving broader adoption of AI at the edge by making advanced analytics more accessible on resource-constrained devices. Several industrial products are designed specifically for AI at the edge. Notable examples include solutions from STMicroelectronics, such as the SPC5STUDIO [27], and the BME688 software suite from Bosch Sensortec [28]. These products highlight the growing interest and investment in edge AI technologies within the industry. All of these technologies are not automatically compatible with the SENSIPLUS system, and many are closed to their ecosystems and therefore difficult to use. Because of the unique nature of this sensor system, it was necessary to create an ad hoc system to take advantage of all of them.

3.2 Architecture

The toolchain is open source and freely available for download, making it accessible to users and developers alike [29]. This section outlines the main modules that make up the toolchain, as illustrated in Figure 3.1. The primary components of the

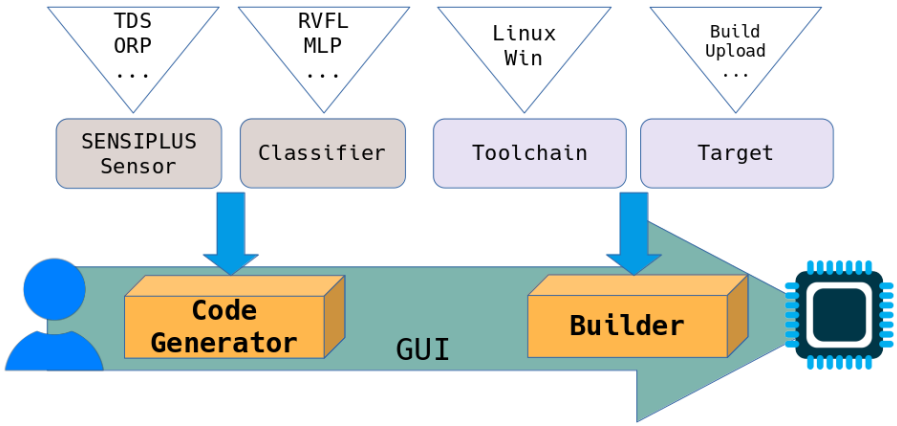


Figure 3.1: Toolchain Architecture.

toolchain are the "Code Generator" and the "Builder". The Code Generator takes the parameters of the selected machine learning model as input. It generates corresponding C code ready for compilation and execution on the target MCU. Once the Code Generator generates the C code, the Builder module compiles this code to produce firmware ready to be loaded onto the MCU. The Code Generator works by using tags within the C code templates and makefiles, allowing the inclusion of necessary elements for specialization. For example, specific tags within the code, as shown in Listing 3.1 and Listing 3.2, serve different purposes. During source generation, the _INIT_CLASSIFIER_ tag is used to insert the code required to initialize the machine learning model, while the _INIT_SENSOR_ tag is associated with initializing the desired sensors. During the build phase, the _OUT_DIR_ tag specifies where the compilation results will be stored. The _HARDWARE_ tag refers to the compilers used in the host system, and the HEX tag refers to a temporary file used during the compilation process. The toolchain is implemented in Java, with some main classes described in the following subsections.

Listing 3.1: Extract of a C Source.

```
/* Completion of the initialization phase
*  where the remaining data structures are associated and filled */
fillConfiguration();
initAllSPMeasurements();
/* SENSIPLUS initialization (involves communication with the chip) */
initChip();
_INIT_CLASSIFIER_
```

_INIT_SENSOR_

Listing 3.2: Extract of a Makefile.

```
# Main-build Target
main-build: generic_classifier
# Tool invocations
generic_classifier: $(HEX)
	@echo 'Building target: $@'
	@echo 'Report:'
	"_HARDWARE_/xtensa-esp32-elf-size" -A "_OUT_DIR_/generic_classifier.elf"
	@echo 'Finished building target: $@'
```

3.2.1 Module for Code Generation

The Code Generator module (Figure 3.2) generates compilable C code based on the selected classifier and sensors. The C code generated by the toolchain is automatically linked to the C APIs available for SENSIPLUS by the Builder module described in the following section. The following is a brief overview of the main classes within the code generator.

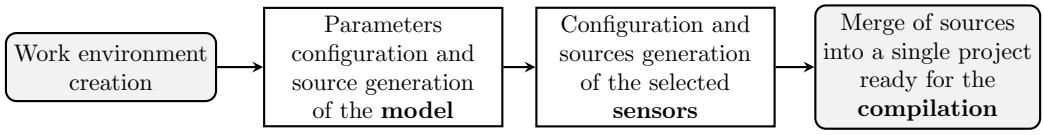


Figure 3.2: Workflow for the Code Generator module.

The "Sensor" class

The abstract Java class "Sensor" has been implemented to represent a generic sensor. The SPC can control two categories of sensors: "direct" and "derived". Direct sensors include temperature or voltage sensors, while derived sensors measure quantities of interest by impedance measurement. The current version of the toolchain already includes several extensions to the Sensor class, including OffChipORP and OffChipSpecificConductivity. To add new sensors, developers must extend the Sensor class and override the methods responsible for generating the C code that calls the SENSIPLUS API for initialization and subsequent measurements.

As a practical illustration, a brief description of the Java classes representing the sensors available in theSmart Cable Water (SCW) configuration is below. SCW is a very compact SPC configuration with a footprint of $12 \times 15\,\mathrm{mm}^2$, using "derived"

sensors consisting of InterDigitated Electrodes (IDEs) connected to the external analog ports of the SPC. The six IDEs on the SCW are functionalized with six different metals: Copper, Gold, Silver, Nickel, Palladium, and Platinum (Figure 3.3). More details about the SCW can be found in section 2.2.1.

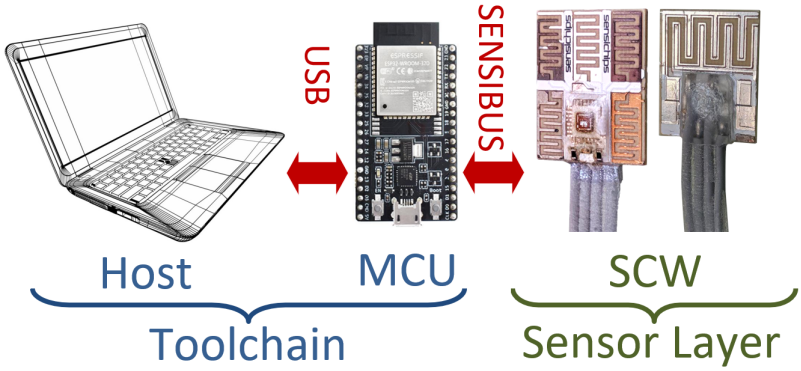


Figure 3.3: A Smart Cable Water (SCW) connected to the MCU through the SENSIBUS, a proprietary one-wire bus adopted by SPC. On the SCW are visible the IDEs in different metals and the SPC.

The following implementations of the Sensor class are available in the current version of the toolchain:

- *OffChipORP*: this class is the C code generator for the oxidation-reduction potential (ORP). The ORP evaluates the tendency of the electrolyte to lose or acquire electrons when the solution changes by injecting a new substance.
- *OffChipSpecificConductivity*: this class is the C code generator for the conductivity measurement expressed in mS/cm, which is a measure of the ability of water to conduct electricity.
- *OffChipPracticalSalinity*: this class is the C code generator for the measure of the salinity in a direct way. The salinity is usually referred to as "practical salinity" and is typically derived from conductivity measurement (not measured directly).

The "Classifier" class

The "Classifier" is an abstract Java class that is a basic abstraction for various classifiers. In the life cycle of a classifier, two primary phases are critical: training and inference. In the current implementation, model training is performed on a

PC or workstation using the Weka library [30]. When training is complete, the trained model is passed as input to this Java class or its extensions, generating the C code necessary to perform inference on the MCU. The current version of the toolchain includes several specialized extensions of the classifier class, such as Multi-Layer Perceptron (MLP) [31], Random Vector Functional Link (RVFL) [32], and Decision Tree [33], among others. New Machine Learning algorithms can be added to the toolchain by extending this Java class. Each subclass is designed to generate the C code for a specific Machine Learning model, using the parameters obtained during the training phase as input. As a case study, the C code generated by the toolchain for the MLP model was tested to evaluate both the inference times and the accuracy achieved on the MCU, comparing these metrics to those obtained using standard libraries on a PC. Details of these evaluations are given in section 3.3.

The "CodeGenerator" class

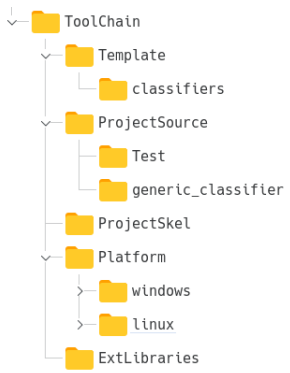
The Java "CodeGenerator" class calls the Java classes for managing sensors and classifiers and returns the C code optimized for the problem under consideration. It also generates the invocation of the appropriate library functions of the SENSIPLUS C API for the MCU. The related C source code is structured as follows:

- code independent from the classification problem at hand. This code initializes the SPC and appropriately manages the resources of the MCU;
- code that implements the inferential engine of the classifier;
- code with calls to sensor control C functions, integrated with code generated by the Sensor's derived classes.

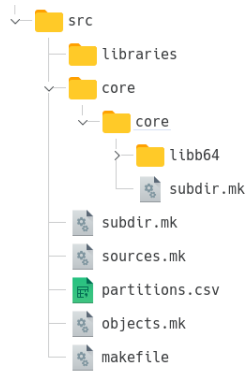
3.2.2 Module for Firmware Generation and Upload

The structure of the presented toolchain is shown in Fig. 3.4a. The first version of the toolchain presented in this work targets a family of processors based on the Xtensa architecture produced by the Espressif company [34]. One of the reasons for this choice was certainly the good ratio quality/performance of these devices [35]. The Arduino has been chosen as a programming model, and the toolchain is cross-platform (can be used on Windows, Linux, or MacOS). Below are the principal components of the toolchain:

- OS-dependent executables that allow compiling, linking, and debugging;
- scripts for communicating with cards hosting a specific MCU, such as the components for loading the firmware in the persistent memory of the devices;



(a) Typical directory structure managed by the toolchain.



(b) Typical makefile structure of a sample project.

Figure 3.4: Directory structure.

- other executables that perform additional tasks but are not strictly necessary for compilation. These include, for example, executables to partition the device memory and load data into it. The specific cases in which they were used will be described below.

Target Class

The Target class implements a sequence of operations into three methods:

- *preExecution*: this step makes all the preliminary activity for the following steps;
- *targetBuildExecution*: this step generates the firmware;
- *postExecution*: this step loads the firmware on the MCU, completing all the toolchain operations.

Several implementations have been derived from these classes to represent different use cases corresponding to the various operations performed on the devices. As mentioned, a partitioning system is essential for loading persistent data such as firmware. Device configurations can be efficiently changed by modifying this memory area, eliminating the need to regenerate the entire firmware. The UloadSpiffTarget Java class is responsible for correctly formatting the available memory, supporting features such as wear leveling, file system consistency checks, and more,

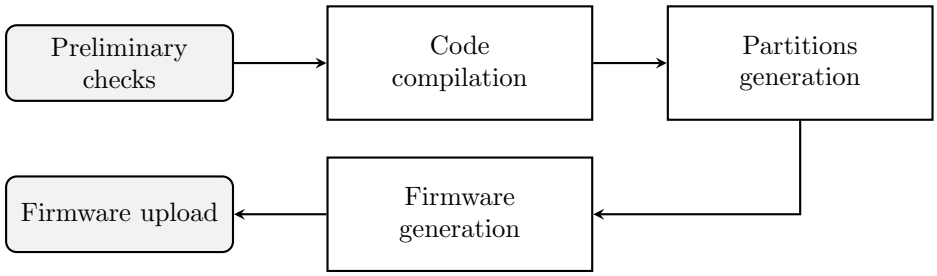


Figure 3.5: The execution model of GenericTarget class.

based on the Serial Peripheral Interface Flash File System (SPIFFS). The UploadFirmwareTarget Java class focuses only on the firmware upload phase to the device memory. Meanwhile, the GenericTarget class implements and manages the compilation process from source code using makefiles (Figure 3.5). Since the build process can be time-consuming, the system uses makefiles to prevent unnecessary builds and optimize the overall process.

3.3 Inference Performance Evaluation

An MLP model was transformed into C language and tested directly on the target device as a case study. Two types of tests were performed on the generated firmware: evaluation of numerical accuracy and inference time, and comparison of accuracy on MCU with accuracy on PC/Workstation. All tests were performed on a board equipped with an ESP32 microcontroller and a floating-point unit: the Wemos D32 pro [36].

3.3.1 Numerical precision and inference time analysis

Table 3.1: MLP tested configurations.

# of neurons input layer	**# of neurons single hidden layer**	**# of neurons output layer**
3	16	6
3	32	6
3	64	6
8	16	6
8	32	6
8	64	6

To evaluate the numerical accuracy and inference time, the MLP has been tested in different configurations regarding the number of neurons in the hidden layer, different activation functions, and different precision of parameter representation. The accuracy and response times have been tested on MLP configurations as shown in the table 3.1. The first test is related to the numerical precision adopted for the model representation, using a 32-bit or 64-bit floating point. The tests show that the result remains the same to the eighth decimal place and varies with the number of bits used. At the same time, the comparison with the reference value obtained on the PC remains unchanged up to the fourth decimal place. The inference time was evaluated with different tests using both floating-point representations. In addition, the impact on the performance of the Softmax activation function 3.1 applied to the output neurons was evaluated. For the other layers, the ReLu activation function 3.2 was always used.

$$P(y = i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.1}$$

$$\text{ReLU} = \max(0, z) \tag{3.2}$$

where:

P(y=i)	is the probability that the input belongs to the i-class.
z_j	is the raw score for the class.
z	represents the input value for the neuron.

The time between input and output, expressed in microseconds (μs), was computed 10000 times for each execution, with randomly generated values applied to the input. The evaluations showed through figures Figure 3.6a, Figure 3.6b, Figure 3.7a highlight the benefit of using a 32-bit floating-point representation. Generally, the inference time for the tested configuration is always less than a millisecond. Finally, the configuration composed of 8 input neurons, 64 in the hidden layer, 6 in outputs, and Softmax, the output activation function, was used to evaluate the current consumption. The Agilent 34401A Digital Multimeter was utilized, highlighting a current consumption equal to 66.4 mA, resulting higher than the idle case (41.8 mA).

3.3.2 Analysis of inference accuracy through a real use case

A model related to a real use case for detecting pollutants in water was used as a final test. In this case, the SCW was used to obtain the measurements. Ten features were collected from sensors based on EIS built on the six IDEs available on the SCW (Table 3.2).

In a previous experimental campaign, a Multi-Layer Perceptron (MLP) with a single hidden neuron layer provided the optimal balance between performance

Table 3.2: Selected features (sensors).

	IDE	Frequency of Acquisition	Physical values
1	PLATINUM	78 kHz	RESISTANCE
2	GOLD	78 kHz	RESISTANCE
3	PLATINUM	200 Hz	RESISTANCE
4	PLATINUM	200 Hz	CAPACITANCE
5	GOLD	200 Hz	RESISTANCE
6	GOLD	200 Hz	CAPACITANCE
7	SILVER	200 Hz	RESISTANCE
8	SILVER	200 Hz	CAPACITANCE
9	NICKEL	200 Hz	RESISTANCE
10	NICKEL	200 Hz	CAPACITANCE

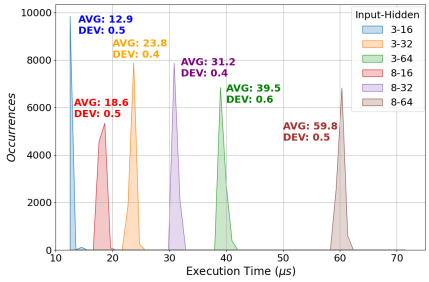
and energy consumption [37]. The model architecture consists of 10 neurons in the input layer, a single hidden layer of 16 neurons, and 13 neurons in the output layer. The activation function used in the output layer is Softmax. The dataset used to train the model includes measurements of 12 different substances and a background measurement WasteWater (WW), and has been made publicly available [38]. The measurement protocol to acquire this dataset was divided into two steps.

- The first 600 samples are acquired in warm-up mode, in this time sensors are exposed to WW only;
- The next 1000 samples are acquired after analyte injection.

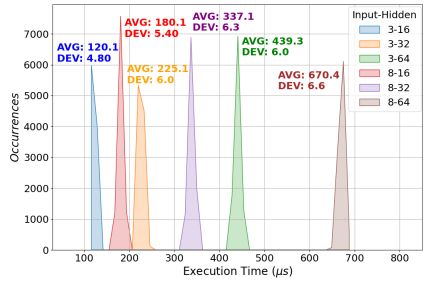
Each acquisition in the study consists of 1,600 samples, with the first 600 samples measured in WasteWater (WW) and the remaining 1,000 samples measured with the analyte mixed with WW. For each compound, 10 acquisitions or experiments were performed. Further details on the substances are given in the Table 3.3. The inference times are shown in Figure 3.7b. Offline validation of the models was performed using the K-fold cross-validation method with K set to 10, resulting in an overall accuracy of 92.75%. The corresponding confusion matrix is shown in Figure 3.8. The results show that the system is particularly effective at identifying certain substances, such as INT_NELSEN and SODIUM_CHLORIDE, which achieved an average accuracy of 99% with a standard deviation of 0.2%. However, the system also confuses certain substances, such as ACETONE and ETHANOL. Of the ten models generated by K-fold cross-validation, the one with the highest accuracy was deployed on the microcontroller using the toolchain system presented in this thesis.

3.4 Conclusions

This paper presents a joint microcontroller, sensing device, and edge intelligence development system called SENSIPLUS-LM. The system includes a multi-sensor microchip equipped with Electrochemical Impedance Spectroscopy (EIS) capabilities and is supported by an open-source end-to-end toolchain. This toolchain facilitates the firmware generation for the microcontroller required to operate the SENSIPLUS microchip, starting from an appropriately trained ML model. Although the toolchain is specifically designed for use with the SENSIPLUS microchip, it is not limited to this application; the target microcontroller can be any Arduino-compliant architecture. The toolchain is released as open-source software and is available online. Detailed information is provided on extending the toolchain, including adding new sensors and ML algorithms. The toolchain extends the SENSIPLUS platform by moving the inference phase of the ML algorithm to the edge, improving local data processing and making it more suitable for use in distributed IoT monitoring systems.

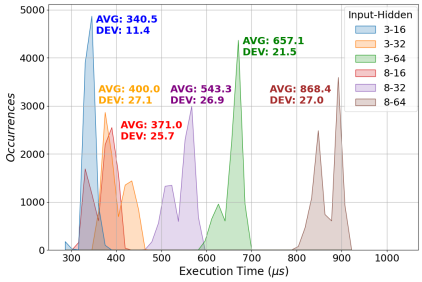


(a) Inference time evaluation with 32-bit floating-point representation.

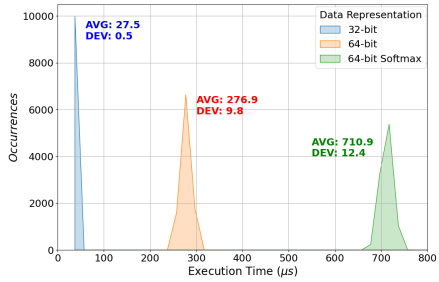


(b) Inference time evaluation with 64-bit floating-point representation.

Figure 3.6: Evaluation of inference time for different MLP configurations. The number of neurons in the input and hidden levels varies while the output is always fixed at six.



(a) Inference time evaluation with 64-bit floating-point representation and Softmax activation function.



(b) Inference times of the proposed real use case. The model is composed of 10 neurons in the input layer, a single hidden layer with a size of 16 neurons, and 13 neurons in the output layer

Figure 3.7: Evaluation of the inference time for different MLP configurations. The number of neurons in the input and hidden layers varies in the first subplots, while the output is always fixed at six. In the second subplot, the performance of the model used in the real use case is highlighted with the number of neurons per layer fixed.

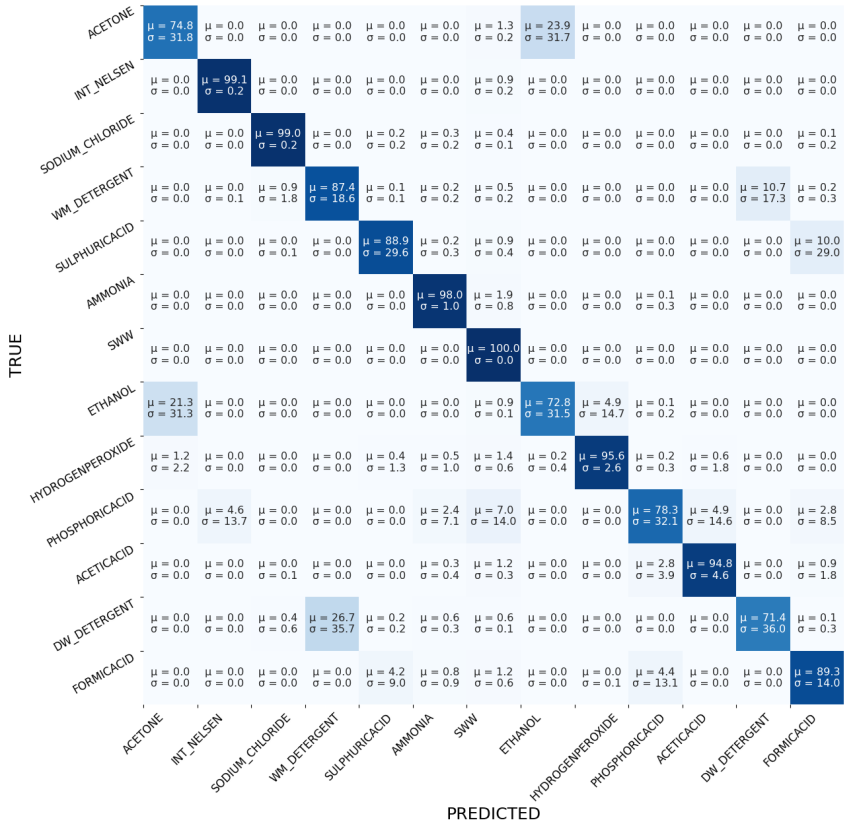


Figure 3.8: Summary confusion matrix, obtained from 10 folds, with mean value and standard deviation.

Table 3.3: Details of the chemical composition of substances. In particular, for each analyte, 10 ml has been inserted in 100 ml of wastewater, the composition of this 10 ml is described in the analyte column.

Substances	**Brief description**	**Analyte composition**	**Concentration in 100ml**
ACETONE	C_3H_6O	10 ml	71727.27 mg/l
INT_NELSEN	Off-The-Shelf product	9.2 ml H_2O + 0.8 ml Dish Detergent	7476.36 mg/l
SODIUM_CHLORIDE	$NaCl$	9.98 ml H_2O + 0.02 g Sodium Chloride	181.81 mg/l
WM_DETERGENT	Off-The-Shelf product	9.2 ml H_2O + 0.8 ml Washing Machine Detergent	7476.36 mg/l
SULPHURICACID	H_2SO_4	9.9ml H_2O + 0.1 ml Sulphuric Acid	1798.20 mg/l
AMMONIA	NH_3	9.7 ml H_2O + 0.3 ml Ammonia 30/33%	2454.54 mg/l
WASTEWATER	Real domestic sewage $pH = 7.4$, conductivity = 1.341 mS		
ETHANOL	C_2H_5OH	10 ml	71727.27 mg/l
HYDROGENPEROXIDE	H_2O_2	8 ml H_2O_2 35%	83703.70 mg/l
PHOSPHORICACID	H_3PO_4	9.8 ml H_2O + 0.2 ml Phosphoric Acid 75%	2327.27 mg/l
ACETICACID	CH_3COOH	9.5 ml H_2O + 0.5 ml Acetic Acid 80%	4772.72 mg/l
DW_DETERGENT	Off-The-Shelf product	9.2 ml H_2O + 0,8 ml Dishwasher Detergent	7476.36 mg/l
FORMICACID	CH_2O_2	9.8 ml H_2O + 0.2 ml Formic Acid 85%	2181.81 mg/l

Chapter 4

In-Water Environmental Monitoring

Environmental monitoring is critical for assessing the health and sustainability of ecosystems as human activities continue to put significant pressure on natural resources. With rapid industrialization, urbanization, and agricultural expansion, the environment is increasingly exposed to harmful pollutants. Therefore, monitoring the quality of air, water, and soil is essential to detect environmental degradation and inform policy decisions to preserve ecosystems and protect public health. Water quality monitoring has emerged as a key area of focus due to its direct impact on human populations and aquatic ecosystems. Freshwater resources essential for drinking, agriculture, and industrial processes are under constant threat from contaminants such as chemicals, heavy metals, and pathogens. Traditional monitoring methods typically rely on manual sampling and laboratory-based analysis, which, while accurate, can be time-consuming and expensive, limiting their scalability. To address these limitations, there is a growing emphasis on Machine Learning (ML) approaches to environmental monitoring, particularly for detecting pollutants in wastewater. Wastewater from households, industry, and agriculture contains a diverse mix of contaminants that, if left untreated, can cause serious damage to ecosystems and public health. ML techniques offer the ability to automate and improve the accuracy of detection and classification of these pollutants, providing real-time analysis and predictive capabilities. By training models on large datasets, these systems can learn to identify patterns, making them particularly effective at distinguishing between various contaminants and detecting anomalies. The effectiveness of ML models in wastewater monitoring is highly dependent on the quality and quantity of the datasets used for training [39]. Large, well-labeled datasets are

essential for these models to achieve high accuracy and generalizability. However, obtaining comprehensive datasets for wastewater monitoring is challenging due to the variability of pollution sources and the dynamic nature of wastewater composition. Publicly available datasets, along with data collected from sensor networks, play a critical role in improving model performance. Robust datasets enable the development of models capable of identifying complex relationships between pollutants and environmental factors, ultimately improving the ability to predict pollution levels and respond to potential environmental threats on time. As a result, part of the activity has focused on building a robust data set.

This chapter first briefly reviews the state of the art in pollutant detection using Machine Learning (ML) techniques, then presents the dataset built in the laboratory used for the subsequent experiments, and finally shows the result of some research activities aimed at building end-to-end systems for pollutant detection in wastewater.

4.1 State of the Art

Wastewater monitoring has become a prominent topic in scientific research, with various technologies contributing to the development of sensors that can detect and classify contaminants to ensure water quality standards. Several systems have been developed for water and air monitoring, with the SENSIPLUS platform being a notable example [40–43]. The monitoring outputs of these systems range from contaminant classification to binary contaminant detection.

Solutions that address specific problems are often preferred over more general monitoring systems that aim to work in different environments. For example, [44] presents a system that detects pollutants in the WasteWater (WW) environment, but it lacks discrimination between specific substances and uses technologies that are now considered obsolete. On the other hand, [45] uses infrared cameras to monitor illegal connections within the sewer system, focusing on a specific challenge rather than broad-spectrum detection.

Another approach was presented by Hyon Wook Ji et al. [46] who developed an image processing system that can estimate wastewater concentrations without distinguishing between substances. While cameras avoid sensor corrosion problems, the system's high energy requirements prevent it from being considered low-cost. High classification accuracy has been achieved in other systems. However, these often neglect energy and cost constraints, as in the case of [47], who developed a system for detecting ammonium and total nitrogen. This system is based on a broader design to identify all nitrogen-derived components.

Regarding practical applications, [48] offers a portable device for monitoring

sewer pumping stations, generating alarms when anomalies are detected. However, this system lacks pollutant classification capabilities.

Regarding processing techniques, the section 4.4 presents a first attempt to apply Natural Language Processing (NLP) techniques, specifically causal models developed for natural language generation, to the task of wastewater pollution detection. Despite this novelty, there are examples in the scientific literature of NLP models being used for unconventional tasks.

For example, language models have been applied in the medical domain for the classification of diagnostic tests through "reverse encoding" techniques [49–51], as well as for the encoding of diagnostic rules [52].

These models have also been adapted for predicting human mobility [53, 54]. More broadly, transformer-based models originally designed for NLP tasks have been successfully applied to various domains, including image processing [55–57], video analysis [58, 59], speech and audio recognition [60, 61], dialogue systems [62, 63], recommender systems [64, 65], reinforcement learning [66, 67], graph-based tasks [68, 69], protein structure prediction [70, 71], autonomous driving [72, 73], and anomaly detection [74, 75].

4.2 A dataset for detecting pollutants in wastewater using low-cost, real-time sensors

At the heart of the data collection system is the Smart Cable Water (SCW), a multi-sensor microsystem designed to detect toxic chemicals, contaminants, hydrocarbons, and organics in water 2.2.1. Its core component is SENSIPLUS, a 1.5 × 1.5 mm, 1.5 mW chip from the Sensichips platform that uses Electrochemical Impedance Spectroscopy (EIS) and a potentiostat to analyze chemical reactions. The SCW has five small InterDigitated Electrodes (IDEs)on the front and one larger one on the back, made of different metalized materials (gold, copper, nickel, silver, platinum). Measurements are taken with the EIS technique at specific frequencies (78125 Hz and 200 Hz) for certain electrodes to evaluate different effects based on the interaction between pollutants and sensors. The system is protected by a transparent resin that allows it to operate underwater.

4.2.1 Measurements setup and adopted sensors

The measurement setup shown in Figure 4.1, consists of the measurement chain (SCW, MCU and PC) together with a magnetic stirrer and a 300 ml beaker containing the analyte. A magnetic stirrer with a 25 mm stirring rod simulates water movement, and all measurements are performed under uniform stirring conditions. The stirrer operates at 50 rpm, a speed at which 100 ml of water avoids the incorporation of air bubbles that could introduce noise into the measurements, while still accurately mimicking typical wastewater flow conditions. The beaker, stirring rod, and sensors are thoroughly cleaned between each test to ensure that each measurement remains independent. All equipment is washed with soap and fresh water, and in cases where wastewater residues are stubborn, the beaker is rigorously cleaned with concentrated 95% sulfuric acid. Each sample acquired consists of 12 values corresponding to the resistance and capacitance of the InterDigitated Electrodes (IDEs) measured at two different stimulus frequencies: 200 Hz and 78 kHz. These frequencies were chosen to capture the different response characteristics of the sensors. At the lower frequency (200 Hz), both resistance and capacitance are measured using five different IDEs: Platinum, Gold, Silver, Nickel, and Copper. At the higher frequency (78 kHz), resistance is measured using only the gold and platinum IDEs. The selected frequencies are based on the Randles equivalent circuit model applicable to IDE electrodes immersed in wastewater.

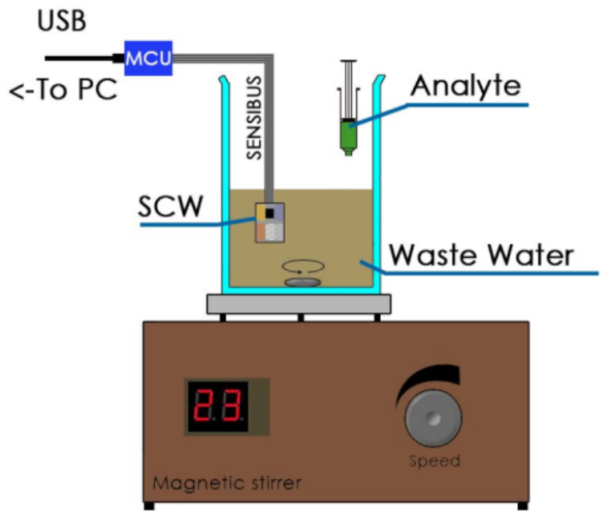


Figure 4.1: Measurement setup

4.2.2 The Dataset

All data acquisitions were performed between 2020 and 2022 using different sensors in two different laboratories, located in Poland and Italy, and the dataset is publicly available in [76]. The dataset includes measurements of 10 different substances in addition to the WasteWater (WW) background, forming a comprehensive dataset. The measurement protocol is divided into two distinct phases:

- In the first phase, the first 600 samples are collected during a warm-up period in which the sensors are exposed only to WasteWater (WW)).
- In the second phase, the next 1000 samples are collected after injecting the analyte into the wastewater.

As a result, each data acquisition contains a total of 1600 samples, with the first 600 representing measurements in pure WW and the remaining 1000 capturing the response after mixing the analyte with the WW. It's important to note that no analyte was injected in the 10 separate WW acquisitions, so all 1600 samples in these cases reflect measurements in pure WW.

Each sample acquisition takes approximately 1.6 seconds, resulting in a total cycle time of approximately 40 minutes. The complete dataset includes 10 measurement cycles for each of the 11 substances (10 contaminants plus WW), all collected according to the above protocol. The complete list of substances with

Table 4.1: Details of chemical composition of substances

	Substances	**Brief description**
1	WASTEWATER	$pH = 7.4$, conductivity = 1.341 mS
2	ACETICACID	CH_3COOH
3	ACETONE	C_3H_6O
4	AMMONIA	NH_3
5	ETHANOL	C_2H_5OH
6	FORMICACID	CH_2O_2
7	HYDROCHLORICACID	HCl
8	HYDROGENPEROXIDE	H_2O_2
9	PHOSPHORICACID	H_3PO_4
10	SODIUM_HYPOCHLORITE	$NaClO$
11	SULPHURICACID	H_2SO_4

their composition is shown in Table 4.1, and the measured quantities are shown in Table 4.2.

Figure 4.2 shows an example of the signals acquired for different substances (e.g., acetone, hydrochloric acid, etc.) over both stimulus frequencies (200 Hz and 78 kHz) using the Platinum IDE. The transition point in each acquisition represents the time of analyte injection.

Some key observations about the temporal behavior of these measurements include

- There is often variability in the baseline (the starting point of the signal curve) between different acquisitions, due to factors such as sensor poisoning, aging, and inherent differences between sensors.
- The shape of the signal curve following the injection event can vary significantly from experiment to experiment.
- It is important to note that the curve should not be interpreted as a conventional "time series", as the post-injection slope and overall shape are strongly influenced by the injection rate, which can vary considerably in real-world scenarios.

It is important to note that water quality is usually monitored by laboratory analyses performed by experienced professionals using complex and expensive equipment. With this kind of approach, due to the cost and time required for water quality analysis, it is quite difficult to prevent environmental disasters. To improve the efficiency of prevention activities, it's necessary to have an effective and reactive

Table 4.2: Aquired physical quantities.

	IDE	Frequency of Acqusition	Physical values
1	PLATINUM	78kHz	RESISTANCE
2	GOLD	78kHz	RESISTANCE
3	PLATINUM	200Hz	RESISTANCE
4	PLATINUM	200Hz	CAPACITANCE
5	GOLD	200Hz	RESISTANCE
6	GOLD	200Hz	CAPACITANCE
7	COPPER	200Hz	RESISTANCE
8	COPPER	200Hz	CAPACITANCE
9	SILVER	200Hz	RESISTANCE
10	SILVER	200Hz	CAPACITANCE
11	NICKEL	200Hz	RESISTANCE
12	NICKEL	200Hz	CAPACITANCE

water analysis through many distributed measurement systems. In this context, low-cost microsensors, which can combine good measurement accuracy as well as good reliability, represent one of the best solution. One of the main advantages of this kind of system is the possibility to use the paradigm of the Internet of Things (IoT) and consequently the application of Artificial Intelligence (AI) techniques to analyze and exploit the information in the generated data. In this sense, the system presented in this work perfectly meets the requirements that a low-cost microsensor system should have to build an effective and reactive distributed measurement system. Finally, one of the main objectives of the presented dataset is to improve the benefit, in terms of accuracy and reliability, of the application of Artificial Intelligence techniques in the field of water quality analysis.

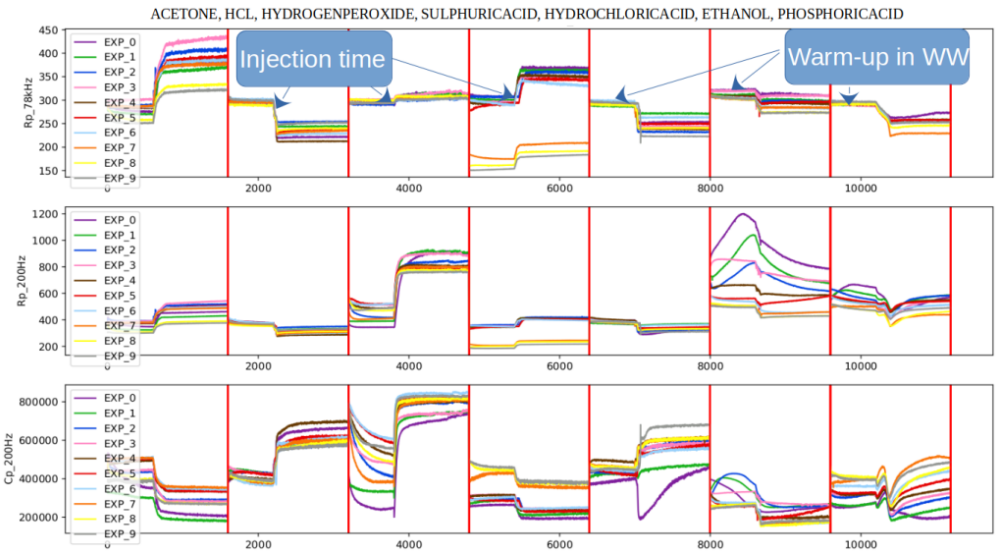


Figure 4.2: Example of acquired signals.

4.3 An IoT-based platform for real-time wastewater pollutants spilling recognition

A review of the scientific literature on applying Machine Learning (ML) to water analysis reveals that the anomaly detection problem is neglected. Not considering anomalies in real systems means that they are unusable in a context other than the laboratory, as the system would not be able to react correctly to substances that were not considered during the training phase, potentially generating false positives. In summary, the open problems in wastewater analysis are mainly related to the often complex and expensive equipment unsuitable for the IoT and pervasive paradigm, and the lack of anomaly and an anomaly detection step. This paper proposes a solution that addresses both of these issues. In terms of IoT readiness, it proposes the adoption of the SENSIPLUS chip, a proprietary device developed by the Italian company Sensichips s.r.l. (presented in chapter 2), which has been proven in reliable measurements for detecting pollutants in air and water [77]. The SENSIPLUS chip, combined with a commercially available Micro Controller Unit (MCU), provides a low-power, low-cost and IoT-ready miniaturized sensor platform. The MCU is required to run the C API provided with the SENSIPLUS chip and to enable the system to communicate with external systems (e.g., via USB or MQTT over TCP/IP) and for the inference phase of various machine

learning algorithms. The second problem is tackled with a two-stage classification system: an anomaly detector and a multiclass classifier, based on the idea that some pollutants are interesting while others simply interfere and do not need to be classified. The anomaly detector makes it possible to determine whether the analyzed substance can be one of interest or something else (for simplicity, unknown). Whenever such a module declares that the substance is not an anomaly, the multiclass classifier module is activated and its computational load is included in the system load. Combining both modules significantly reduces false positives while maintaining high accuracy for the substances of interest. The combination of the developed platform and the new concept of supervised two-stage classification represents the main of this work to the state of the art.

4.3.1 Metodology

The detection chain

Figure 4.3 shows the overall detection system based on the SCW visible inFigure 4.4a, a proprietary IoT-ready smart sensor system by Sensichips s.r.l. composed of InterDigitated Electrodes (IDEs) and based on SENSIPLUS [78]. The latter is a tiny analytical sensor platform with 1.5 mW power consumption and communication capabilities such as SPI, I^2C and SENSIBUS (a proprietary 1-wire communication protocol). The SENSIPLUS requires an MCU to run its C API, which includes the machine learning algorithm inference engine. The ESP32/ESP8266 with USB and WiFi communication capabilities was selected as the MCU. The ESP32/ESP8266 can also guarantee data transfer via (for example) MQTT on TCP/IP. In this configuration, the MCU can act as a simple bridge to transmit the data collected by the sensors to the cloud (e.g. via MQTT), and as a device that performs local processing to detect substances of interest by executing appropriate machine learning algorithms. During operation, the SCW can be submerged in water, and communication and control signals are transmitted through a suitable cable. SENSIPLUS is a microchip capable of interrogating on-chip and off-chip sensors with its versatile and accurate Electrochemical Impedance Spectroscopy (EIS) in the frequency range between 3.1 mHz and 1.2 MHz.

With SENSIPLUS, it is possible to perform measurements with multiple sensors; in particular, the SCW system has 6 IDEs. The physical principle of detecting and identifying a given set of substances is related to the RedOx dynamics of catalytic precious metals. Such a phenomenon can be observed as an electrical behavior. Figure 4.4b shows the modeled electrical equivalent circuit of two electrodes immersed in a water solution, known in the literature as the Randles circuit. As seen from the electrical circuit, each electrode is modeled by a double-layer capac-

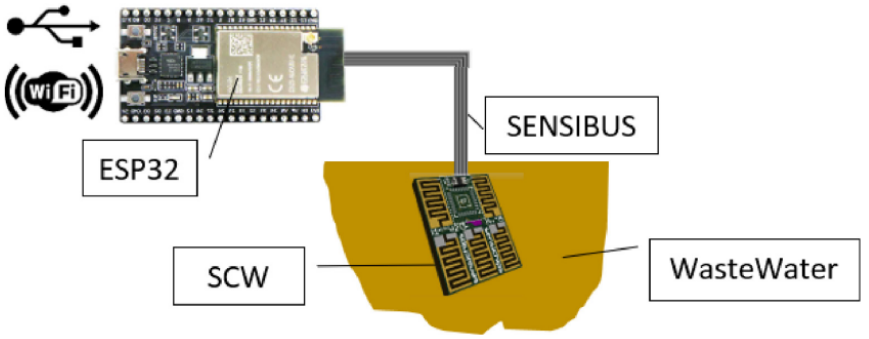
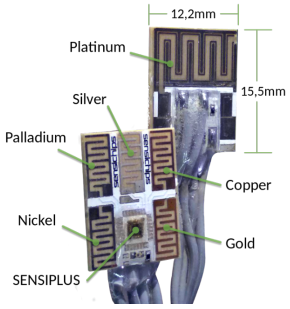
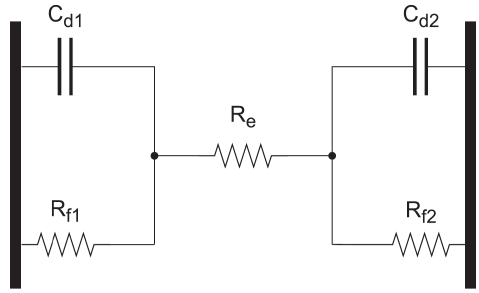


Figure 4.3: The overall detection system deployed

itance C_d and a faradic resistance R_f, which considers the interface between the water solution (called bulk) and the electrode itself. The model values depend on the electrode composition, geometry, bulk composition, etc. The parameter R_e is the equivalent resistance of the bulk, so it depends mainly on the bulk composition and the electrode area. To maximize the sensitivity to the substances of interest and the RedOx dynamics, the 6 IDEs of the SCW were functionalized by coating them with six different metals: (M1) gold, (M2) copper, (M3) silver, (M4) nickel, (M5) palladium, and (M6) platinum. (M1) through (M5) IDEs are each 3 mm by 7 mm, while (M6) is 12 mm by 8 mm (see Figure Figure 4.4a).



(a) Smart Cable Water (SCW) with InterDigitated Electrodes functionalized by coating them with six different metals



(b) Randles equivalent circuit

Figure 4.4: Smart Cable Water

Dataset acquisition for training

The proposed system is intended to detect and recognize substances spilled in wastewater. Consequently, the best solution to build a good dataset for the training phase would be to collect all the measurements directly in a controlled drain of a sewer network. However, this is not a viable solution for two main reasons:

- Measurement point of view: all measurements should be taken from the same and reliable conditions; however, due to the typical background environmental composition of sewage, achieving an acceptable level of reliability conditions is impossible.
- Health point of view: due to viruses, bacteria, etc, and other hazards, operating directly in the sewer system would constitute a biological hazard.

To solve the above problems, the Synthetic WasteWater (SWW) was created to simulate the wastewater composition and a measurement setup as described in Figure 4.5 to generate an appropriate data set. The assumed recipe for the SWW is inspired by a simplified version developed in [79]. In addition, to better reproduce a real wastewater scenario, the pH of each batch of the SWW was corrected according to [80], where measurements on the real wastewater are reported. The detailed chemical composition of the SWW is shown in Table 4.3. Fourteen substances were spilled in the SWW background: (1) acetic acid; (2) acetone; (3) ethanol; (4) ammonia; (5) formic acid; (6) phosphoric acid; (7) sulfuric acid; (8) hydrogen peroxide; (9) synthetic sewage; (10) sodium hypochlorite; (11) sodium chloride; (12) dishwashing detergent; (13) laundry detergent; (14) Nelsen. The listed substances can be divided into groups 1-9 (Group 1) and 10-14 (Group 2). Group 1 contains only substances that the system should recognize, while Group 2 contains only the outlier samples that the system should be able to reject.

The measurement procedure for each substance to create a data set consists of two phases:

- *Warm-up phase:* to allow all sensors to stabilize, 600 samples are acquired at a rate of 0.5 Hz (total warm-up time: 900 s) in pure SWW.
- *Measurement phase:* after the first 600 samples, the substance of interest is injected into the SWW, and to record the entire evolution of the sensor after the injection, another 1000 samples are acquired at the same sampling rate (total measurement phase time: 2000 s).
- Resistance measured at a frequency of 78 kHz, for the gold and platinum IDEs.

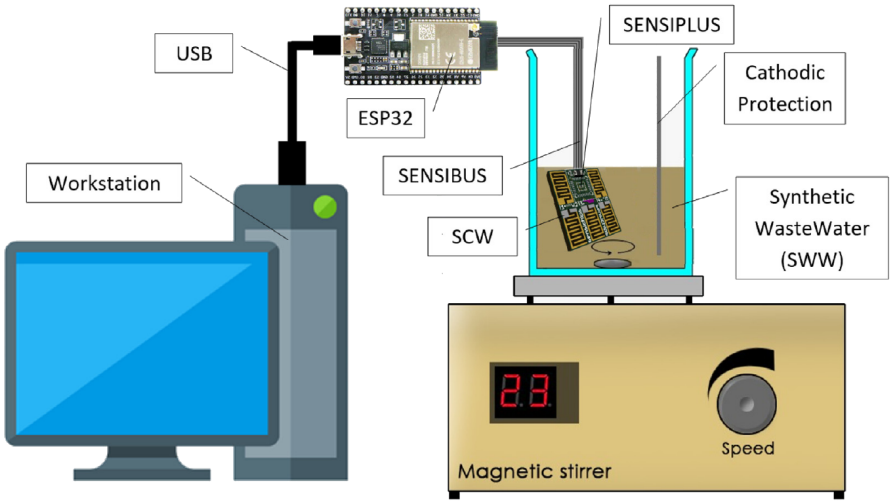


Figure 4.5: Measurement Set-Up for dataset acquisition

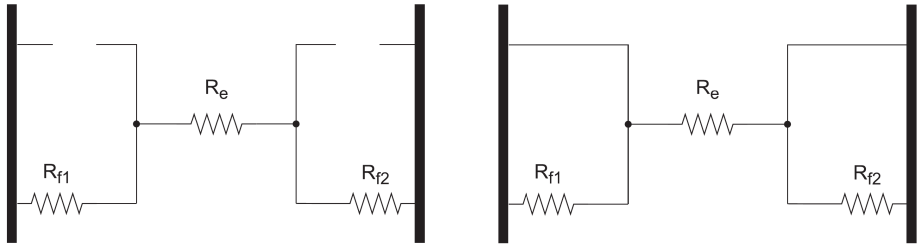
- Resistance and capacitance were measured at a frequency of 200 Hz for gold, platinum, silver, and nickel.

Table 4.3: Synthetic waste water chemical composition.

Compounds	**Concentration [mg/l]**
Fertilizer	91.74
Ammonium Chloride	12.75
Sodium Acetate Trihydrate	131.64
Magnesium Hydrogen Phosphate Trihydrate	29.02
Monopotassium Phosphate	23.4
Iron (II) Sulfate Heptahydrate	5.80
Starch	122.00
Milk Powder	116.19
Yeast	52.24
Soy Oil	29.02

The obtained dataset has been made publicly available here [81]. One of the main problems related to Machine Learning (ML) is feature identification, i.e., selecting informative features derived from sensors that maximize the classification accuracy. In this case, the electrical equivalent circuit described in the previous section was chosen to record the following features

To obtain a feature vector of size ten (6 resistance and 4 capacitance). The palladium and copper IDEs were not used in this experimental campaign. These features were chosen because of the different behavior of the equivalent circuit at low and high frequencies. In particular, both C_d exhibit high impedance at low frequencies and can be represented as an open circuit (see Figure 4.6a). Thus, the measurements depend on the faradic or the bulk resistance (R_e). On the other hand, at high frequencies, the two C_d have a low impedance and can be represented as a short circuit (see Figure 4.6b): the measurements depend mainly on the bulk resistance.



(a) Randless a low frequency (b) Randles at high frequency

Figure 4.6: Randles at different frequencies

Dataset structure and usage

For each substance, ten acquisitions of 1600 samples obtained by the above measurement procedure were collected, resulting in 16000 samples. The k-fold cross-validation procedure was used for evaluation. Cross-validation is primarily used in applied Machine Learning (ML) to estimate the skill of a machine learning model on unseen data. Its use generally results in a less biased or optimistic estimate of model performance than other methods, such as a simple train/test split. Typically, the first step in k-fold cross-validation is to shuffle the collected data randomly, but in this case, given that measures belonging to the same experiment are highly correlated, was chosen as the unit for k-fold, assuming a full acquisition (1600 samples) of all compounds.

To find the best anomaly detection and multiclass classifier model to use for the whole system, the whole dataset was organized into ten folds *(Fold 0, Fold 1, . . ., Fold 9)*. Each fold contains nine additional splits *(Split 0, Split 1, . . ., Split 9)* and one test.

The given splits are organized as follows:

- *Training data:* used to train both anomaly detection and multiclass classifier model.
- *Test data:* used to find the best model's hyperparameters for anomaly detection and multiclass classifier.

The final evaluation consists of the samples that are not included in either the training data or the test data of all the splits related to the given fold. To keep things clear was used a fixed nomenclature: the number within the name of the given Fold indicates the experiment (data collection) used to perform the final evaluation. In contrast, the number within the given Split indicates the experiment used for the related Test data. The training data is composed of all the experiments except the one used for the related test set and the one used for the final evaluation, which, as mentioned before, contains data that are not visible from both the training and the test data of the related fold. For example, the Fold0 contains the Split from 1 to 9, excluding the Split 0, since the Experiment 0 of all substances is used to build the related Test Set. The test data of split 1 will consist of experiment 1 of all substances. In contrast, the training data will consist of all remaining experiments (except experiment 1 used for the test, and experiment 0 used for the final evaluation). In this way, the test data of Split 2 will comprise experiment 2, while the corresponding training data will exclude experiments 2 and 0, and so on. The final evaluation of Fold 0 is made by Experiment 0 of all substances. See Figure 4.7 for a graphical representation of the data set splitting. It is worth specifying that in Figure 4.7, Exp 0, Exp 1, ..., Exp 9 means acquisition 0, 1, ..., 9 of all substances. Finally, it is worth specifying that regarding the multiclass classifier and the anomaly detection model, the training, test, and final evaluation set ratios during the learning phase were respectively: 80%; 10%; and 10%. Furthermore, to properly validate and test the learned anomaly detection models, the validation and test sets were contaminated with outlier points taken from the substances belonging to Group 2.

4.3.2 Classification

The classification system is organized in two phases: (i) Data Preprocessing; and (ii) Classification. As can be seen inFigure 4.8, the Data Preprocessing phase (i) normalizes the raw data coming from sensors and discriminates through a Finite State Machine (FSM, see Figure 4.9) if there should be submitted to the Classification phase (ii) or not.

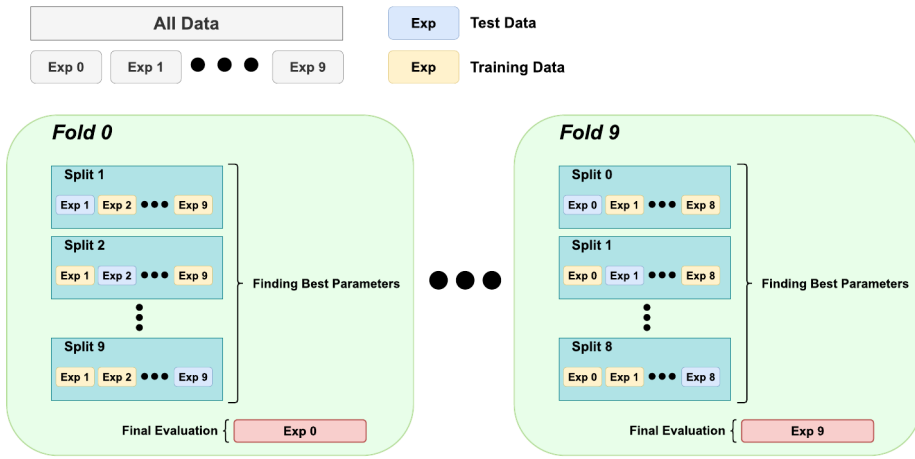


Figure 4.7: Data Set Structure. Exp 0, Exp 1, ..., Exp 9 means respectively acquisition 0, 1, ..., 9 of all substances

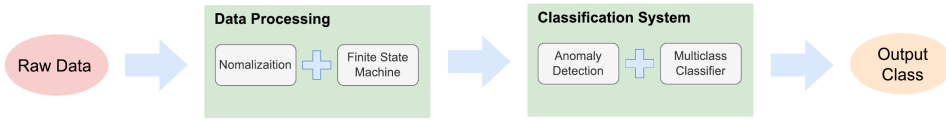
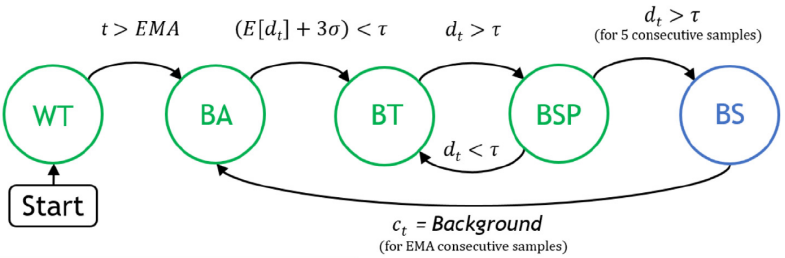


Figure 4.8: An overall view of the system



Symbols	
t	time sequence
d_t	distance of s_t from b_t
EMA	Exponential Moving Average coefficient = 25
τ	the threshold (0.05)
c_t	classification of sample t
$E[d_t]$	the mean over last EMA
σ	the std dev over last EMA sample

States		Output
WT	WaiT	BKG
BA	Baseline Acquisition	BKG
BT	Baseline Tracking	BKG
BSP	Baseline SusPended	BKG
BS	Baseline Stopped	The sample is forwarded to the classification module

Figure 4.9: Finite state machine

Data preprocessing

The Data Preprocessing phase is realized in two steps:

- normalization of the raw data coming from sensors, through the creation of a robust baseline signal;
- decides if the normalized sample should be forwarded to the anomaly detector or classified directly.

The baseline signal $\boldsymbol{b_t}$ is generated by the union of the FSM with the application of an Exponential Moving Average (EMA) according to the following equation:

$$\boldsymbol{b_t} = \begin{cases} \boldsymbol{s_t} & t = 0 \\ \boldsymbol{b_{t-1}} & t > 0, S \in \{BS, BSP\} \\ \alpha \boldsymbol{s_t} + (1-\alpha) \cdot \boldsymbol{s_{t-1}}, & t > 0, S \in \{WT, BA, BT\} \end{cases} \tag{4.1}$$

where s_t are the sensors' raw data, Wait (WT), Baseline Acquisition (BA), Baseline Tracking (BT), Baseline Suspended (BSP), and Baseline Stopped (BS) are the states of the FSM.
The FSM aims to build a robust baseline capable of coping with: the variability

between sensors/chips, the sensor drift, the environmental noise, interferences, etc. The first two states (WT and BA) guarantee that noise and/or interferences do not affect the baseline. Once the FSM reaches the BT state, the system tries to detect the injection of a substance, revealed by a peak in the raw data concerning the baseline generated with EWA (see below).

The BSP state is an intermediate state between BT and BS that tries to filter out signal spikes by waiting if a cluster of samples confirms the presence of a spilled substance. Once the system is confident there is the spilling of a substance (after 5 acquisitions), the FSM moves to the BS states. Consequently, the normalized samples are passed as input to the classification phase algorithm.
Regarding the EMA, the α parameter is the reciprocal of $\mathrm{EMA_c}$ (coefficient empirically set to 25). The following formula gives the normalization value:

$$\boldsymbol{f}_t = \boldsymbol{s}_t / \boldsymbol{b}_t \tag{4.2}$$

where the $\boldsymbol{f}_t$ is the normalized feature vector, while $\boldsymbol{s}_t$ is the raw sensor data and the $\boldsymbol{b}_t$ is the baseline signal computed as described by the Equation 4.1.

Figure 4.21 shows the entire FSM system. In particular, t is the current time sample, while τ is a threshold that, in our case, has been empirically set equal to 0.05. Regarding the $\boldsymbol{d}_t$ parameters, it represents the Euclidean distance between the normalized features vector f_t and the unit vector $\boldsymbol{u}$ (a vector of ones) in a 10-dimensional space that is the size of the vector $\boldsymbol{s}_t$ (see Equation 4.3).

Starting from the Euclidean distance $\boldsymbol{d}_t$ evaluated between $\boldsymbol{s}_t$ and $\boldsymbol{b}_t$ in the feature space, there is a peak that reveals an injection when $\boldsymbol{d}_t$ is greater than a threshold τ (empirically established to 0.05).

Looking at the Equation 4.2, it is clear that the vector $\boldsymbol{f}_t$, when $\boldsymbol{b}_t$ is equal to $\boldsymbol{s}_t$ is equal to the unit vector. For this reason, the Euclidean distance has been computed concerning the unit vector and so when $\boldsymbol{d}_t$ is equal to zero means that the baseline signal $\boldsymbol{b}_t$ is perfectly tracking the sensors signals s_t.

$$\boldsymbol{d}_t = |\boldsymbol{f}_t - \boldsymbol{u}| \tag{4.3}$$

As seen in Figure 4.21, the current state of the FSM can change according to a given rule. In particular, the FSM starts with the WT state. In this state, the classification system will simply compute and store into a vector the first EMA_c distance computed over the $\boldsymbol{s}_t$ measured samples, as reported in the Equation 4.2. Once the distance vector has been filled, the FSM can pass in the BA state. Here the system will keep updating the distance vector and, once the variability of the vector (computed as the mean plus three times the standard deviation) is below a given threshold, the system can move to the next state. At this point, the system

will check if a substance has been spilled in the water, and this is done by checking when the current distance is major of a given threshold. Once the FSM moves to the BSP state, to not confuse the spill of a substance with a measurement spike or simple noise, the system will check that the current distance remains above the threshold for five consecutive samples (BSP). Otherwise, the system will return to the BT state. Finally, once the FSM is in BS state, the current normalized samples are inputted to the detection system. If the sample classification equals the background substance in this state, the FSM will return to the BA state.

Figure 4.10 shows the entire system depicted so far.

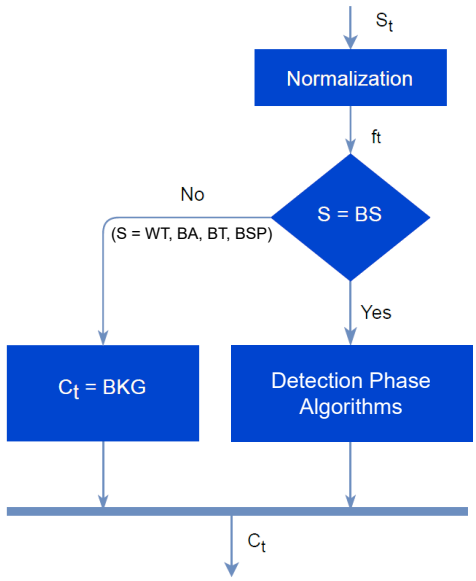


Figure 4.10: Finite State Machine Flow Chart

Where S indicates the state of the FSM, C_t is the classification of the sample at time t, and BKG is the background substance.

Detection Phase

In a real scenario, there are plenty of substances that flow in the sewerage network, therefore it is crucial to be able to distinguish between the substances of interest and the other ones.
In this sense, the main goal of this phase is to determine if the given flowing substance is one of the substances of interest, to be then able to predict its name

correctly.
The detection phase is divided into two main parts:

- Anomaly Detection
- Multiclass Classification

Anomaly Detection
Regards the anomaly detection algorithms, we can mainly distinguish them in two approaches:

- Outlier Detection
- Novelty Detection

In the outlier detection algorithms, the training data contains outlier samples. In this case, the estimators try to fit the regions where the training data is the most concentrated, ignoring the deviant observations. The training data is not polluted with outlier samples in the novelty detection algorithms. In this context, we want to determine whether a new observation is an outlier. In this sense, an outlier is also called a novelty.
Our case is better represented by the novelty detection approaches according to our data set and the application field. This is because, in our application field, we want to discard all those substances that are usually present in the sewage system, and we want to recognize only the substances of interest that represent a minimum part of the substances that can be found in wastewater. To have as complete a point of view as possible, we trained and tested anomaly detection models built with either novelty or outlier approaches:

- Novelty Detection: One-class SVM, Local Outlier Factor, and KNN
- Outlier Detection: Elliptic Envelope and Isolation Forest

All the algorithms have been taken from the sci-kit learn library except for the KNN, which has been taken from the Python Outlier Detection (PyOD) library [82]. As described in Section 4.3.1, we have divided the entire data set into ten cross-validation folders, each of which contains additional nine folders containing training and a validation set. For the outlier detection data set concern, it is the same depicted in Section 4.3.1 with the addition of some outlier samples in the training set (about 10%).

Multiclass Classification

Starting with the results obtained in previous work, we have trained and optimized the accuracy of a KNN on the described data set. It is noted that, unlike anomaly detection, the training and validation sets are formed with only the samples of the substances of interest.
In both cases, anomaly detection and multiclass classification, the grid search approaches have been chosen to optimize the models' accuracy. All the models parameters are detailed in the Table 4.4 and Table 4.5.
Finally, the entire system, composed of the data processing and the detection systems, is shown in Figure 4.11.

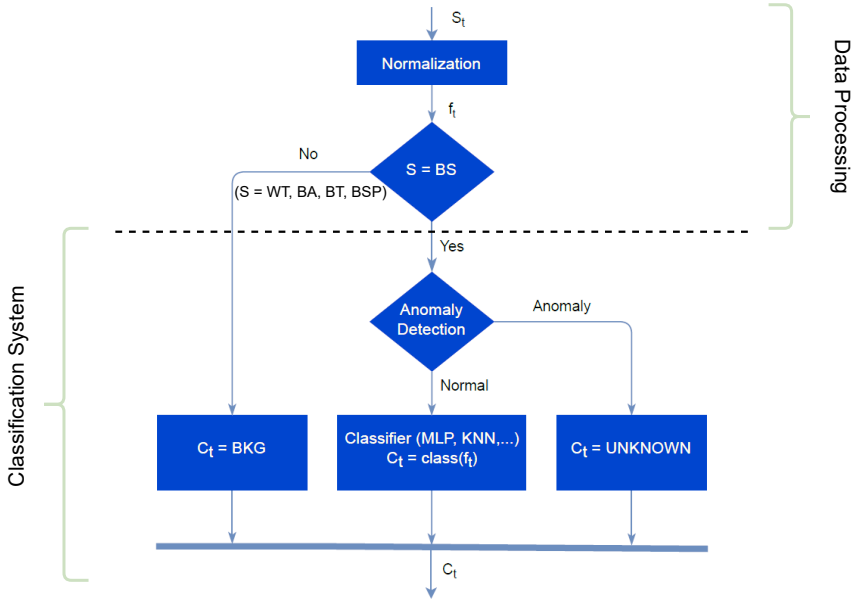


Figure 4.11: Entire System Flow Chart

Eventually, to find out the best anomaly and multiclass classifier model, the cross-validation technique over the ten sub-sets of the data set (see Section 4.3.1 for more details) has been used. Once the best model of each classifier has been found, the entire system will be tested based on the test data.
It is important to point out that the proposed detection system does not relay over any pattern/trajectory recognition, or time series, or, in other words, it is time-independent. This feature allows us to build an IoT-ready system capable of detecting and recognizing the given spilled substance based only on the current

Table 4.4: Anomaly Detection Models Parameters

Classifier	Parameters	
KNN	contamination	[0.01, 0.05, 0.1, ..., 0.5]
	N neighbors	[10, 100, 200, ..., 500]
SVM	ν	[0.01, 0.05, 0.15, ..., 1.0]
	Kernel	Radial basis function
	γ	[auto, scale, 0.01, 0.05, 0.15, ..., 1.0]
Local Outlier Factor	contamination	[0.01, 0.05, 0.1, ..., 0.5]
	N neighbors	[10, 100, 200, ..., 500]
Elliptic Envelope	contamination	[0.01, 0.05, 0.1, ..., 0.5]
Isolation Forest	contamination	[auto, 0.01, 0.05, 0.1, ..., 0.5]
	N estimators	[50, 100, 150, ..., 500]

Table 4.5: Multiclass Classification Model Parameters

Classifier	Parameters	
KNN	algorithm	ball tree
	N neighbors	[10, 100, 150, ..., 500]
	weights	[uniform, distance]

samples, as shown in Figure 4.11. In this sense, we can refer to our system as an IoT-ready platform for real-time pollutant spilling detection.

The Algorithm 2 and 1 shows the pseudo code regarding the Training and Test procedure. As described in the previous sections, the training procedure is the same for anomaly detection and multiclass classifiers. For the test procedure concern, instead, it has been built to be able to test the entire system (anomaly detection and multiclass classifier) rather than only the multiclass classifier one. For that reason, the Test procedure takes as input an extra parameter "*doAnomaly*" that serves to decide if the test must be performed over only the multiclass model (case *doAnomaly* = FALSE) or on both anomaly detection and multiclass models (case *doAnomaly* = TRUE). In the latter case, the online classification procedure (Algorithm 3) is called. It is worth specifying that the Algorithm 3 represents the procedure implemented on the end-to-end system to perform online tests of the entire system. The time complexity of the overall chain is, in the worst case, the sum of the time complexity of a kNN with an SVM at inference time that is compatible with the computational capability of the selected MCU [83]. The time complexity of the kNN algorithm is $O(n * d)$, where n is the number of samples

Algorithm 1 Training procedure

input: A dataset $\mathcal{F}$ representing a single Fold, list of classifiers to train, hyperparameters
output: Best Classifier
begin

$\mathcal{F}_n = normalizeDataSet(\mathcal{F})$;
for clf in $classifiers$ **do**
  $X_{train}, Y_{train} = loadTrainigData(\mathcal{F}_n)$;
  $X_{validation}, Y_{validation} = loadValidationData(\mathcal{F}_n)$;
  for $param$ in $hyperparameters$ **do**
    $clf.set_params(param)$;
    $clf.fit(X_{train})$;
    $Y_{pred} = clf.predict(X_{validation})$;
    $Accuracy.append([clf, evaluate(Y_{pred}, Y_{validation})])$;
  end for
end for
$clf_{best} = getBestClf(Accuracy)$;
return clf_{best};
end

in the training set, and d is the total number of features. The time complexity of our kernel SVM is linear with the number of support vectors n_s and the number of features d and can be represented as $O(n_s * d)$. The elapsed time for both steps is around $500ms$ on the ESP8266, allowing a complete evaluation between two acquisitions.

4.3.3 Experimental Results

For each case (anomaly and multiclass classifier) the best model to verify the entire system over the test set has been selected. In the following subsections, the obtained results are reported.

Anomaly Detection Results

The best results have been obtained in the Fold0 case. In the case of Fold0, all the experiments between 1 to 9 have been used as training and validation sets following the cross-validation technique, while experiments 0 of all substances have been used as the test set.
The best results are reported in Table 4.6, while the average plus the standard

Algorithm 2 Test procedure

input: A TestSet $\mathcal{T}$, best anomaly model ($anly$), best multiclass model (clf), doAnomaly
output: [Accuracy, CM]
begin

groundtruth = $getGroundtruth(\mathcal{T}_n)$
for $sample$ in $\mathcal{T}_n$ **do**
  if $doAnomaly$ **then**
    $outClass.append(onlineClassidication(sample)$;
  else
    $sample_n = normalize(sample)$;
    $state = getFsmState(sample_n)$;
    if $state \neq BS$ **then**
      $outClass.append$("BKG");
    else
      $outClass.append(clf.predict(sample_n))$;
    end if
  end if
end for
$ConfusionMatrix = evaluate(outClass, \text{groundtruth})$;
return $[Accuracy, CM]$;
end

deviation (STD), obtained over all the splits of the Fold0 data set, are reported in Table 4.7. It is worth noting that the obtained results show almost the same performance across the used algorithms. Thus, it is impossible to declare a winner easily. For that reason, since the application field of the proposed system best fits the novelty detection approaches, to test the entire system has been used the One-class SVM classifier.
Furthermore, to statistically validate the obtained results, we performed the Wilcoxon rank-sum test ($\alpha = 0.05$). Indeed, Table 4.7 also shows the p–value of the Wilcoxon test. From the table, it is possible to see that the performance differences between the three algorithms that best perform (One-Class SVM, Elliptic Envelope and Isolation Forest) are not statically significant (p-value > 0.05). Regarding the Local Outlier Factor and the KNN algorithm, it is possible to notice that the p-value is < 0.05, highlighting a statistical difference between the obtained results. Finally, it is worth noticing that the Wilcoxon test has been performed by evaluating all chosen figures of merit (Accuracy, F1Score and MCC).

Algorithm 3 Online Classification procedure

input: Sample $\mathcal{S}$, anomaly detection classifier (anly), multiclass classifier (clf)
output: predicted class ($outClass$)
begin

$\mathcal{S}_n = normalize(\mathcal{S})$;
$state = getFsmState(\mathcal{S}_n)$;
if $state \neq BS$ **then**
  $outClass =$ "BKG";
else
  if $anly.predict(\mathcal{S}_n) =$ inlier **then**
    $outClass = clf.predict(\mathcal{S}_n)$;
  else
    $outClass =$ "UNKNOWN";
  end if
end if
return $outClass$;
end

Regards the reported figure of merit they have been computed by the following formulas:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.4}$$

$$\text{F1 Score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad \text{where} \quad \begin{cases} \text{precision} = \frac{TP}{TP+FP} \\ \text{recall} = \frac{TP}{TP+FN} \end{cases} \tag{4.5}$$

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{4.6}$$

Where the True Positive (TP) are all the outlier samples classified as an outlier, True Negative (TN) are all the inlier samples classified as inlier, False Positive (FP) are all the inlier samples classified as an outlier, and False Negative (FN) are all the outlier samples classified as an inlier.

Multiclass Classifier Results

The best result obtained in the Fold0 is obtained by the KNN algorithm, using a number of neighbors (N) equal to 10 and adopting uniform weights. The obtained accuracy is equal to 99.37%. In terms of mean accuracy and standard deviation over the nine folds contained in Fold0, the obtained result is A%= (91.0 $\pm$ 5.7)%.

Table 4.6: Best Results Anomaly Detection

Approach	Algorithm	Accuracy	F1 Score	MCC	Parameters		Split
Novelty	One-Class SVM	0.9546	0.8868	0.8675	ν Kernel γ	0.01 rbf 0.45	6
	KNN	0.5982	0.0938	-0.2485	contamination N neighbors	0.45 10	1
	LocalOutlierFactor	0.8624	0.7639	0.7123	contamination N neighbors	0.01 400	7
Outlier	Elliptic Envelope	0.9545	0.8864	0.8671	contamination	0.05	6
	Isolation Forest	0.9547	0.8872	0.8679	contamination N estimators	0.1 350	4

Table 4.7: CrossValidation0 Results

Algorithm	Accuracy	F1Score	MCC	p-value
One-Class SVM	0.9358 ± 0.0171	0.8474 ± 0.0352	0.8115 ± 0.0497	–
KNN	0.5651 ± 0.0226	0.0902 ± 0.0018	-0.2762 ± 0.0188	5.6e−6
Local Outlier Factor	0.8201 ± 0.0346	0.7137 ± 0.0382	0.6519 ± 0.0463	5.6e−6
Elliptic Envelope	0.9325 ± 0.0157	0.8448 ± 0.0311	0.8069 ± 0.0445	0.4860
Isolation Forest	0.9463 ± 0.0130	0.8689 ± 0.0277	0.8423 ± 0.0391	0.7317

Entire System Results

Two main tests have been done to highlight the benefits obtained from the usage of anomaly detection followed by the multiclass classifier for the entire system concerns.

Once with only the multiclass classifier and once with anomaly detection plus the multiclass classifier. The results are shown in Figures 4.12 and 4.13. As can be seen from the two confusion matrices, the outlier substances used are:

- Dish Wash Detergent (DW_DETERGEMT)
- Nelsen (INT_NELSEN)
- Washing Machine Detergent (WM_DETERGENT)
- Sodium Chloride (SODIUM_CHLORIDE)
- Sodium Hypochlorite (SODIUM_HYPOCHLORITE)

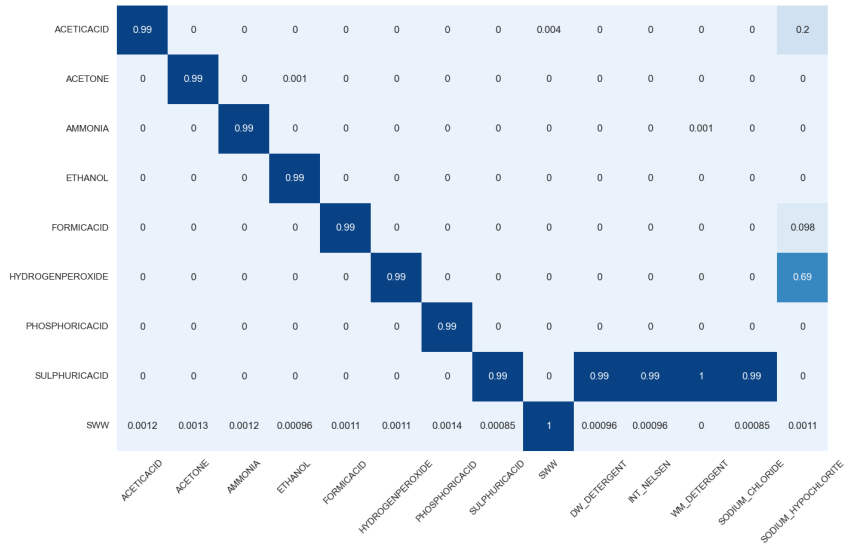


Figure 4.12: Multiclass Classifier Results

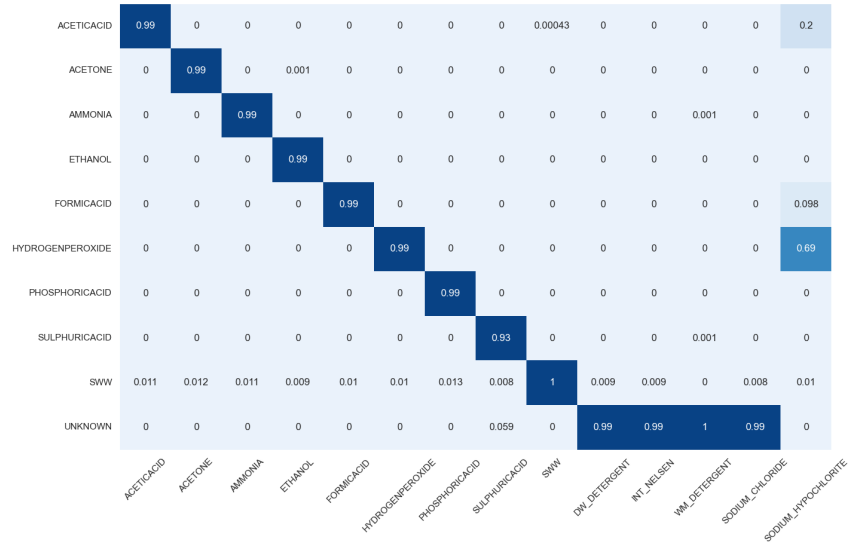


Figure 4.13: Entire System Results

In the case of the Multiclass classifier only, the outlier substances get erro-

neously confused with one of the known substances generating many false positive alarms. To solve this problem, as described in the previous sections, an anomaly detection system capable of working as a false positive reduction filter has been added before the multiclass classifier.
As reported in Figure 4.13, with the addition of the anomaly detection system, most outlier samples get correctly labeled as "UNKNOWN". More precisely, 79.4% of the outlier samples has been correctly labeled as "UNKNOWN", while the remaining 20.6%, which represents all the sodium hypochlorite samples, gets mostly confused with the hydrogen peroxide (according to what is shown in Figure 4.12). Finally, as already depicted in Section 4.3.3, since the obtained results (see Table 4.6 and 4.7) show almost the same performance across the used algorithms, and given the application field of the proposed system, in the reported result the One-class SVM classifier has been used.

4.3.4 Field tests

To obtain field tests, two preliminary experiments were conducted on real scenarios: one at the Acqualatina treatment plant in Borgopiave (Latina, Italy) (see figure 4.14), and the second on a series of wells located in Via Castelbottaccio (Rome, Italy) in collaboration with ACEA S.p.A. [84] (see figure 4.15).



Figure 4.14: Borgopiave. Green circle represents the sensing manhole, while red circles represent the spiking manhole positioned at 60m from the sensing manhole.

A flotation system has been designed to allow the immersion of the SCW in water at the proper depth, as shown in figure 4.16.

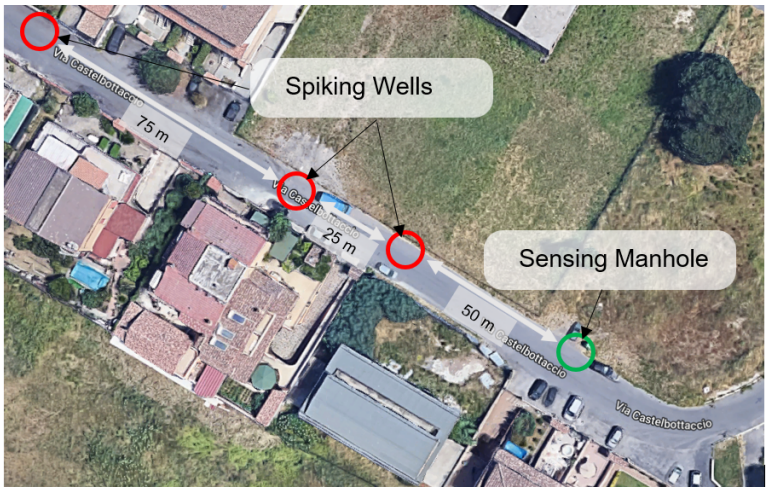


Figure 4.15: Castelbottaccio. The green circle represents the sensing manhole, while red circles represent the spiking manholes respectively positioned at 50m, 75m, and 150m from the sensing manhole

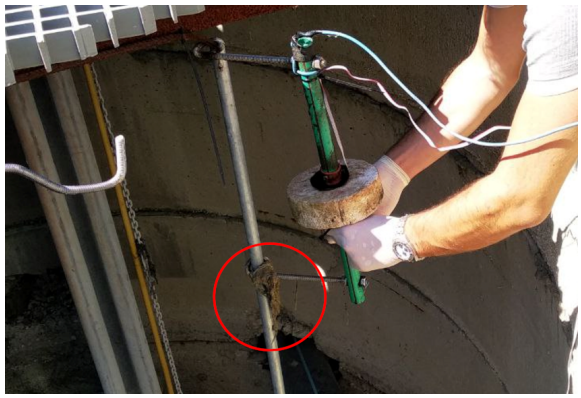


Figure 4.16: The flotation system for the immersion of the SCW. In red are highlighted some dirt.

To install the sensing system completely inside the manhole, a prototype of the measurement system has been developed, as shown in figure 4.17. As can be seen, the measurement system is composed of an IP56 waterproof certificate box, a Raspberry Pi4, a GSM hat for Raspberry Pi based on SIM7600E-H with two external antennas, a 20000mAh power bank, an ESP8266 board connected to the SCW via a 10m SENSIBUS cable. This configuration can ensure a continuous

measurement and transmission of about 1 week. The prototype presented here was designed for a field test without considering energy consumption. A solution based exclusively on MCU (without Raspberry) and the LoRaWAN (Long Range Wide Area Network) standard could work continuously for several months of continuous monitoring with periodic transmissions, exploiting an adequate local memory.

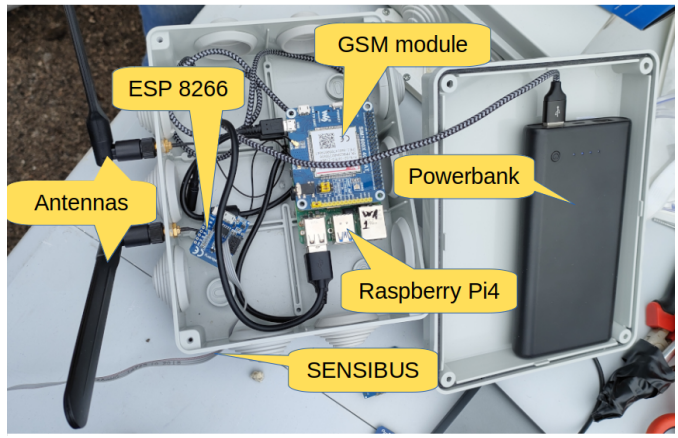


Figure 4.17: Measurement system prototype

In this context, many substances have been tested: Phosphoric Acid, Sodium Hypochlorite, Acetic Acid, Formic Acid, Ammonia, and Hydrogen Peroxide. Unlike laboratory tests, in the real context, they encountered many problems, such as the accumulation of scale or air bubbles near the sensors of the SCW. After some preliminary tests, some of the problems encountered were solved, and although other problems were still present, it was still possible to reach an accuracy of more than 80%.

4.3.5 Discussion

From the Experimental Results section, given an outlier sample as input to a multiclass classifier, the output will fall under one of the known classes. This behavior leads to generating a number of false alarms equal to the number of outlier samples, making the system useless in a real scenario application. The results shown in Figure 4.12, clarify the drawbacks of using only the multiclass classifier system to recognize a given substance. In this case, indeed 100% of the outlier samples represented by the dish wash detergent, washing machine detergent, sodium chloride, and sodium hypochlorite has been mainly confused with the sulphuric acid and the hydrogen peroxide generating a great number of false alarms. A multiclass

classifier, used alone, cannot reject any outlier sample. For that reason, an anomaly detection module as a false alarm filter has been introduced to solve this behavior. Table 4.6 and 4.7 show the results obtained by the anomaly detection system. As can be seen, the performance of the One-Class SVM, Elliptic Envelope and Isolation Forest are pretty similar, this means that the three algorithms can be equally used. Moreover, to statistically validate the obtained results, the Wilcoxon rank-sum test ($\alpha = 0.05$) has been performed, and the results, shown in Table 4.7, didn't show any significant differences. Finally, the chosen figures of merit show that the anomaly detection algorithms can distinguish between the outlier samples and the normal ones correctly.

At this point, putting the anomaly detection and multiclass classifier systems together has been possible to reach the results shown in Figure 4.13. The results show that the anomaly detection system has rejected most outliers samples (around the 80%) by labeling it as "UNKNOWN". As can be seen, all the sodium hypochlorite samples have been mainly confused with hydrogen peroxide. Even though this is a behavior that worsens the system's performance, it can still be considered an acceptable behavior. Indeed, even if the two substances are chemically different (i.e., sodium hypochlorite is a polar compound while hydrogen peroxide is non-polar), they have a similar oxidation potential: $1.6V$ for the sodium hypochlorite and $1.75V$ for the hydrogen peroxide [85]. Moreover, among all the substances of interest, hydrogen peroxide is the only compound to be considered a strong oxidant (in a range that goes from $+3V$ for the oxidants to $-3V$ for the reducers). This similarity is particularly evident with the measurements at $78kHz$. The confusion between sodium hypochlorite and hydrogen peroxide can be considered acceptable for all those reasons.

For the substance of interest concern, instead, Figure 4.13 shows as the multiclass classifier results (see Figure 4.12) are substantially maintained. Indeed, the overall accuracy reached by the entire system over the substances of interest is around 98.44% with a 0.93% of accuracy loss concerning the multiclass classifier system alone (99.37%).

With the consideration made, we can say that the improvement made by putting an anomaly detection system before the multiclass classifier has been proven. The system has been able to reject around the 80% of outliers samples and correctly recognize around 98.44% of inlier samples. Finally, given a real scenario application, it is clear that the presence of an anomaly detection module is of vital importance for the utility of the system itself.

In conclusion, after all the discussed analysis, we would like to report what we believe represents the major system's weaknesses. In particular, from the classification system point of view, as already discussed, one problem is related to all

those substances which share some chemical properties, as stated before. Another weakness is related to sensor poisoning. More in detail, from some laboratory tests, it has been noted that SCW's sensors are particularly sensitive to acids substances, capable of poisoning SCW's sensors, inhibiting their ability to distinguish the different substances for a while. For sure, the poisoning problem is a complex one that countless other substances can cause, but, to the best of our knowledge, acids are the most difficult to recover. Eventually, another problem regarding the classification system is related to the computational complexity that tends to increase linearly with the number of learned substances, so as a consequence, a suitable MCU has to be selected.
As the tests on the field concern, another system weakness related to the solid waste emerged, which can get stuck over the SCW's sensors altering all the measurements made, leading, in the worst cases, to a degradation of the overall system's performance.

Although the proposed algorithm could be applied to an array of sensors for real-time spilling recognition, to the best of our knowledge, there are no similar publicly available data sets nor papers that follow a similar approach. This makes it impossible to provide experimental results other than those obtained from our data. At the same time, in recent scientific literature, many papers have faced the problem of wastewater analysis. Still, nothing of these use an array of generic IDE in real-time recognition of different substances.

We also discarded the possibility of considering the signals during the injection as time series because their pattern over time would be highly dependent on the rate of injection of the substance (quantity per second), rather than its nature. We preferred an approach where the orthogonality of the adopted sensor, read "instantaneously", gives us the right information, not considering what happens in the time dimension.

As a final remark, it is known that Deep Learning [86] Algorithms (DLA) are well-established in many fields. However, we haven't considered such an approach in this paper because the feature space that we obtained from sensors is very small. DLAs are well-established for time series, images, Natural Language Processing, etc., but their effectiveness is questionable when the feature space is too small. This assumption is also demonstrated in a previous paper, where Convolutional Neural Network (CNN) and Long-Short Term Memory models (LSTM) have been considered [43], and where traditional machine learning techniques like Multi-Layer Perceptron or kNN have outperformed CNN and LSTM.

4.3.6 Conclusions and Future Directions

In conclusion, the proposed work has been meant to develop a stable and robust detection system capable of working in an aggressive environment such as the one represented by the sewage network. The complex environment implies that many different substances can be present, even those whose danger level is not significant and, therefore, not to be detected by the proposed system. Nevertheless, adopting a classical supervised ML approach, whatever substance would be recognized as belonging to the training set. The important novelty that has been carried out and proven effective in this work is implementing a two-stage scheme to reduce false alarms strongly and keep classification accuracy very high. To do that, a Finite State Machine was intended to filter, process, and normalize the measured sensors data, and then a detection system was built. The detection system is divided into two main parts, one represented by the One-class SVM classifier, an anomaly detection algorithm to reject all the samples belonging to the unknown substances, and one represented by the KNN multiclass classifier to recognize the given substance belonging to those of interest.
From the obtained results, shown in Section 4.3.3, it can be seen that the developed system works as supposed, drastically reducing the false positive errors given by the outlier samples and keeping accuracy on the substances of interest higher than 0.93 in all considered cases.

Regarding future developments, the authors would like to continue testing in real scenarios to validate quantitatively the promising results obtained in the laboratory activity and enforce the generalization property of the proposed system. Furthermore, the sodium hypochlorite confusion shown in Figure 4.13, as discussed in Section 4.3.5, suggests that the anomaly detection system could confuse substances with some common chemical properties. A possible way to reduce this phenomenon as much as possible would be to investigate an optimal set of independent features that can exploit the chemical differences to maximize the overall system performance.

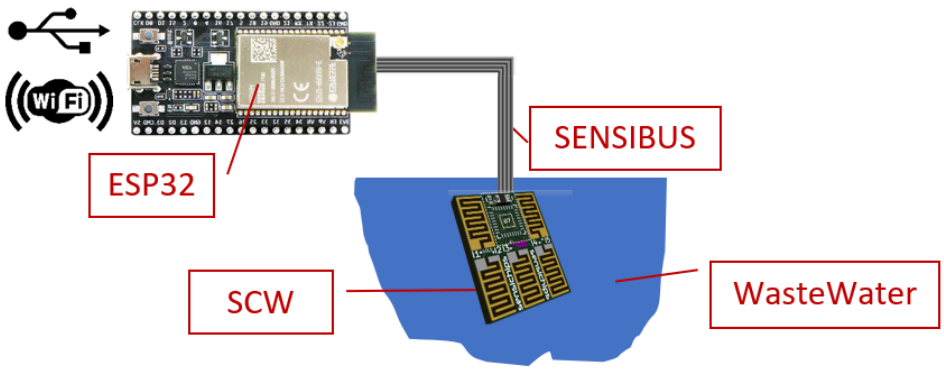


Figure 4.18: The hardware acquisition chain at a glance.

4.4 Natural Language Generation pollution detection with low-cost sensor platform

The proposed system is end-to-end and contains hardware and software components, which are detailed in the following.

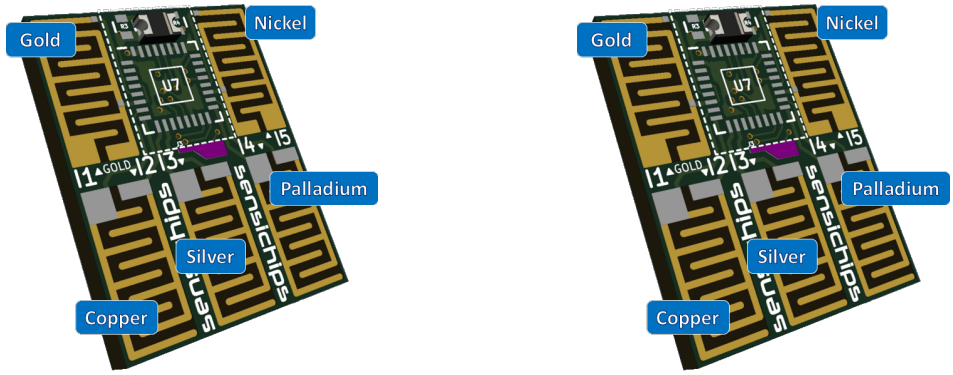
4.4.1 System at a glance

Hardware

The hardware part of the acquisition chain can be seen in Figure 4.18 where the following components are depicted: the Smart Cable Water (SCW), that is the sensing element; the SENSIBUS cable, a proprietary one-wire cable that allows communication with and control of the SCW; and a Micro Control Unit with an onboard firmware for controlling SCW, gathering and transmitting data to the cloud.

The SCW is a low-cost multi-sensory proprietary system (see section 2.2.1), based on SENSIPLUS technology, capable of carrying out Electrochemical Impedance Spectroscopy (EIS) and Voltammetry measurements. SENSIPLUS is a proprietary technology of Sensichips s.r.l. developed in collaboration with the University of Pisa [78].

The SCW is equipped with multiple sensors consisting of 6 Inter Digitated Electrodes (IDE) realized on a base made of copper and functionalized with Gold, Oxide of Copper, Platinum, Silver, Nickel, and Palladium (see Figure 2.3). These sensitive elements, in conjunction with the EIS available on the chip, constitute the sensors adopted for the collection of the samples present in the dataset. In detail,



(a) Front (b) Rear

Figure 4.19: Smart Cable Water with its Inter Digitated sensors

measurements have been performed on five of these sensors. The IDE metalized with Platinum and Gold has been analyzed at two specific frequencies: 200Hz and 78KHz, while Copper, Silver, and Nickel only at 200Hz. Different stimulus frequencies allow the exploitation of different frequency responses since the interactions between the metals on the sensors and the pollutants vary according to this parameter. In total, 12 quantities proportional to the capacity and resistance of the IDEs were acquired. Table 4.8 reports the correspondence between IDEs and frequencies.

Acquisition and Pre-Processing Software

The software components of the elaboration chain can be seen in Figure 4.20 where the following components are visible: the C API implemented as firmware for the MCU, a Finite State Machine (FSM) for baseline acquisition and injection detection of substances, the classification system. The MCU's firmware controls the SENSIPLUS chip through the SENSIBUS channel, collecting raw data and transmitting it to the computational module. The computational module (that could be a workstation connected through USB or systems in the cloud-connected with TCP/IP through Wi-Fi) is responsible for running the FSM and the classification system. The classification system is described in Section 4.4.3, while the FSM is described in the following.

The FSM is represented in Figure 4.21 and works in two steps:

- baseline extraction: a baseline signal is extracted to normalize raw data;

Table 4.8: Sensors used in the experiments.

IDE	Acquisition Frequency	Integer Value Proportional to
Platinum	78 kHz	Resistance
Gold	78 kHz	Resistance
Platinum	200 Hz	Resistance
Platinum	200 Hz	Capacitance
Gold	200 Hz	Resistance
Gold	200 Hz	Capacitance
Copper	200 Hz	Resistance
Copper	200 Hz	Capacitance
Silver	200 Hz	Resistance
Silver	200 Hz	Capacitance
Nickel	200 Hz	Resistance
Nickel	200 Hz	Capacitance

- forwarding decision: for each sample, the FSM decides whether to forward it to the classifier, also providing the injection time.

The FSM generates the baseline signal $\boldsymbol{b}_t$ by an Exponential Moving Average (EMA):

$$\boldsymbol{b}_t = \begin{cases} \boldsymbol{s}_t & t = 0 \\ \boldsymbol{b}_{t-1} & t > 0, S \in \{BS, BSP\} \\ \alpha \boldsymbol{s}_t + (1-\alpha) \cdot \boldsymbol{s}_{t-1}, & t > 0, S \in \{WT, BA, BT\}, \end{cases} \tag{4.7}$$

where s_t are the sensors' raw data at time t; {WT, BA, BT, BSP, BS} are the possible states of the FSM and correspond respectively to {Wait, Baseline Acquisition, Baseline Tracking, Baseline Suspended, and Baseline Stopped}.

The α parameter in EMA is the reciprocal of $\mathrm{EMA_c}$ that has been empirically set to 25. The normalized signal $\boldsymbol{f}_t$ forwarded by the FSM, is evaluated as

$$\boldsymbol{f}_t = \frac{\boldsymbol{s}_t}{\boldsymbol{b}_t}, \tag{4.8}$$

where $\boldsymbol{s}_t$ is the raw data collected from sensors and the $\boldsymbol{b}_t$ is the baseline signal computed as described by Equation 4.7. $\boldsymbol{f}_t$, $\boldsymbol{s}_t$, and $\boldsymbol{b}_t$ are n-dimensional vectors with n equal to the number of sensors (see Table 4.8); the division in the equation is element-wise.

Thanks to this baseline, the system can mitigate sensor drift, environmental noise, signal spikes, and variability between sensors.

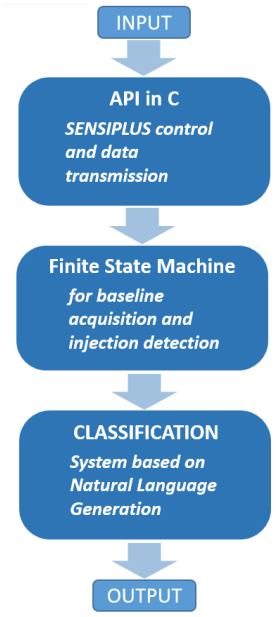


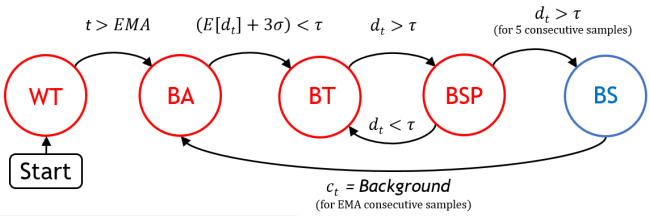
Figure 4.20: The software chain at a glance.

In the schema of the FSM reported in Figure 4.21, t is the current time, τ is a threshold empirically set to 0.05, $\boldsymbol{d}_t$ is the Euclidean distance between $\boldsymbol{f}_t$ and a vector of ones denoted as $\boldsymbol{u}$ in the feature space.

When $\boldsymbol{b}_t$ is equal to $\boldsymbol{s}_t$ in Equation 4.8, the $\boldsymbol{f}_t$ vector is the unit vector. As a consequence, the Euclidean distance has been computed for the unit vector, and $\boldsymbol{d}_t$ equal to zero means that the baseline signal $\boldsymbol{b}_t$ is perfectly tracking the sensors signals s_t:

$$\boldsymbol{d}_t = |\boldsymbol{f}_t - \boldsymbol{u}| \tag{4.9}$$

The WT state is conceived to "fill" the EMA; the BA state is reached automatically after EMA_c samples. Through the BA state, the FSM starts to follow the signal waiting for good tracking. Good tracking is obtained by analyzing the distance with the baseline. Once the variability of the distance, computed as its mean plus three times the standard deviation, is below a given threshold τ (empirically established to 0.05), the system can move to the BT state. The system will then check if a substance has been spilled in the water by checking when the current distance is greater than τ. When the signal moves away from the baseline, the state becomes BSP for a while. Once the FSM moves to the BSP state, the system will check that the current distance remains above the threshold for



Symbols	
t	time sequence
d_t	distance of s_t from b_t
EMA	Exponential Moving Average coefficient = 25
τ	the threshold (0.05)
c_t	classification of sample t
$E[d_t]$	the mean over last EMA
σ	the std dev over last EMA sample

States		Output
WT	WaiT	BKG
BA	Baseline Acquisition	BKG
BT	Baseline Tracking	BKG
BSP	Baseline SusPended	BKG
BS	Baseline Stopped	The sample is forwarded to the classification module

Figure 4.21: Finite State Machine.

five consecutive samples (BSP); otherwise, the system comes back to the BT state (to avoid confusing the spill of a substance with a measurement spike or noise). Finally, when the FSM reaches the BS state, the current normalized sample $\boldsymbol{f}_t$ is forwarded to the classification module.

The Classification Module

The proposed classification module is based on deep learning for natural language processing, and in particular on Transformer-based [87] models. Employ T5 [88], which is a large text-to-text language model pre-trained on a multi-task mixture of unsupervised and supervised tasks, the former being unsupervised de-noising objective tasks, while the latter being text-to-text language modeling objective ones. For a complete overview of tasks and prompts please refer to [88].

The T5 model architecture is similar to that of a general Transformer model and it is composed of a stack of encoder blocks, which transform the input text into a latent representation, and a stack of decoder blocks which translate the latent distribution into a new output text. Each block comprises a self-attention module, optional encoder-decoder attention, and a feed-forward network. Since it is a text-generation model, it takes a textual input and generates a textual response.

The model was adapted to the task of substance prediction by textualizing the raw sensor input and training the model to produce a string containing the nomenclature of the pollutant present in the wastewater. All parts of the classification

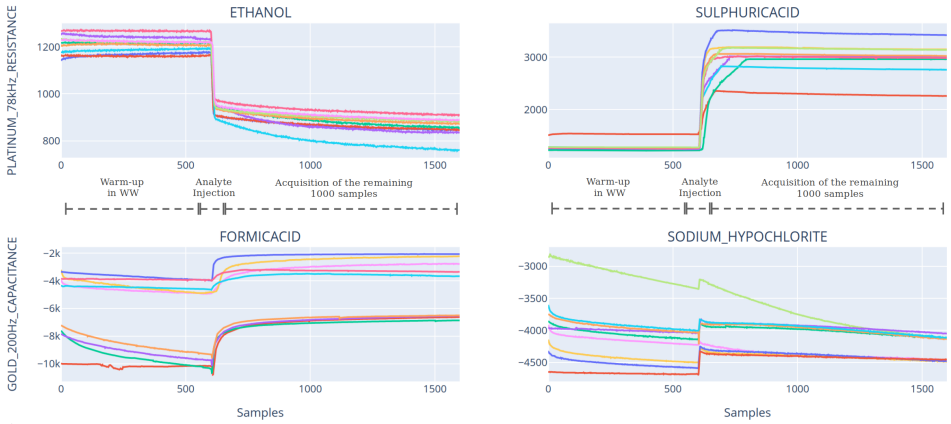


Figure 4.22: Example of acquired signals. The figure shows a subset of the signals present in the dataset. In detail, there are two features collected on four different substances. For each substance, there are nine experiments.

module are described in detail in the following Sections.

4.4.2 Data

Substances

The dataset used in this work aims to identify pollutants in WasteWater (WW), paying attention to spills of chemical compounds that could compromise public safety and/or the efficiency of purification systems. The acquisitions were made in the laboratory to simplify data collection. Measurements at experimental sites were excluded for two reasons: to ensure safety due to biological risks related to the presence of unknown bacteria or pollutants and to have controllable measurement conditions. The composition of WW is not stable over time, for example, due to atmospheric events such as rain. In detail, all the samples were acquired between 2019 and 2021 in two different laboratories in Poland and Italy and were recently made public [89]. Table 4.9 reports the substances used. The dataset consists of 10 acquisitions for each substance (including the WW or background) and was obtained using the measurement protocol described below.

The dataset

To create the dataset which is used in the experimental part of this paper was employed a measurement system composed of a PC as a control device, a MCU

Table 4.9: Substances used in the experiments.

Substance	Description
Wastewater	pH=7.4, conductivity=1.341mS
Acetic acid	CH_3COOH
Acetone	C_3H_6O
Ammonia	NH_3
Ethanol	C_2H_5OH
Formic acid	CH_2O_2
Hydrochloric acid	HCl
Hydrogen peroxide	H_2O_2
Phosphoric acid	H_3PO_4
Sodium hypochlorite	NaClO
Sulphuric acid	H_2SO_4

which manages the communication between the PC and the multi-sensor system, and the SCW that acquires the sensor's signals. The different substances have been injected into a beaker containing 300ml of WW, where the SCW is immersed. A magnetic stirrer was used to simulate the movement of the WW, ensuring the same conditions for each measurement session. The rotation of the anchor, 25mm long, was set at 50 rpm in such a way as to reduce the presence of air bubbles that could make the measurement noisy (turbulent regime). The acquisition of the samples present in the dataset was carried out according to a measurement protocol divided into two steps:

1. initially 600 samples are collected in WW, in warm-up mode;
2. subsequently, the substance of interest was injected, and an additional 1000 samples are collected.

This protocol was repeated ten times for each substance (ten acquisitions for each substance). A total of 1600 samples are collected for each substance and each acquisition, with an acquisition rate of about 1.6 seconds, for an overall run time of about 40 minutes for acquisition (Figure 4.22).

4.4.3 Experiments

Problem Formulation

Having the measurements obtained on 10 substances plus the background substance (i.e., WW). To get definitive and stable results and to avoid randomness and bias, this experiment uses the k-fold validation method. Specifically, a 10-fold validation

Table 4.10: Number of samples by substance. For each acquisition, 600 samples are collected in Wastewater and 1000 samples after a substance injection. Each fold's training set comprises 9 acquisitions for each substance, while the test set comprises 1 acquisition.

Substance	**Training set**	**Test set**
Wastewater	$(600 \cdot 10 \cdot 9 + 1600 \cdot 9)$	$(600 \cdot 10 \cdot 1 + 1600 \cdot 1)$
Acetic acid	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Acetone	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Ammonia	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Ethanol	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Formic acid	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Hydrochloric acid	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Hydrogen peroxide	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Phosphoric acid	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Sodium hypochlorite	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Sulphuric acid	$(1000 \cdot 9)$	$(1000 \cdot 1)$
Total samples	158,400	17,600

process was implemented where the experiment used for testing was rotated from 1 to 10 and the remaining experiments were used for training. This approach ensures that the samples from each experiment are used once for testing and that the overall performance metrics are averaged across all 10 test sets. In this way, a robust and reliable evaluation of the effectiveness of the models was achieved. For each substance, there are 600 samples collected in the so-called "warm-up mode", which means that the monitoring sensors were only exposed to WW during this period. Then there are also 1000 samples collected after substance injection, which means that in this period the sensors were exposed to both wastewater and the specific substance (if injected).

Following the k-fold technique detailed above, for each substance (including WW), 9 acquisitions have been used as the training set and 1 acquisition has been used as the test set. The effectiveness of the models is thus evaluated on the average of the test acquisitions, and only on the 1000 samples collected after the warm-up phase. Details on the composition of the dataset folds can be found in Table 4.10.

Metrics

To evaluate the performance of the model, different effectiveness metrics were chosen, aimed at measuring different aspects of the model's effectiveness. The following notation was used for the set of correctly and incorrectly identified samples:

- TP (True Positives): samples correctly identified as belonging to a substance of interest.
- TN (True Negatives): correctly identified negative samples, i.e., samples collected in WW.
- FP (False Positives): samples collected in WW, but classified as one of the 10 substances of interest.
- FN (False Negatives): samples collected in the presence of the substance of interest but classified as WW.

Furthermore, the following notation to denote the metric averaging method was used:

- metric_m: micro averaged metric; is the sum of the contributions from all classes.
- metric_M: macro averaged metric; compute the metric independently for each class and then take the average, hence treating all classes equally.
- metric_W: weighted metric; each class contribution is weighted by the relative number of samples available for such a class.

In addition, the following metrics are also calculated:

- Accuracy, defined as
$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}};$$
- Matthews Correlation Coefficient MCC for multi-classification (also called Rk statistic), defined as
$$\text{MCC} = \frac{(\text{TP} \cdot \text{TN}) - (\text{FP} \cdot \text{FN})}{\sqrt{(\text{TP} + \text{FP}) \cdot (\text{TP} + \text{FN}) \cdot (\text{TN} + \text{FP}) \cdot (\text{TN} + \text{FN})}};$$
- Precision, defined as
$$\text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}};$$
- Recall, defined as
$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}};$$
- F1-Score, defined as
$$\text{F1} = 2 \cdot \frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}.$$

Baseline Methods

A set of learning algorithms was selected and applied to the collected dataset to compare the proposed solution with standard Machine Learning techniques and obtain a reference baseline. The choice was made to have a sufficiently exhaustive representation of different approaches with different complexity. As a result, algorithms belonging to the following categories were adopted: boosting, bagging, tree-based, instance-based, kernel-based, Artificial Neural Networks, and ensemble classifies.

Table 4.11: The selected hyper-parameters for the various ML algorithms employed.

Model	Sub Model	Parameter	Selected value
K-NN	–	KNN	100
K-NN	–	algorithm	BallTree
K-NN	–	distance	Euclidian
SVM	–	kernelType	RBF
SVM	–	cost	1
SVM	–	gamma	0
AdaBoost	–	weightThreshold	100
AdaBoost	Decision Stump	–	–
AdaBoost	J48	confidenceFactor	0.25
AdaBoost	J48	numDecimalPlaces	2
AdaBoost	Random Forest	numIterations	100
AdaBoost	Random Forest	maxDepth	unlimited
MLP	–	hiddenLayers	20
MLP	–	decay	TRUE
MLP	–	learningRate	0.3
MLP	–	momentum	0.2
Random Forest	–	numIterations	100
Random Forest	–	maxDepth	0
Bagging	–	bagSizePercent	100
Bagging	–	numIterations	10
Bagging	REPTree	minNum	2
Bagging	REPTree	numFolds	3
Bagging	REPTree	noPruning	FALSE

As for the boosting algorithm, AdaBoost [90] was selected with different types of weak classifiers: decision stump, J48 tree [33] and a more complex Random Forest [91]. For the Bagging [92], REPTreewas chosen, a simple tree learner that

uses the information gain heuristic to choose an attribute and a binary split on numeric attributes (faster than C4.5) [93]. For decision tree-based algorithms, Random Forest [91] was chosen. The classic k-nearest neighbors (KNN) algorithm [94] was chosen for instance-based algorithms, support vector machines (SVM) [95] for kernel-based algorithms, and a multi-layer perceptron (MLP) for algorithms in the artificial neural networks category [31]. Finally, a majority vote between MLP, KNN, SVM, and RandomForest for ensemble-based algorithms.

The selected ML algorithms were preliminarily optimized through a grid search on their respective hyperparameters. Table 4.11 shows, for each algorithm, which hyperparameters were selected. The WEKA (Waikato Environment for Knowledge Analysis – version 3.8.6) implementation of these algorithms was used in the experimental phase [96]. The input of the ML algorithms is represented by the feature vector calculated in Equation 4.8 and therefore contains the instantaneous measurement of all the sensors identified in Table 4.8, following normalization for the baseline. Note that to maintain a fair comparison, the baselines and the proposed model (discussed in the next sections) were trained on the identical dataset using the same features; this ensures that any observed performance differences are solely due to the algorithmic capabilities, not data or feature discrepancies.

Textification of the Input

Unlike the baseline methods, the T5 model requires textual input to be trained for natural language generation. For this reason, it is first necessary to define a methodology to describe the input features (i.e., the observations of the set of sensors) in natural language. For this purpose, the so-called "textification" (or prompting) of the input features is used, an approach that has been successfully applied in the medical domain, in particular in the automatic coding and prediction of diagnostic texts [49–51], as well as in the prediction of human mobility [53, 54].

This transformation essentially takes an array of floating point values corresponding to the input features and translates it into a text, which will be then the input of the model.

Our approach works as follows. First, let us recall that each measurement is made of 1600 timestamps, $t \in [1, 1600]$ indicating the warm-up phase where the sensors are exposed to wastewater only, injection time happening at $t = 600$, and $t \in [601, 1600]$ indicating the phase after injection, where sensors are exposed to wastewater and the substance.

For each acquisition (out of the 10 present in each training set), two time stamps are used: t_b in the warm-up phase and t_a after the injection phase. Then, a piece of text with the following pattern is created for each of the sensors:

> the {capacitance/resistance} of {sensor}
> at {frequency} is {value}

By linking together the text for all the sensors and adding contextual information, a text with the following pattern is created:

> at time t_b ($600 - t_b$ before injection),
> the {capacitance/resistance} of {sensor}
> at {frequency} is {value},
> ...
> at time t_b ($t_a - 600$ after injection),
> the {capacitance/resistance} of {sensor}
> at {frequency} is {value},
> ...

where t_b and t_a are the sampled timestamps, and the other variables (such as {capacitance/resistance} are depending on the specific sensor used. Let us make it clear by reporting a real full example, where $t_b = 582$ and $t_a = 1141$:

> at time 582 (18 before injection),
> the Resistance of Platinum at 78kHz is 0.0444,
> the Resistance of Gold at 78kHz is 0.0151,
> ...
> the Capacitance of Nickel at 200Hz is 0.002.
>
> at time 1141 (541 after injection)
> the Resistance of Platinum at 78kHz is 0.783,
> the Resistance of Gold at 78kHz is is 0.9439,
> ...
> the Capacitance of Nickel at 200Hz is 0.992.

The process was repeated for all the acquisitions present in the dataset. Note that the proposed methodology allows to sample multiple t_b points to predict the substance present at t_a; in this case, the final prediction is obtained by taking the mode (i.e., majority voting) over the different predictions made by the model for the same acquisition.

Model Training and Inference

The model is developed using the PyTorch and HuggingFace frameworks. All data and code used in the paper will be made available at [81].

The model used is `T5-base`, which consists of an encoder and decoder stack, each consisting of 12 blocks. Each block contains self-attention mechanisms, optional encoder-decoder attention, and a feed-forward network. The attention is of dimension 64, while the embeddings are of dimension 768. The final model has about 220 million parameters.

The model weights are initialized with the pre-trained weights of the original T5 model. A custom prefix "predict:" is used to feed textual input to the model, and the strings "input:" and "target:" are used to distinguish between model input and target. The model is trained on a Linux server equipped with 16x Intel(R) Core(TM) i7-10700 CPU @ 2.90GHz, 64GB RAM, and 2x NVIDIA Geforce RTX 3090 GPUs for 3 epochs. As the objective, a conventional multi-class cross-entropy loss function is used, where the number of classes is equal to the size of the vocabulary, defined as

$$\mathcal{L} = -\frac{1}{B}\sum_{b=1}^{B}\sum_{k=1}^{|V|} y_k^b \log(\hat{y}_k^b),$$

where the superscript b represents the batch and B is the batch size, $|V|$ is the vocabulary size, y represents the true token to be predicted, and $\hat{y}$ is the output probability distribution over the vocabulary at each time step.

To perform the inference, the output text is generated by beam search, i.e. the output sequence is generated token by token by feeding the input to the decoder via cross-attention layers and auto-regressively generating the decoder output. The early stopping parameter is set to true so that beam generation stops when all beam hypotheses reach the `EOS` (end of sequence) token. Experimentally was found that the fine-tuned model generates substance names for each beam, so there was no need to implement a constrained beam search to force the model to output only correct strings as output (i.e., only produce one of the 10 substance names or WW). Since the training set can be augmented by sampling multiple time stamps for each acquisition, we aggregate the final predictions of the model using majority voting (i.e., the mode function) to have a single prediction for each t_a.

4.4.4 Results

Effectiveness

Table 4.12 shows the effectiveness metrics computed considering the average effectiveness score over each test set for the baselines (upper part of the table) and for

Table 4.12: Average effectiveness metrics computed over the test sets. metric_{m} denotes the micro-averaged metric, metric_{M} the macro-averaged metric, and metric_{W} the weighted metric.

Model	**Acc**	**MCC**	$\mathbf{Prec_m}$	$\mathbf{Prec_M}$	$\mathbf{Prec_W}$	$\mathbf{Rec_m}$	$\mathbf{Rec_M}$	$\mathbf{Rec_W}$	$\mathbf{F1_m}$	$\mathbf{F1_M}$	$\mathbf{F1_W}$
AdaBoostM1_DecisionStump	0.418	0.208	0.418	0.004	0.192	0.418	0.111	0.418	0.418	0.031	0.267
AdaBoostM1_J48	0.802	0.767	0.802	0.720	0.744	0.802	0.732	0.802	0.802	0.718	0.765
AdaBoostM1_RandomForest	0.767	0.722	0.767	0.672	0.705	0.767	0.673	0.767	0.767	0.652	0.720
Bagging	0.764	0.718	0.764	0.687	0.720	0.764	0.672	0.764	0.764	0.649	0.721
knn	0.803	0.769	0.803	0.732	0.746	0.803	0.730	0.803	0.803	0.722	0.765
MLP	0.794	0.758	0.794	0.724	0.745	0.794	0.715	0.794	0.794	0.707	0.759
RandomForest	0.779	0.737	0.779	0.682	0.711	0.779	0.693	0.779	0.779	0.672	0.733
SVM	0.802	0.768	0.802	0.749	0.754	0.802	0.726	0.802	0.802	0.722	0.765
Vote	0.810	0.778	0.810	0.747	0.756	0.810	0.739	0.810	0.810	0.732	0.773
T5 (1-sample)	0.915	0.865	0.915	0.950	0.950	0.915	0.950	0.950	0.915	0.895	0.895
T5 (2-samples)	0.914	0.864	0.914	0.936	0.936	0.914	0.936	0.936	0.914	0.894	0.894
T5 (5-samples)	0.927	0.877	0.927	0.958	0.958	0.927	0.958	0.958	0.927	0.907	0.907
T5 (10-samples)	0.958	0.908	0.958	0.976	0.976	0.958	0.976	0.976	0.958	0.938	0.938
T5 (50-samples)	0.944	0.894	0.944	0.958	0.958	0.944	0.958	0.958	0.944	0.924	0.924
T5 (100-samples)	0.975	0.925	0.975	0.988	0.988	0.975	0.988	0.988	0.975	0.955	0.955

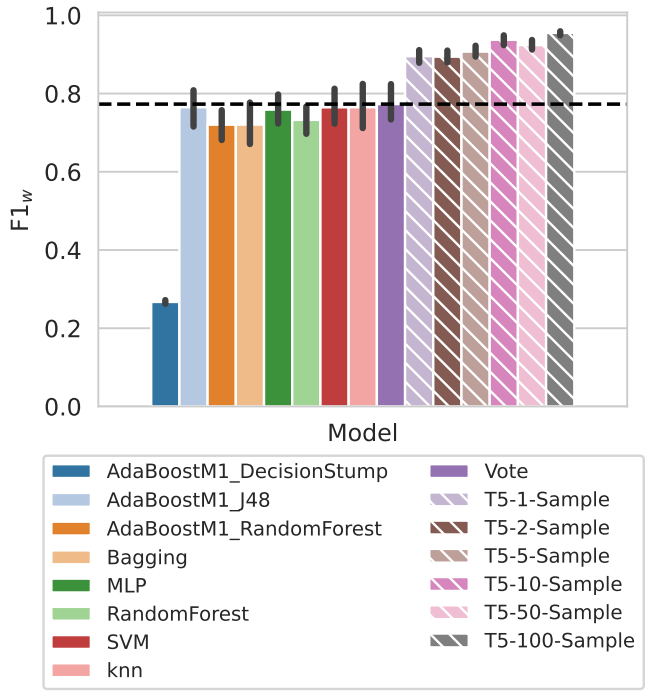


Figure 4.23: Average effectiveness metrics computed on the test sets for $F1_W$. The dotted line represents the best baseline method.

the proposed approach (lower part of the table). Given that the proposed methodology can be provided in input with multiple t_b timestamps (see Section 4.4.3), the lower part of Table 4.12 shows the different effectiveness scores computed when sampling 1, 2, 5, 10, and 100 t_b timestamps and aggregating the predictions of the model.

As can be seen by inspecting the table as a whole, the proposed methodology outperforms the whole set of baselines for all the considered metrics, even in the most restrictive case where only one t_b timestamp is provided (i.e., T5 (1-sample)). This behavior is also visually shown in Figure 4.23, which shows the value for the $F1_W$ metric on the test sets: the different models are arranged along the x-axis, while the y-axis shows the metric value, and the horizontal dashed line represents the performance of the best baseline method. The bars on top of each value represent the variance of the metric over the different folds. The plots for the other metrics are similar and thus not reported.

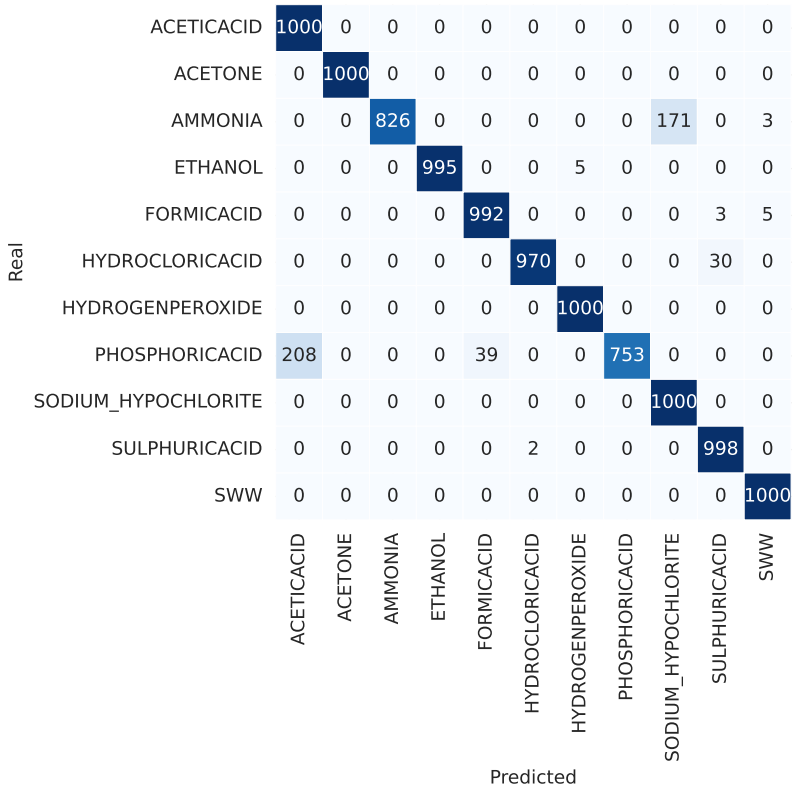


Figure 4.24: Distribution of the real and predicted values for the T5 (100-samples) model.

Figure 4.24 shows the confusion matrix produced by the most effective method, i.e. T5 (100 samples), shown in the last row of Table 4.12. As can be seen from the matrix, the model correctly classifies almost all substances perfectly (the values on the diagonal are close to 1000), except for ammonia and phosphoric acid, where the model achieves an accuracy of about 0.8. In particular, ammonia is often mistaken for sodium hypochlorite (171 times out of 1000), and phosphoric acid is mistaken for acetic acid (208 times out of 1000). By examining the data set, it can be assumed that this is probably because the signals detected by the sensors show similar trends between the two pairs of substances. In particular, the initial conditions before the injection time are very similar.

Focusing on the lower part of the table 4.12, can be seen that there is a cor-

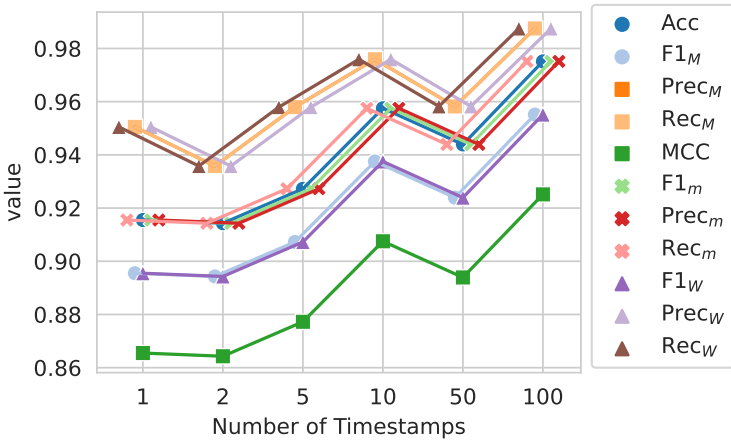


Figure 4.25: Average effectiveness metrics were obtained when varying the number of timestamps t_b fed to the T5 model.

relation between the number of timestamps measured before the injection and the model's effectiveness. More specifically, the more timestamps t_b are provided to the model, the higher the overall effectiveness score. This behavior is also shown in Figure 4.25, which plots the number of timestamps fed into the model on the x-axis and the value for the different effectiveness metrics on the y-axis. This result suggests that, in practice, the model with more samples from the warm-up phase can give a more accurate prediction. This is due to the fact that seeing more timestamps during the warm-up allows the model to better capture the signal variance that might occur before the substance is injected into the wastewater.

Efficiency

The inference time has been measured over repeated experiments on a NVIDIA Geforce RTX 3090 GPU. Disabling batching, it takes the system on average 76 milliseconds to perform one prediction (i.e., one full text-generation performed with beam search and the early stopping parameter enabled, see Section 4.4.3 for more details on the inference process) for a timestamp, or 76 seconds for 1000 timestamps. The system can therefore output approximately 13 predictions per second. It is also possible to further reduce the inference time by performing batch inference (the GPU used for the experiments allows batch sizes larger than 512 samples).

The inference time was also measured on the available CPUs (16x Intel(R)

Core(TM) i7-10700 CPU @ 2.90GHz). On average, the system takes 450 milliseconds to process a sample.

Such results demonstrate that the trained model can be used for real-time prediction in both a GPU-equipped environment and a CPU-only machine.

In addition to the inference efficiency, which is crucial for deploying a model in real-world scenarios, it is important to consider the computational cost and complexity of the training phase in the proposed approach. T5 and other causal models have millions or even billions of parameters, which necessitate a large amount of data and computational power to optimize. These models are typically trained using the standard transformer architecture, which has a complexity of $\mathcal{O}(N^2 d)$, where N is the sequence length and d is the hidden dimension of the model [97]. This means that the number of operations required to train the model grows quadratically with the input sequence length and linearly with the model's hidden dimension.

Despite the high computational cost, pretrained models can be repurposed for various tasks through finetuning, which involves adapting a pretrained model to a specific task by training it on a small amount of task-specific data. This process is considerably less expensive than training the model from scratch because most pretrained models are already optimized for the underlying language modeling task. In practice, fine-tuning a pre-trained model usually involves training it for only a few epochs, typically three, which can take anywhere from a few minutes to a few hours, depending on the size of the task-specific dataset. In this work, the weights of the model are obtained from the HuggingFace library and only the fine-tuning phase of the proposed approach was performed, which took about an hour on the GPU architecture described above. Once fine-tuned, the model can be used indefinitely for the specific task discussed in this work.

4.4.5 Discussion and Conclusion

In this section have been studied the capabilities of natural language processing models, in particular generative causal models, and more specifically T5, for the task of detecting the presence of pollutants in wastewater. For this purpose, unlike the state-of-the-art machine learning models, a transformation of the input features is applied, called textification, to translate them into a textual form and to be able to feed them into a generative natural language model. The latter is trained to classify each sample based on whether it contains a pollutant or not and to identify it if present. The proposed methodology is experimentally evaluated to confirm its effectiveness against a set of state-of-the-art baselines. Experimental results show that the proposed methodology outperforms the baseline methods, and its efficiency and effectiveness allow for its deployment and practical use.

Since the proposed approach is unconventional and may seem strange or counterintuitive at first glance, the following is a discussion of why such an approach makes sense and works in practice.

Recent work demonstrated the vast ability of transformers and attention based models to generalize on a large variety of tasks, including those where the model has not been trained on [98–101], or even to tasks not directly related to or not naturally expressed using natural language processing, such for example images [55, 56], videos [58], reinforcement learning [67], and graphs [68].

The ability of transformer-based models for generalization comes from the attention mechanism, and from the almost task-agnostic training procedure. It consists, in its base form, in reconstructing part of the input item, being it masked or perturbed using domain-specific techniques or to predict the continuation of the input (if the masked part is the last part of the input). Combined, these techniques allow the model to learn meaningful and -most importantly- general latent relationships in input sequences, and the ability to relate those to the network's output. For example, networks applied to texts show the ability to reconstruct missing text or generate it from a prompt, for images and videos the ability to reconstruct corrupted or missing images and frames, for graphs to learn complex graph sub-structures (i.e., arrangements of a set of nodes and edges), and so on. Besides those specific abilities, networks based on transformers and trained with masking or causal objectives (i.e., predict masked parts or predict the continuation of the input) show high generalization abilities across tasks and domains.

Despite the promising results obtained with this approach, some limitations must be reported. One of the main limitations is that the proposed approach relies on the knowledge of the injection time of the contaminants. This means that if the injection time is not known, the system may not be able to accurately classify the pollutants in the wastewater. In this paper, the problem is addressed by relying on a finite state machine that can accurately identify the injection time. However, future research will focus on developing integrated methods to overcome this limitation and provide an integrated single system.

Another limitation of the proposed approach is related to the availability of data, since a certain amount of labeled data is needed to train the deep learning model. Obtaining such data requires access to polluting substances or contaminated wastewater, and this can be difficult in practical situations. Future work will investigate alternative ways to generate synthetic data or explore transfer learning techniques to mitigate the data scarcity issue. The results of this work open a new research direction that will allow us to tackle environmental tasks such as the analysis and detection of polluting substances employing language models. Future work aims precisely at pursuing a broad adoption of natural language-based mod-

els on a variety of domains and tasks related to the identification of substances. Furthermore, it will also focus on studying the generalization and explanation abilities of the model by leveraging zero and few-shot learning techniques as well as interpretability frameworks.

Chapter 5

Smart Batteries

The growing global demand for sustainable energy solutions has positioned batteries as critical components in many applications, from Electric Vehicles (EVs) to renewable energy storage systems. Smart batteries play a crucial role as the world transitions to cleaner energy sources. These advanced batteries, equipped with sophisticated Battery Management System (BMS), provide real-time monitoring, optimization, and control to ensure efficiency, safety, and longevity. They are fundamental to reducing the environmental footprint of energy storage, improving overall system reliability, and advancing sustainability goals. Smart batteries are essential for managing range, charging efficiency, and system safety in electric vehicles. Accurate estimates of State of Charge (SoC) and State of Health (SoH) are critical for optimizing battery utilization. Advanced techniques such as Electrochemical Impedance Spectroscopy (EIS), coupled with machine learning models, provide more accurate SoC estimates, reducing the need for extensive calibration and shortening measurement times. This accuracy is especially critical as EVs become more prominent in transportation. A key initiative that underlines the European Union's commitment to the sustainable use of batteries is the launch of the "Battery Passport" as part of its 2023 Battery Regulation (Regulation 2023/1542) to ensure the sustainability and transparency of batteries. Mandated by the EU's 2023 Battery Regulation, this digital tool increases transparency throughout the battery lifecycle. The Battery Passport provides detailed information on a battery's composition, manufacturing process, carbon footprint, and recyclability, ensuring environmental and social standards compliance. Through QR codes, manufacturers and end-users can access comprehensive data on each battery, facilitating informed decisions about repair, reuse, and recycling. This passport plays a key role in achieving the EU's sustainability goals by ensuring traceability, reducing environmental impact, and promoting circular economy principles. In renewable

energy applications, smart batteries are essential for managing intermittent energy supplies, stabilizing grids, and ensuring the efficient use of renewable resources such as solar and wind. They enable the storage of excess energy and its redistribution during peak demand periods, improving the overall efficiency of the energy system. In summary, smart batteries, supported by initiatives such as the EU Battery Passport, are fundamental to the future of sustainable energy systems. They optimize performance and safety and align with global efforts to decarbonize industries and reduce the environmental footprint of energy storage solutions.

5.1 Statistical Approach to LFP Battery State of Charge Analysis

The most widely used method for SoC evaluation is coulomb counting [102]: it is based on integrating battery current and, as a drawback, requires knowledge of an initial SoC value needed for integration. This work uses Electrochemical Impedance Spectroscopy (EIS) for SoC estimation because it overcomes the coulomb counting problem. EIS is a Nondestructive technique [103, 104] that measures the electrical response of an electrochemical system following the application of a small alternating current stimulus signal at different frequencies [105–107]. Applied to batteries, the system's electrical response is used to estimate the battery impedance at different frequencies. Therefore, an EIS measurement estimates the battery impedance in a given frequency range (e.g. from 10 mHz to 1 kHz). A wide frequency range is essential to capture processes occurring on different time scales. High frequencies characterize electrolyte resistance and electrode/electrolyte interface capacitance, while intermediate frequencies probe charge transfer kinetics and electrode porosity effects. Low frequencies reveal mass transport phenomena. This range separates overlapping processes and provides a comprehensive understanding of battery performance and aging. Numerous studies in the literature have already shown a relationship between impedance and SoC [108–110]. The open challenge is identifying a methodology that allows SoC estimation from impedance measurement during normal battery operation. This involves developing a fast and accurate method to perform online estimates.

Typically, approaches used to estimate SoC using impedance measurements can be divided into 2 types: model-based and data-driven. In the first case, impedance measurements are used to estimate an Equivalent Circuit Model (ECM) consisting of basic electrical components that can replicate battery behavior [111, 112]. The electrical components used to estimate the ECM have a precise electrochemical meaning that helps model developers get real battery operation feedback. Having the ECM available, it is possible to use it directly to predict battery voltage,

knowing current and SoC, or invert the model to estimate SoC from knowledge of battery voltage and current. The main disadvantage of this approach concerns the time required to identify the ECM; in fact, time-consuming battery characterizations must be performed to derive the components of the ECM as a function of the SoC. Conversely, data-driven methodologies often use Machine Learning (ML) techniques to build an SoC estimation model starting from EIS data [113, 114]. In this case, the Electrochemical Impedance Spectroscopy (EIS) measurements do not reflect a precise electrochemical meaning within the ML model but are used for iterative learning of the technique. In this context, an interesting application is feature selection, where the goal is to select the best frequencies for EIS measurement that can maximize the accuracy of SoC estimation while minimizing EIS measurement time [115]. The main disadvantage of using data-driven approaches is the large amount of data required to train an estimated model effectively.

This research proposes the first step towards a statistical approach for estimating SoC from EIS measure. Specifically, the Kernel Density Estimation (KDE) [116] technique was used to explore the possibility of identifying correlations between the SoC and an estimated probability density function. To this end, we developed an experimental setup and characterization procedure for an experimental campaign of EIS measurements on 7 Lithium Iron Phosphate (LFP) batteries.

The section is organized as follows: subsection 5.1.1 gives full details of the experimental campaign carried out, subsection 5.1.2 describes how the KDE technique works and how it was applied to the battery SoC estimation, in subsection 5.1.3 the obtained results are discussed and in subsection 5.1.4 the conclusions are provided.

5.1.1 Experimental campaign

The experimental campaign was specifically aimed at evaluating the impedance of Lithium Iron Phosphate (LiFePO4 or LFP) batteries during their discharge process. The impedance measurements were performed using the Electrochemical Impedance Spectroscopy technique (EIS). The coulomb counting method is used for the State of Charge (SoC) evaluation during the battery discharge. In the implemented case, the current measurement rate is $0.5Hz$ (one measure every 2 seconds). The rate is not extremely high, but the discharging current is constant, so the error in the SoC estimation is negligible.

The complete measurement setup is shown in Fig. 5.1. A *Hioki BT4560* battery impedance meter was used to measure the impedance of batteries precisely. A *ZKEtech EBD-A20H* electronic load is also equipped to execute constant current battery discharge processes as described above. To control and monitor the environmental temperature, the batteries were placed inside a custom-made climate chamber. The chamber uses a temperature controller to drive two Peltier

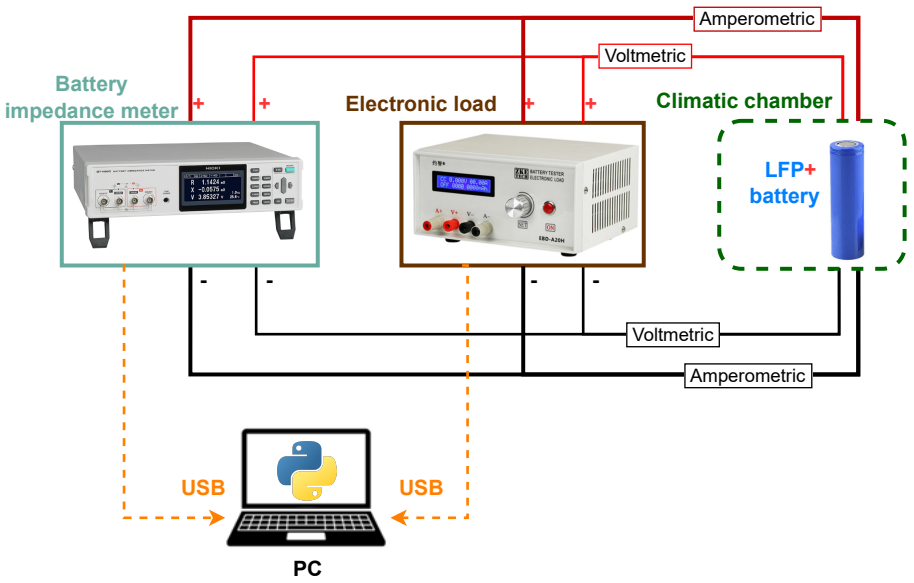


Figure 5.1: Schematic diagram of the developed measurement set-up.

modules. The controller has a resolution of $0.1°C$ and a global Accuracy of $\pm 1°C$. Lastly, a PC manages the entire operation via USB interfaces with the instruments. The chosen devices under test (DUT) are 7 LFP batteries, *O'CELL IFR14500EC-600 mAh* with a nominal voltage of 3.2 V and a nominal capacity of 600 mAh. Both the impedance evaluation and the discharging phase (where also the voltage is measured) are carried out with a four wires technique to reduce the impact of contact and cable resistances.

Before the discharging test, every DUT's capacity is evaluated by simply charging it and, after, fully discharging it until the minimum voltage (provided by the manufacturer) is reached. The capacity is computed with the coulomb counting method. All settings used to carry out the experimental characterization are listed in Tab. 5.1. During the characterization phase, a delay between the discharge phase and the EIS is required to allow electrochemical recombination processes to occur, this time is called *Relaxing time*. As demonstrated in literature [117, 118], the relaxing time affects the impedance measurements. The relaxing time is not unique but rather depends on the battery model employed. In this case, the relaxing time is experimentally evaluated in 15 minutes, by evaluating preliminary measurements at different times from the discharge and selecting the minimum steady-state time.

Table 5.1: Settings used for the experimental characterization.

Settings	**Values**
Frequency range	$[10\ mHz; 1\ kHz]$
Number of frequencies	29
Relaxing time	15 min
Electronic load discharge current	120 mA
ΔSoC	5 %
Cut off voltage (V_{min})	2 V
Climate chamber temperature	20 $°C$

The measurement protocol is illustrated in the Fig. 5.2, and listed below:

- **Battery Capacity evaluation**: the battery capacity is evaluated.
- **Battery in climatic chamber**: the battery is placed inside the climatic chamber for several hours to wait for the battery thermal regime.
- **Battery charging**: the battery is charged with the technique of *constant current-constant voltage* (CC-CV) until it reaches the full State of Charge.
- **Waiting for the relaxing time**: waiting for the battery relaxation period required to allow the electrochemical recombination processes within the batteries to reach equilibrium.
- **Impedance measurement**: at the end of the charging, the relax time has waited, and after, the EIS measurement is carried out.
- **Battery discharge of ΔSoC**: the battery is discharged, with a constant current, until the desired State of Charge is reached (ΔSoC).
- $\mathbf{V_B < V_{min}}$: checking the full discharge condition, the battery is considered fully discharged when the cut-off voltage is reached, and in that case, the characterization procedure is ended. In batteries, this represents the minimum voltage at which a battery is considered fully discharged and should no longer deliver power to prevent damage.

Fig. 5.3 shows the obtained results from the experimental characterization on a single battery. Typically, the obtained impedances are shown on a Nyquist diagram,

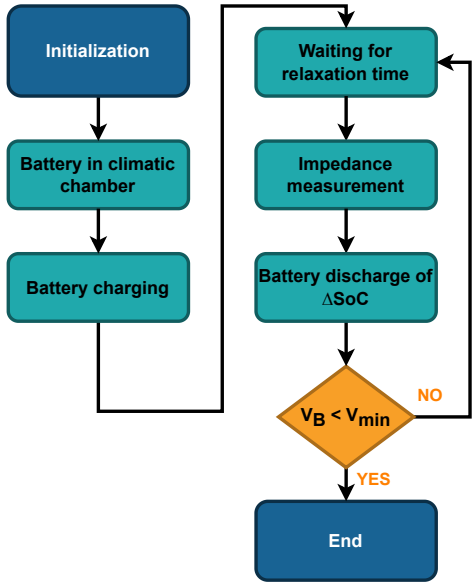


Figure 5.2: Implemented measurement procedure.

with the real part of the impedance plotted on the x-axis and the imaginary part (with the opposite sign) of the impedance plotted on the y-axis. It's clear that as well known in the literature, the impedance spectroscopy changes with the battery's SoC. For visual reasons, a subset of all the SoC is shown. An SoC-depended behavior in the inflection point (around 65 $m\Omega$ on $Re\{\dot{Z}\}$ axis) could be noted, as well as a slight and well-known in literature linear correlation between $Re\{\dot{Z}\}$ and SoC at high frequencies.

The dataset structure is depicted in Fig. 5.4. For every battery tested, 29 impedances are collected for each SoC level. Each measurement comprises two real numbers (a real and an imaginary part), resulting in 58 real values, potentially usable as features. The dataset's wideness regarding the number of batteries, number of SoCs, and measured frequencies make it perfect for the performed statistical analysis.

5.1.2 Description of the considered approach

A statistical approach was applied to the dataset obtained from the experimental campaign to identify possible correlations between the SoC and the impedance measurements. In particular, the Kernel Density Estimation (KDE) technique has

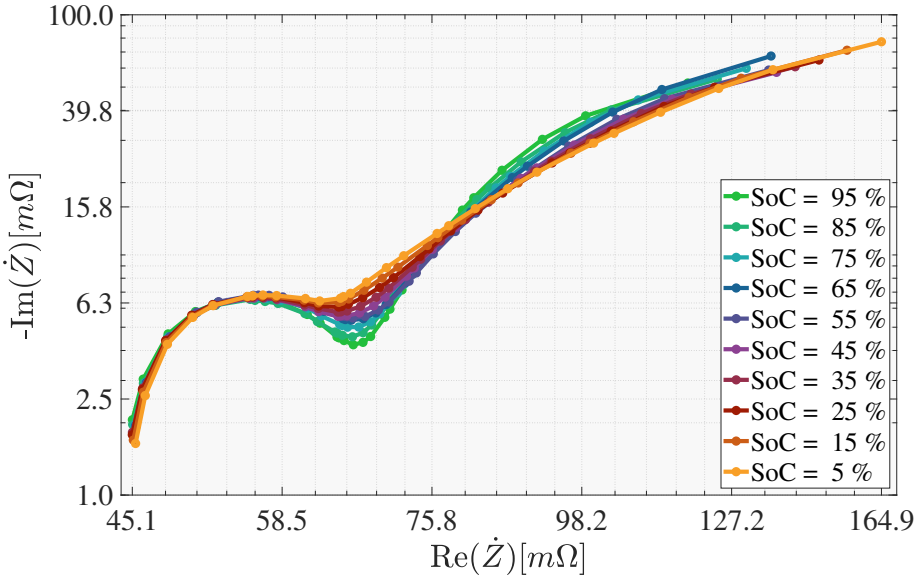


Figure 5.3: Nyquist diagram for a battery and a subset of measured SoCs.

been applied.

KDE is a statistical technique for estimating the Probability Density Function (PDF)) of a random variable [119]. In detail, KDE is a non-parametric technique that uses Kernel functions as weights to estimate the PDF of a random variable. For the KDE application, the variables should be expressed as quantitative nature variables (i.e., they should be expressed in numerical terms).

Specifically, the KDE technique was applied to the dataset described in Section 5.1.1 individually for each battery, for each SoC level, and by dividing the real part by the imaginary part. Therefore, for each battery and each SoC level, a PDF was estimated for the real part and another PDF for the imaginary part. To apply the KDE technique to a dataset, some operating parameters need to be specified. These parameters are listed below and detailed in this application case.

- Sample observations, i.e., the data on which to estimate the PDF; in this case, the input data are represented by measurements (real or imaginary, separately) at different frequencies.
- Type of Kernel function (e.g., normal, triangular, uniform, etc.); in this case, the selected Kernel function type is normal.

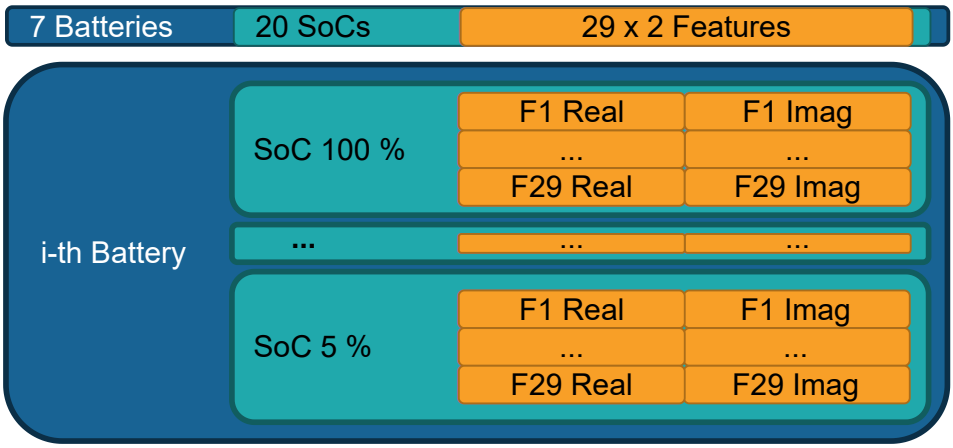


Figure 5.4: Structure of the obtained experimental dataset.

- Number of points to evaluate the PDF; we used 1000 points.
- Bandwidth, i.e., the parameter capable of varying the degree of approximation of the PDF; we chose the bandwidth value according to Silverman's rule [120].

Fig. 5.5 shows an example of the PDF estimated by the KDE. The x-axis shows the sample values, while the y-axis shows the PDF. The obtained PDF has two important features for this analysis: the "Peak" value and the "Width" value of the PDF calculated at half of the Peak value. The next Section shows a preliminary analysis aimed at identifying correlations between the Peak and Width parameters with respect to the SoC.

5.1.3 Preliminary results

Preliminary results of applying KDE to EIS measurements are discussed in this Section. First, considering a single battery, for example, the estimated pdfs from KDE for the real part and the imaginary part of the EIS measurements for some SoC levels are analyzed. Subsequently, the important features of the estimated PDFs (Peak and Width) are analyzed in detail to highlight correlations with the SoC.

Fig. 5.6 shows the obtained result by applying the KDE to the EIS measurements on the real part and the imaginary part on a single battery. The figure shows the estimated pdfs for a subset of available SoC values to increase readability. The

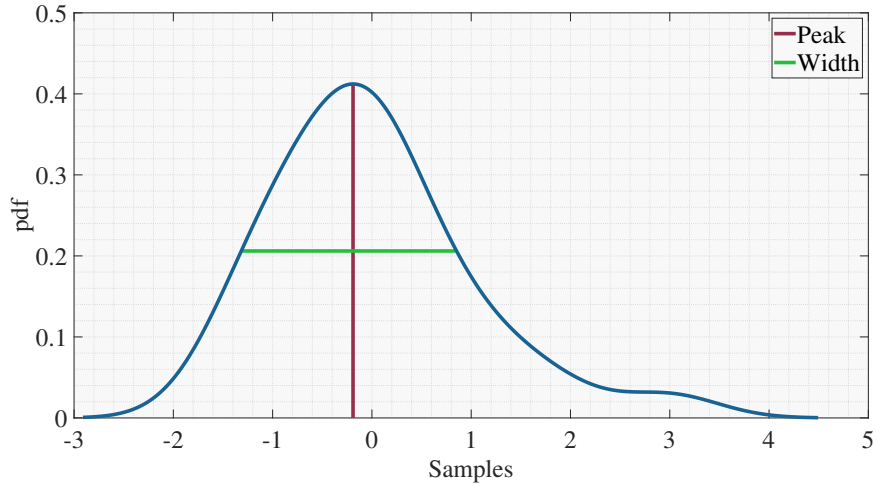


Figure 5.5: Example of the pdf estimated by the KDE technique.

estimated PDFs are characterized by a Peak value and a Width value calculated at half the Peak value. It is possible to note how for the pdf relative to the real part Fig. 5.6 (a), the Peak value obtained for each curve increases as the SoC increases. Conversely, the Width value shows an opposite behavior, i.e., it decreases as the SoC level increases. As for the obtained pdfs by applying KDE to the imaginary part of the EIS measurements Fig. 5.6 (b), the behaviors highlighted for the real part are less evident.

Including the estimated pdfs obtained for all SoC levels and all batteries, all Peak and Width values were identified and stored for each curve. Fig. 5.7 shows the Peak and Width behaviors with respect to the SoC for both the real and the imaginary part of the EIS measurement for all considered batteries.

In particular, Figs. 5.7 a) and b) show the Peak and Width trends as the SoC varies for the real part of the EIS measurements. It is possible to see that all batteries respect the same behavior. In detail, the Peak value increases as the SoC increases almost linearly, thus showing a potential positive correlation with the SoC. The same interesting aspect is found for the Width value, which decreases as the SoC increases almost linearly, thus showing a potential negative correlation with the SoC. Therefore, it is evident that there is a two-way link between the Peak and Width of the pdfs with the battery SoC.

Figs. 5.7 c) and d) show the Peak and Width trends as the SoC varies for the imaginary part of the EIS measurements. The Peak value shows an increasing

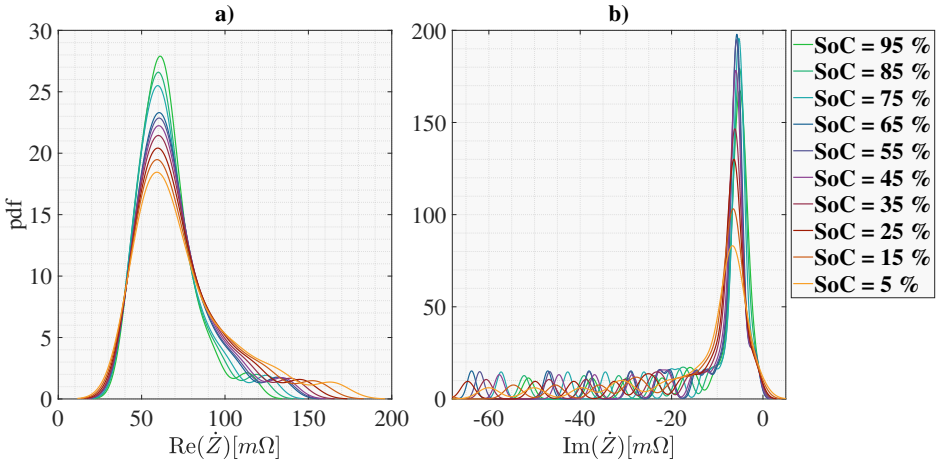


Figure 5.6: Output of the KDE technique applied to the a) real part and b) imaginary part of the EIS measurements.

linear behavior, as for the real part, for SoC values between 5 % and 35 %. Over 35 % the curves follow a decreasing trend with a lower slope. The Width value also shows a decreasing trend, as for the real part, for SoC values between 5 % and 35 %. Over 35 % the curves follow an increasing trend with a lower slope. Therefore, for both parameters, there is a trend reversal at SoC values of around 35%. In particular, the Peak behavior can be represented by a concave function with a maximum around a value of SoC close to 35 %. In contrast, the Width behavior can be assimilated to a convex function with a minimum around the same value of SoC.

The preliminary obtained results, both for the real and imaginary parts, have highlighted possible correlation schemes between the EIS measurements and the battery SoC. It should be noted that the obtained trends were found for all the examined batteries. The only differences between the batteries are the offsets in the obtained curves, presumably due to different manufacturing conditions or State of Health conditions due to the effect of calendar aging; both aspects are beyond the author's control.

5.1.4 Conclusions

In this preliminary study, new correlation patterns between impedance and SoC have been identified using exclusively statistical approaches, and we can consider this correlation to be established. In particular, the relationship between SoC and

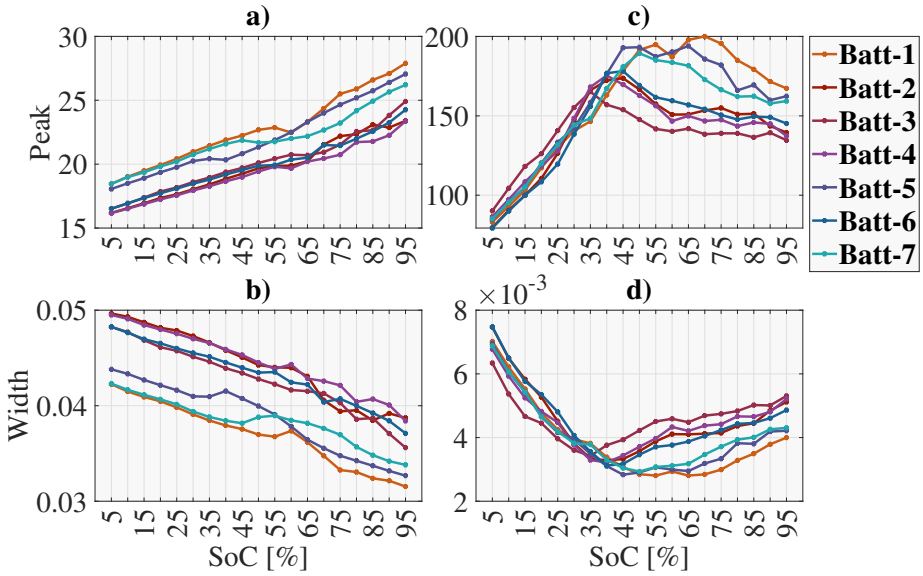


Figure 5.7: Output of the KDE technique applied to the EIS measurements: a) trend of the Peak value of the pdf with respect to the SoC for the real part, b) trend of the Width value of the pdf with respect to the SoC for the real part, c) trend of the Peak value of the pdf with respect to the SoC for the imaginary part, d) trend of the Width value of the pdf with respect to the SoC for the imaginary part.

the Peak/Width of the real component appears to be linear, creating a bijective relationship between them. At the same time, the relationship between SoC and the Peak/Width of the imaginary component is not bijective. Still, it represents possible information that can be combined with the real component's findings. To make the method applicable for real-time SoC estimation, however, further developments are needed. One area of optimization concerns the shape of the pdfs obtained from the KDE, intending to reduce the measurement time. To achieve this, it could be possible to improve feature extraction by minimizing the time spent on measurements (selecting the highest frequencies). Moreover, it is crucial to implement normalization schemes to standardize the various responses obtained from different batteries. This would make the presented method more suitable for use with data-driven approaches. Finally, machine learning techniques could be considered to estimate the SoC based on the parameters extracted from the pdfs, such as Peaks and Width. This could lead to accurate and reliable results in SoC

estimation, improving what is currently available in the scientific literature.

5.2 Battery Equivalent Series Resistance for Shuntless Coulomb Counting and SOC Estimation

Measuring the state of charge (SOC) of batteries is a crucial task for battery management systems, as it is a key part of the more general problem of battery state estimation, including state of health, state of energy, and state of power [121]. Several classes of methods for SOC estimation are available in the literature [122]. These can be classified as (i) direct methods, such as the open-circuit voltage (OCV) or impedance methods, which are based on a direct relationship between a battery parameter, e.g., the open-circuit voltage, and the SOC; (ii) coulomb-counting-based methods; (iii) tracking methods, such as Kalman-filter-based algorithms. Niri et al. [123, 124] proposed a different approach for predicting the remaining energy of a battery in an electric vehicle using load analysis with Markov models. Moreover, Gallien et al. [125] described an alternative SOC estimation method based on the relationship between magnetic susceptibility and SOC for lithium iron phosphate (LiFePo) batteries. Machine learning methods for SOC estimation have also been proposed, which are based on a relationship between the measured input quantities, i.e., voltage and current, and the SOC [126].

Coulomb counting is one of the most widely used methods for practical online SOC measurements [127]. This method consists of measuring the current that flows in and out of a battery and then integrating the measurement results over time. Using this method, the SOC at time instant t is calculated as

$$SOC(t) = SOC(t_0) + \frac{1}{C}\int_{t_0}^{t} I(\tau)d\tau, \tag{5.1}$$

where $SOC(t_0)$ is the SOC at time instant t_0 with $t_0 < t, I(\tau)$ is the current through the battery measured at time τ, and C is the capacity of the battery. When starting from a known initial SOC, this method allows to compute of the updated SOC at later time instants [128].

The main advantage of the coulomb counting method lies in its great practical value because it can be used in the operational phase while the current is flowing through the battery, e.g., while an electric vehicle is traveling. Conversely, direct methods such as that based on the open-circuit voltage cannot be used in the operational phase because, in such a phase, it is not possible to disconnect the battery from the load to read the open-circuit voltage. However, Movassagh et al. [128] showed that coulomb counting is affected by several uncertainty sources, namely

the current measurement uncertainty, the approximation used for current integration, the uncertainty of the battery capacity, and the timing uncertainty of the current sampling. The presence of such uncertainty sources represents the main disadvantage of the coulomb counting method. To mitigate the impact of some of these uncertainty sources, Baccouche et al. [129] proposed an improved coulomb counting algorithm based on a piece-wise model of the OCV-vs.-SOC characteristic curve and a periodic recalibration of the capacity. The method is experimentally shown to provide 2% accuracy for SOC estimates. Liu et al. [130] described an alternate method for SOC estimation, which combines current integration and an adaptive extended Kalman filter, shown to provide accurate results with a reduced computational complexity. Furthermore, Wang et al. [131] used a deep learning method for SOC estimation, which integrates coulomb counting with cloud-based convolutional neural networks and results in a root-mean-square error lower than 1.5% with a maximum error of 5%. Miao et al. [132] demonstrated experimentally that the combination of machine learning with coulomb counting may decrease the error for stand-alone coulomb counting in extreme temperature conditions. Moreover, the impedance track method described in [133] performs enhanced coulomb counting functionality by employing additional information, such as voltage and temperature measurements. The coulomb counting methods described above require current measurement, e.g., by means of an external shunt resistor. However, in several high-power applications, such as electric vehicles and energy storage systems, the usage of a shunt resistor implies a loss in power and efficiency since considerable energy is dissipated when a large current flows through this resistor [134]. Moreover, safety and reliability issues can occur due to a device being placed in the path of high current flow. Therefore, SOC measurement solutions that do not require a shunt resistor would provide numerous benefits, including a reduction in power consumption, a smaller size, a reduction of cost and complexity associated with the calibration of the shunt, and mitigation of safety and reliability issues. Thus, the need for developing and characterizing the accuracy of simple methods for shuntless SOC measurements arises. In this context, current-sensorless methods for estimating the SOC were proposed by Chun et al. [135]. These methods are based on filtering the voltage measured at the battery terminals and on fitting equivalent circuit models, and a maximum error of 5% was reported. In this work, a novel SOC estimation method based on coulomb counting is introduced, where the current is measured using the internal resistance of the battery itself, rather than an external shunt resistor. Contrary to published coulomb counting methods, the proposed method requires only the measurement of battery voltage and the usage of look-up tables that are pre-generated during a calibration phase containing information about the internal resistance and open-circuit voltage of

the battery. Differently from the current-sensorless approach in [135], the proposed method does not rely on a battery equivalent circuit model or on filtering techniques, thus resulting in reduced complexity and wide applicability, since it is not necessary to assume a specific equivalent circuit. In this work, the proposed method's feasibility is verified by applying it to experimental data, which consists of current and voltage measurements obtained using a commercial source measure unit (SMU) lab instrument on a lithium polymer battery under test following a realistic operational profile for electric vehicles. Although the feasibility study of the proposed method is carried out using lab instrumentation, the technique can potentially enable online cell-level condition monitoring for in situ and in operando applications [136], made possible by integrated cell management units (CMUs), such as the one proposed by Manfredini et al. [137]. Therefore, the main technical contributions of this work can be summarized as follows:

- A shuntless SOC estimation method is proposed that exploits the knowledge of the internal resistance of the battery under test;
- The proposed method is validated by experimental results performed on a lithium polymer (LiPo) rechargeable battery with a realistic current profile emulating an electrical vehicle scenario.

Further investigation into the achievable performance and reliability of the proposed method when applied to other batteries and different current profiles is beyond the scope of this work and is the subject of future research directions. The remainder of this section is organized as follows. In subsection 5.2.1, the proposed method is described. Then, in subsection 5.2.2, the experimental setup used to validate the method is presented, followed by experimental results. Finally, the results are discussed and conclusions are drawn in subsection 5.2.3.

5.2.1 Proposed Shuntless SOC Measurement Method

The proposed method aims to obtain an estimate of the SOC without using a shunt resistor to measure current. The only measurement that is required during operation is that of the battery voltage under load V_load. Furthermore, look-up tables (LUTs), obtained during a preliminary calibration phase, are employed. These LUTs provide information about how the equivalent series resistance (ESR), i.e., the internal resistance of the battery, and the open-circuit voltage (OCV) vary for different values of SOC. These LUTs are built during the calibration phase by measuring the battery's OCV, its voltage under load V_load, and the current for different SOC values with a reference instrument. A detailed description of the procedure to build the LUTs is given in section 5.2.2. The ESR, coupled with

the knowledge of the OCV, is used in place of the external shunt to estimate the current and perform coulomb counting in the operational phase. A block diagram illustrating the operation of the proposed method is shown in Figure 5.8.

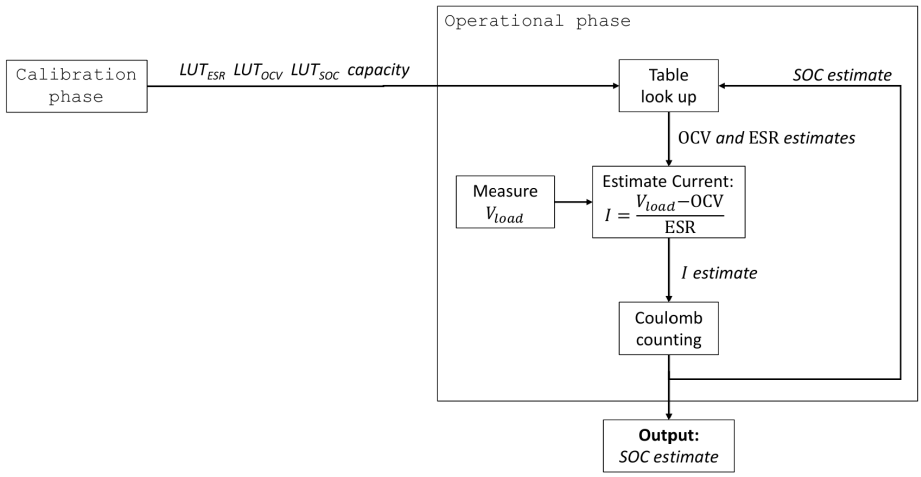


Figure 5.8: Operation of the proposed method, divided into two phases: calibration phase where the LUTs are built, and operational phase, where the LUTs are used together with voltage measurements to estimate the current and SOC.

The assumptions made for the proposed method are the following: (i) the capacity of the battery, the ESR-vs.-SOC, and the OCV-vs.-SOC relationships for the battery under test are known. These are measured in a preliminary offline calibration phase and encoded in LUTs; (ii) the SOC value at the initial time instant when the proposed method starts is known; (iii) measurements of the battery voltage are available during operation, while current is flowing through the battery. The pseudo-code in Algorithm 5.9 describes in detail the operation of the proposed method. Specifically, this method takes as input the required LUTs that are pre-generated during a battery characterization phase. Specifically, LUT OCV contains the OCV-vs.-SOC relationship, LUT ESR contains the OCV-vs.-ESR relationship, and LUT SOC serves as an index table. The method also requires an estimate of the battery capacity and returns as output an estimate of the SOC at every time step.

Starting from a known initial SOC, which here is assumed to be 100%, the algorithmis first initialized by measuring V load and obtaining an initial estimate of the current as $I = (Vload - OCV)/ESR$, where OCV and ESR are given by the respective LUTs. Then, at each iteration, coulomb counting is performed

Algorithm 1 Estimate SOC.

Input: (obtained in the calibration phase) Look-up-tables $LUT_{OCV,I_1}, \ldots, LUT_{OCV,I_n}$, $LUT_{ESR,I_1}, \ldots, LUT_{ESR,I_n}$, LUT_{SOC}, battery capacity C

Output: SOC estimate, denoted as $\widehat{SOC}$

begin
  Start condition: battery fully charged
  Measure $V_{load}(1)$ ▷ First voltage measurement
  $\hat{I}(1) \leftarrow \frac{V_{load}(1) - LUT_{OCV}(1)}{LUT_{ESR}(1)}$ ▷ Initial estimate of current
  $\widehat{SOC}(1) \leftarrow 100\%$ ▷ Initial estimate of SOC
  $k \leftarrow 2$
  while True **do**
    $\hat{q} \leftarrow \hat{I}(k-1) \times (t(k) - t(k-1))$ ▷ Charge increment (coulomb counting)
    $\widehat{SOC}(k) \leftarrow \widehat{SOC}(k-1) + \hat{q}/C$ ▷ Estimate SOC at time step k
    return $\widehat{SOC}(k)$
    Measure $V_{load}(k)$ ▷ Voltage measurement at time step k
    $j \leftarrow$ index of the entry of LUT_{SOC} closest to $\widehat{SOC}(k)$
    $\widehat{OCV} \leftarrow$ linear fit of $LUT_{OCV,I_1}(j), \ldots, LUT_{OCV,I_n}(j)$ eval. at $\hat{I}(k-1)$
    $\widehat{ESR} \leftarrow$ linear fit of $LUT_{ESR,I_1}(j), \ldots, LUT_{ESR,I_n}(j)$ eval. at $\hat{I}(k-1)$
    $\hat{I}(k) \leftarrow \frac{V_{load}(k) - \widehat{OCV}}{\widehat{ESR}}$ ▷ Estimate current at time step k
    $k \leftarrow k+1$
  end while
end

Figure 5.9: The pseudo-code for the Algorithm proposed.

using the previous estimate of I to obtain a SOC estimate, followed by an update procedure that uses the new measured value of V_load and table look-up operations to update the estimate of I. Since the OCV-vs.-SOC and ESR-vs.-SOC curves vary depending on the current flowing through the battery, in the calibration phase, the LUTs are built for n different values of the current $I1, ..., In$, thus resulting in $LUT_{OCV,I1}, ..., LUT_{OCV,In}$. Then, in the operational phase, at each time step k, the best-fit line (in a least-squares sense) on the data in $LUT_{OCV,I1}, ..., LUT_{OCV,In}$ is determined, thus obtaining a linear function $f(I)$ relating current to OCV. This operation allows for using the LUTs for any generic current profile so that there is no need to use different LUTs for different current profiles. An estimate of OCV is then obtained by evaluating this function for the current estimated at the previous time step, i.e., $\widehat{OCV} = f\left(\hat{I}(k-1)\right)$. As described in Algorithm 1, the estimates of the ESR are obtained using the same linear regression technique as that used for the OCV. Note that, in principle, the SOC may be estimated by simply inverting the OCV-vs.-SOC curve. However, this inversion requires that the OCV of the

battery under test is measured after the relaxation period when the current is not flowing through the battery. This makes it unusable in online scenarios. Conversely, the proposed method can be employed online during battery operation. A known initial SOC is a precondition for the proposed method. In a practical scenario, this precondition is satisfied when the battery is fully charged, i.e., after the standard charging procedure described by the manufacturer is completed. In this case, the known SOC is equal to 100%. Alternatively, when the battery is not used for a time interval longer than the relaxation period, the proposed method can still be used if the SOC is estimated using other methods, such as referring to the OCV-vs.-SOC curve or by electrochemical impedance spectroscopy. Finally, in practical applications, the calibration procedure should be repeated periodically to account for the variations in the battery's behavior due to aging.

5.2.2 Experimental SOC Estimation Results

To validate the proposed shuntless SOC estimation method, we performed experiments on a Mikroe SR674361P, 3.7 V LiPo pouch battery. As specified in the datasheet [138], this battery has a nominal capacity of 2000 mAh a rated voltage of 3.7 V with a limited charge voltage of 4.2 V and a discharge cutoff voltage of 2.75 V. The specified internal resistance is less than 160 $m\Omega$, as measured by the manufacturer at 1 kHz. The following paragraphs present the experimental setup, the experimental procedure, and the obtained results.

Experimental Setup

The experimental setup consisted of a bench-top instrument, the Keithley 2450 source measurement unit (SMU), which provides current signals and simultaneously measures current and voltage, connected via a 4-wire configuration to the battery under test. The SMU operation was timed and controlled by a custom-developed script written in the Test Script Processor (TSP) language. The measurement data acquired by the SMU were transferred to a PC, which was processed by custom routines implemented in the MATLAB numerical computation environment. During the experiments, the battery was placed in a small climate-controlled chamber at 25 ± 1 °C temperature. Before the experiments, the battery was preconditioned by cycling it five times at 1 A (0.5C). Note that similar preconditioning procedures are indicated by several standards for battery testing, e.g., the International Electrotechnical Commission (IEC) 62660-1 standard on "Secondary lithium-ion cells for the propulsion of electric road vehicles—Part 1: Performance testing" [139].

Experimental Procedure

Following the structure of the proposed method described in subsection 5.2.1, the measurement procedure used in the experiments was divided into two phases: the calibration phase and the operational phase, described in detail in the following.

Calibration Phase

During this preliminary phase, the LUTs were built. The LUTs consisted of the OCV-vs.-SOC and ESR-vs.-SOC curves, which were obtained by starting from a fully charged battery and discharging it with a constant current I_0. During the discharge, V_load and $_0$ were constantly measured by the SMU. Every two seconds, the current was also interrupted by the SMU to measure the open-circuit voltage (OCV) of the battery after a 10 ms transient period. Using the available measured quantities, the battery internal resistance was estimated as $ESR = (V_load - OCV)/I_0$, as in the widely adopted current-pulse method for battery internal resistance measurement [140]. The procedure was stopped when the voltage across the battery reached the cut-off value of 2.75 V specified by the manufacturer. The battery capacity was also measured during this stage by integrating the current. The discharging procedure was repeated four times, each time with a different value of the constant discharge current: 1 A, 750 mA, 500 mA, and 250 mA. These current values correspond to the following C-rates: 0.5C, 0.375C, 0.25C, and 0.125C.

Operational Phase

After the calibration phase, an experiment was performed to validate the proposed method in the operational phase. In this stage, the SMU generated a time-varying current that emulated the behavior of an electric vehicle as defined by the Worldwide Harmonized Light-Duty Vehicles Test Procedure (WLTP) [141]. In particular, this current profile is publicly available in [142], and it was obtained from the WLTP speed profile by simulating a light electric vehicle. For practical reasons, in this work, the WLTP current profile in [143] was normalized so that its maximum absolute value was 1 A, which corresponds to the largest current that the SMU lab instrument can generate. The resulting WLTP current profile, which has a duration of 1800 s, is shown in Figure 5.10. The SMU was programmed to iteratively generate the WLTP current profile until the battery was fully discharged, which took approximately 13 hours, corresponding to 25.8 iterations of the WLTP profile.

In the operational phase, the V load measurement results, acquired by the SMU with a sampling period of 1 s, were fed to the proposed method in Algorithm 1, together with the LUTs from the previous calibration stage. This emulates a practical scenario where the only available measurements are those of the voltage

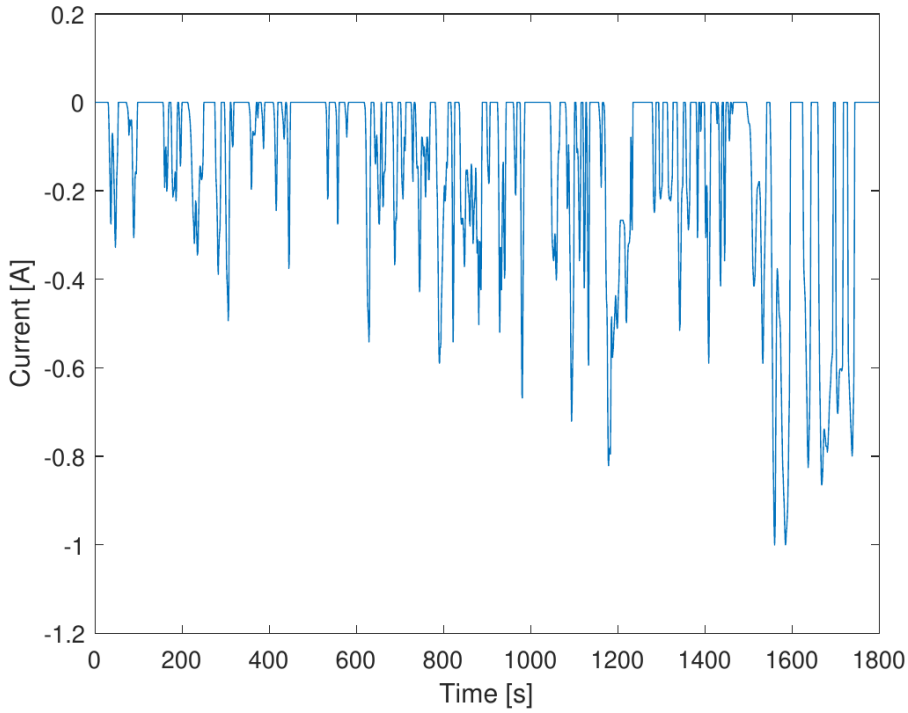


Figure 5.10: WLTP current profile used to validate the proposed method in the operational phase.

at the battery terminals. The current measurements performed by the SMU were not used by the proposed method but were used to calculate the ground-truth reference to evaluate the performance of the proposed method. Therefore, the reference SOC at each time step was obtained by integrating the current measured by the SMU. The SOC estimation error e of the proposed method was calculated as the difference between the estimated SOC and the reference SOC, i.e.,

$$e(k) = \widehat{\mathrm{SOC}}(k) - \mathrm{SOC}(k) \tag{5.2}$$

where k is the time step and N is the number of samples in the acquired record of the signals. The root-mean-square error (RMSE) of the SOC estimation is defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\widehat{\mathrm{SOC}}(k) - \mathrm{SOC}(k)\right)^2}. \tag{5.3}$$

Results

The results of the preliminary calibration phase are shown in Figure 5.11 and Figure 5.12, where the curves are measured for the following values of the current: 250 mA (0.125C), 500 mA (0.25C), 750 mA (0.375C), and 1000 mA (0.5C). Furthermore, the capacity measured during calibration with a C-rate of 0.5C is 1.94 Ah. The curves in Figure 5.11, showing the behavior of OCV for varying SOC, were used to build LUT_{OCV}. Instead, the curves in Figure 5.12, showing the behavior of ESR for varying SOC, were used to build LUT_{ESR}.

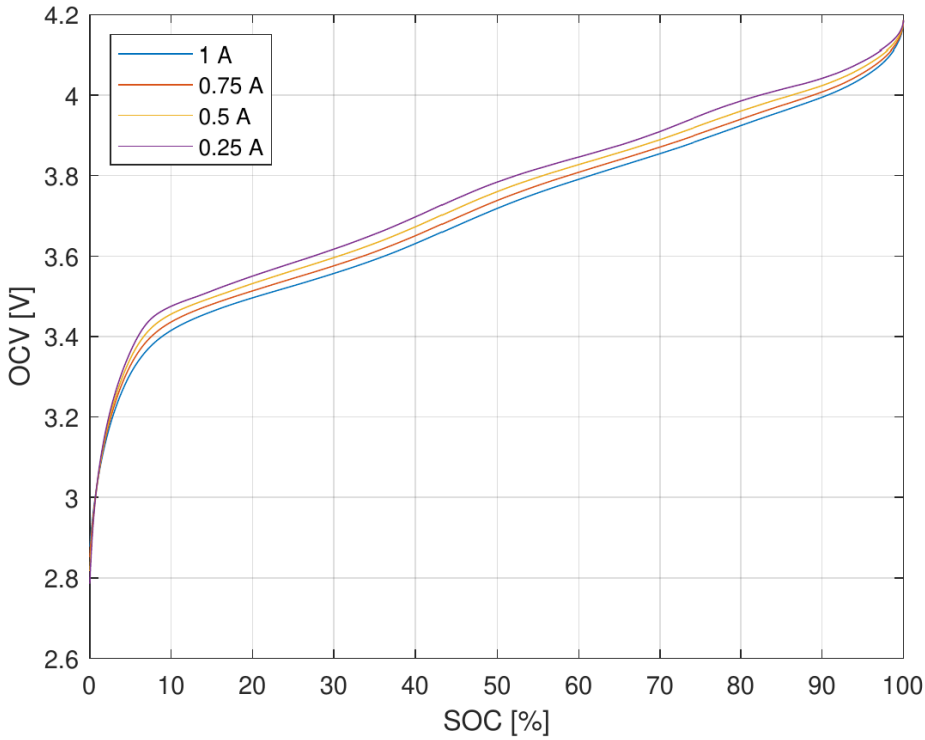


Figure 5.11: Experimental data: OCV as a function of SOC. The curves were obtained in the calibration phase for different values of the discharging current.

An example of the fitting operation performed at each time step in the operational phase is shown in Figure 5.13, where a linear function is fit to the OCV data measured at the same SOC for currents of 250 mA, 500 mA, 750 mA, and 1000 mA during the calibration phase and encoded in the LUTs. The good fit observed in this figure confirms that, for the proposed method, the behavior of the OCV

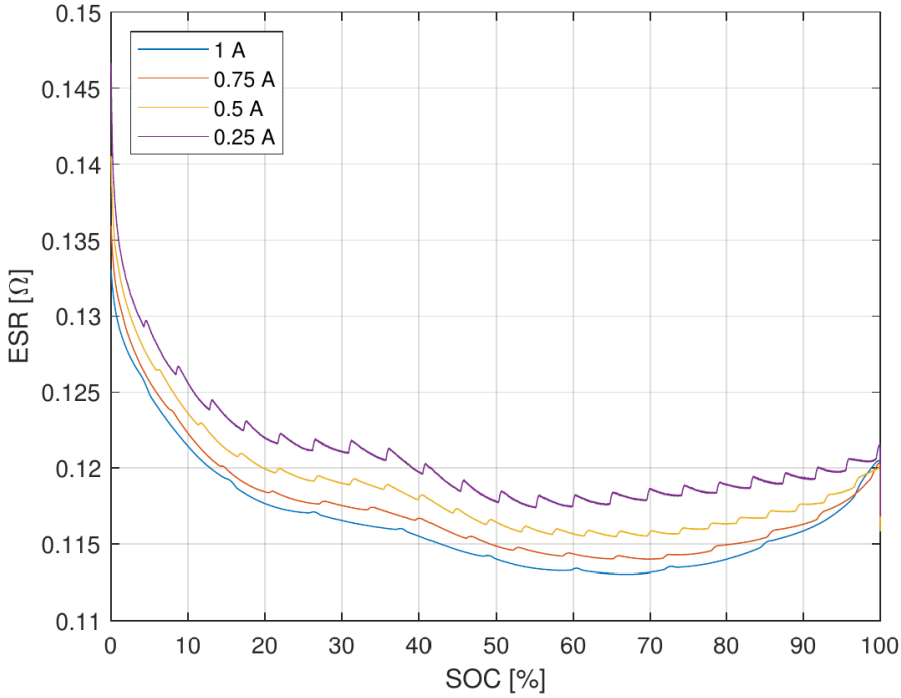


Figure 5.12: Experimental data: ESR as a function of SOC. The curves were obtained in the calibration phase for different values of the discharging current.

can be approximated to be linear with a varying current in the considered current range.

Furthermore, the results obtained in the operational phase with the realistic WLTP current profile are shown in Figure 5.14 and Figure 5.15. Figure 5.14 shows the estimated current as well as the reference current, i.e., that measured by the SMU. From Figure 6b, a good agreement between the estimated current and the reference current can be observed. The current error in Figure 6c is defined as the difference between the estimated current and the reference current according to the definition provided by the International Vocabulary of Metrology (VIM) [144]. Specifically, the root-mean-square difference between the estimated current and the reference current is approximately 64 mA, i.e., 6.4% of the maximum considered current. This error is relatively small, and the spike near the end of the cycle visible in Figure 6a, where the error reaches approximately 90% for a brief period of a few seconds, does not noticeably affect the RMSE computed over the entire cycle. Finally, the estimated SOC is shown in Figure 7, together with

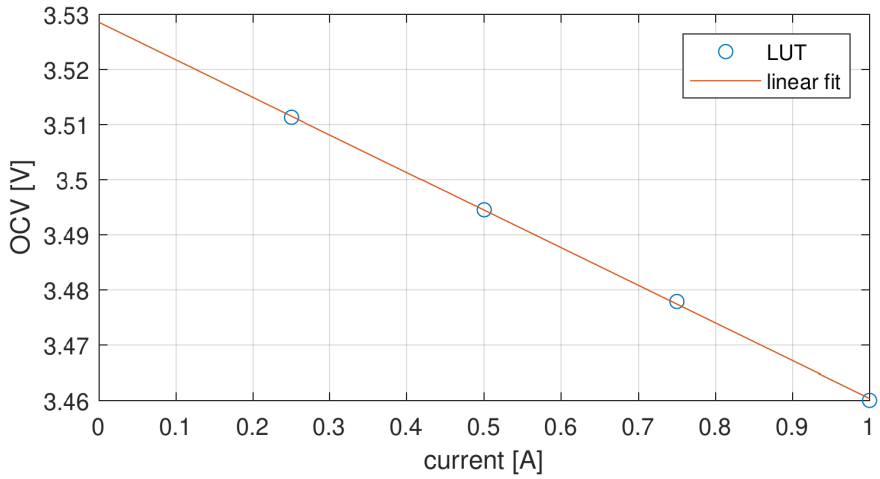


Figure 5.13: An example of the best-fit line computed during the operational phase to obtain an estimate of OCV from an estimate of the current. The fit was performed on four data samples from the LUTs acquired at different currents: 250 mA, 500 mA, 750 mA, and 1000 mA.

the corresponding error. The RMSE of the SOC estimation is 0.82%, and the maximum SOC estimation error is 1.45%.

5.2.3 Discussion and Conclusions

The results presented previously prove the feasibility of the proposed shuntless SOC estimation method, with an accuracy that is sufficient for the most common battery monitoring applications (RMSE 0.82%). Such results were obtained by emulating a real-world scenario in the electrical vehicle field. Therefore, they indicate that the proposed shuntless method can be employed for in operando and situ applications. The SOC estimation literature contains other methods achieving errors of less than 1%, such as the proposed method, but most of these methods require the usage of an external shunt. Instead, the proposed method requires only the measurement of the voltage at the battery terminals and the usage of LUTs obtained via calibration. Therefore, it may overcome the energy consumption, reliability, and safety issues associated with the usage of an external-shunt-resistor-based current monitoring system. When comparing the results with the current sensorless method presented in [135], the achieved SOC error is smaller. Specifically, a maximum error of 1.45% is observed in this work, whereas a maximum error of 5% was reported in [135]. However, the proposed method follows a different approach

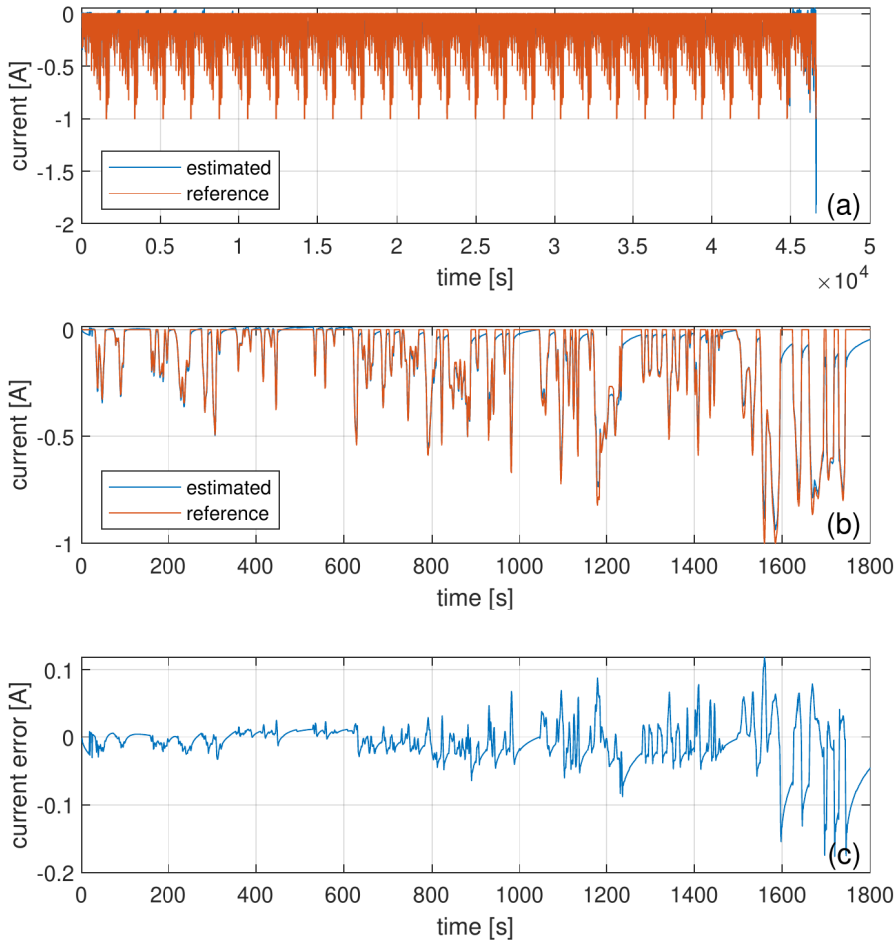


Figure 5.14: Experimental results in the operational phase. (a): Current estimated by the proposed shuntless method with a WLTP profile; (b): magnification of the curve in (a), showing a single repetition of the WLTP profile; (c): current estimation error, computed as the difference between the estimated and reference value of the current.

as it does not rely on a battery equivalent circuit model and it does not use filtering techniques. Thus, it potentially has a smaller implementation complexity and wider applicability since it is not necessary to assume a specific equivalent circuit for the battery. Instead, it relies on a preliminary calibration of the battery under test. Therefore, it can be easily applied to different battery types and

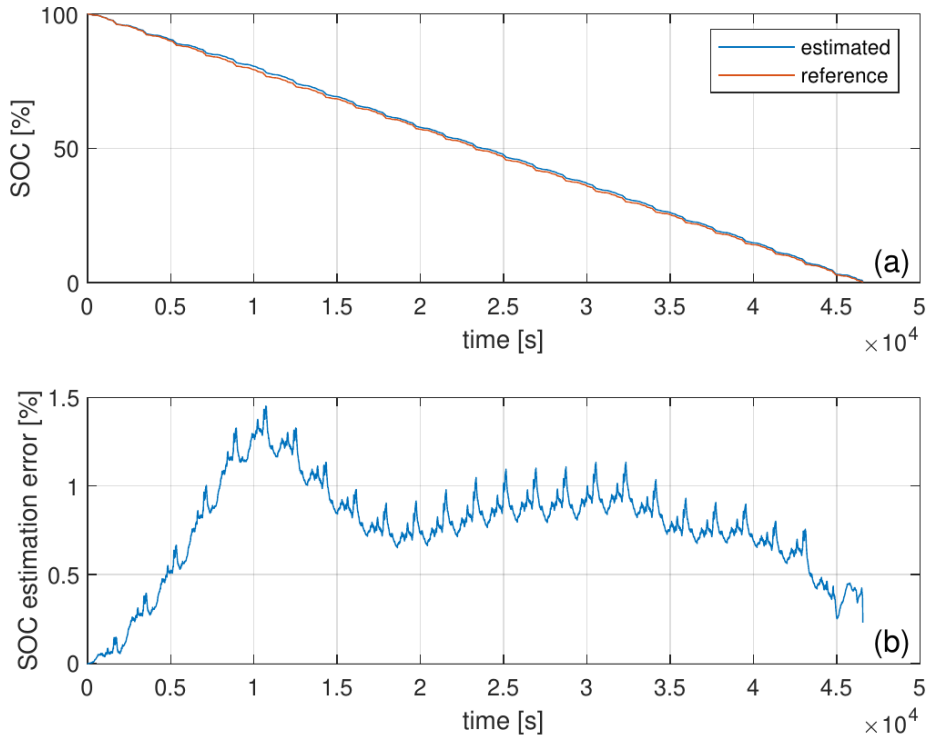


Figure 5.15: Experimental results in the operational phase: (a) SOC estimated by the proposed shuntless method; (b) estimation error.

various battery-powered systems. Some factors that could potentially limit the applicability of the proposed method are the differences between specific battery samples, the influence of aging, and the effect of the operating temperature. To compensate for the first factor, i.e., the variations in the OCV and ESR profiles between individual battery cells, the calibration procedure described in this work should be applied individually to every battery of interest. Moreover, to mitigate the influence of battery aging, which modifies the OCV and ESR behavior with SoC, such calibrations should be repeated periodically as part of the maintenance process throughout the battery life-time, e.g., the battery should be re-calibrated after completing a certain number of operational cycles. Finally, temperature is another factor that affects the discharge profile of a battery by changing its internal resistance [145] and OCV characteristics [146]. A future development of the proposed method should therefore take into account the effect of temperature. This

could potentially be done by acquiring the LUTs at several different temperatures during the calibration phase and then employing measurements from additional temperature sensors in the operational phase. Depending on the application, such sensors could already be present in the battery-powered system of interest.

In this work, the possibility of using information about a battery's equivalent series resistance to estimate its state of charge without directly measuring the current was investigated. In particular, a method for estimating the SOC of batteries without using a shunt for coulomb counting was presented, and its feasibility was investigated experimentally. The proposed method resulted in a SoC estimation RMSE of 0.82% in a realistic use case where a WLTP current profile was applied to a lithium polymer cell. Therefore, its feasibility is confirmed. The described technique holds potential with the new cell-level sensors and distributed Battery Management System (BMS), such as the ones proposed by the cell management unit (CMU) from Sensichips [137] since individual cell re-calibration and measurement are made possible. Potential industrial and practical applications of the proposed method are in the electric vehicle and stationary energy storage fields. In these fields, the absence of a shunt resistor would avoid the energy loss associated with large currents flowing through the shunt resistor and thus improve efficiency and safety. Future developments include using Electrochemical Impedance Spectroscopy (EIS) information for building OCV and ESR LUTs, which may provide benefits in terms of AC impedance characterization over a wide range of frequencies and allow the management of fast variations of the current. Additionally, applying the proposed technique to several cells connected in series, as in battery packs, may improve accuracy when estimating the SOC of the overall battery pack. Additional future perspectives include further tests using other current profiles and on different batteries, investigations on the effect of smoothing and compression of the LUTs, and the implementation of the proposed method in an embedded processor for real-time in situ applications.

5.3 New method for EIS-based SoC estimation optimizes accuracy and time.

The crucial task of a Battery Management System (BMS) is to provide security and longevity of the battery. This can be done through the accurate and continuous monitoring and control of the battery's State of Charge (SoC) and State of Health (SoH). The accurate and real-time knowledge of these parameters, see [147, 148], allows to accomplish some crucial tasks such as the estimation of the remaining life range, the battery cell balancing [149], the temperature monitoring and control, and so on. Regarding the SoC estimation, the most used techniques are based on the

Coulomb counting and the Open Circuit Voltage (OCV). The Coulomb counting is the current integration over time [150]; in this case, the measurement error and knowledge of the initial SoC are huge problems. Also, Lithium Iron Phosphate (LFP) batteries have a flat relationship between voltage and SoC, meaning that a small error in voltage measurement generates a large error in SoC estimation; this is the main disadvantage of the OCV–based method. Electrochemical impedance spectroscopy (EIS) could be a good candidate to replace Coulomb/OCV counting techniques in SoC estimation [151, 152]. It does not suffer from the problems just discussed and has the potential advantage of being able to be applied online in real operating conditions [153]. EIS techniques require generating a stimulus signal (usually sinusoids) and measuring the battery voltage and current. The stimulus signals frequencies are in the range between a few mHz to a few kHz [154, 155]. The main problem with EIS concerns precisely the big measurement time, which is related to using the lower frequencies of the stimulus signals. They greatly increase the measurement time by reducing the possibility of being able to estimate the SoC in real–time.

This work tries to find a solution to this issue, proposing a first measurement method able to minimize the measurement time given a target Accuracy in SoC estimation when an EIS approach and knowledge based on SoC classes are considered. Accordingly, this work aims to provide methodologies for optimizing the EIS used for battery SoC estimation. Given the generality of the proposed approach, it is applicable in all contexts where the problem of designing a system for real–time SoC estimation using the EIS technique is addressed. Depending on the requirements of different applications, such as the accuracy to be achieved and the measurement time required to perform SoC estimation, the proposed method allows optimizing the choice of frequencies used in the EIS to satisfy the constraints imposed. Potentially, considering the three parameters of interest in the SoC estimation such as the number of classes in which SoC is divided, the target Accuracy and the measurement time, the proposed method can optimize two of these parameters by fixing the third one. For example, having fixed the number of classes in which SoC is divided, the method allows maximizing the classification Accuracy without constraints on measurement time, or minimizing measurement time without constraints on the classification Accuracy, and finally optimizing in the sense of finding a trade–off between the classification Accuracy and measurement time when there are constraints on both due to specific applications. The method is based on two operating stages: i) the experimental characterization of the real behavior of the considered batteries, ii) the time–Accuracy optimization based on suitable feature selection and Machine Learning (ML) approaches. As for the ii) point some considerations can be addressed. In EIS, stimulus frequencies

in the mHz to kHz range are typically used, and the number of frequencies is at the operator's discretion. The greater the number of frequencies, the better the spectrum knowledge but the longer the measurement time. The selection of the minimum number of measurements at the key frequencies is the crucial point of this optimization. Adopting a ML approach, we can assume that each measure, at a different frequency, can be considered a feature of the algorithm. Then this problem can be solved with a feature selection (FS) approach. FS reduces the number of input variables when developing a predictive model. The benefits of FS include improved model performance, reduced overfitting, increased interpretability, and reduced computational complexity [156].

The approach based on FS was also proposed in [157], where the authors acquired a dataset of 54 frequencies on Li-ion batteries at five different temperatures. They then used a Pearson's Correlation Coefficient–based method to extract the frequencies most correlated with the SoC to use as input for a ML model and linear regression. In their approach, it is unclear which frequencies are selected and how much the selection impacts the reduction in measurement time. In addition, as better shown in the work, correlation offers lower selection performance than other proposed fitness functions. Other applications of FS concern the health status of batteries. In [158], the authors performed FS from voltage temperature and current measurements to identify indicators most correlated with the health status of LFP batteries. The batteries are aged through 121 equivalent charge and discharge cycles. As features, the authors use the time the battery has to spend between two voltage thresholds, temperature, current, and energy measurements. In the end, 69 features were obtained. After Gray's relationship analysis, the features correlated to SoH were sorted from least to most. By defining a correlation threshold, selecting the most important features is possible.

The measurement method proposed in the work extends the research activity presented in [115], aimed to minimize the time required to perform EIS through a feature selection technique based on Genetic Algorithms (GA). We have extended that research activity in several directions. To better validate our results, we augmented the experimental data by considering more batteries than those used in [115]. We have also investigated the effectiveness of the Particle Swarm Optimization algorithm [159] as a search strategy alternative to GA–based techniques [160]. Furthermore, we have implemented two fitness functions. The first was used to evaluate the performance achieved by training a supervised learning model (wrapper approach). In this case, we tested several ML models to choose the best–supervised learning model. On the other hand, the second fitness was based on a correlation analysis using Pearson's Correlation Coefficient. The experimental results demonstrated the effectiveness of the FS approach proposed; we achieved similar or even

higher SoC estimation accuracies than using all available features (i.e., not making feature selection) but with significantly shorter measurement times. The selection of features with the highest Accuracy allowed a huge reduction of the measuring time, from 4 minutes with all features to 1.5 seconds with the selected features. Furthermore, FS it is also important to note that some features are misleading for the classification algorithms. Therefore, their elimination allowed for a gain in the SoC evaluation Accuracy. Indeed, with all features, the maximum Accuracy for the SVM classifier is 0.87, and the same classifier with the selected features has a maximum Accuracy of 0.98.

The innovative points of the work are listed below.

- First, a FS–based experimental method was developed to optimize EIS for real–time battery SoC estimation applications. The proposed method has general validity and can be applied during the design of a battery SoC estimation system, regardless of the application context.
- Subsequently, to experimentally validate the proposed approach, an intensive experimental campaign was carried out on LFP batteries. This activity allowed the authors to obtain a dataset for characterizing batteries at different stimulus frequencies and SoCs. The authors intend to share the dataset with the scientific community in future publications.
- Finally, the proposed method achieves better SoC estimation performance than using all features, significantly improving the measurement time.

This work is organized as follows. Section 5.3.2 reports the experimental campaign describing the set–up and the measurement protocol used to create the dataset. Section 5.3.3 describes the conducted preliminary tests to choose the best-supervised learning model to consider within one of the fitness functions. Section 5.3.5 explains the implemented procedures to perform the Feature Selection. Finally, Section 5.3.6 shows the obtained results, and Section 5.3.9 provides the conclusions of the work.

5.3.1 The proposed method

Adopting impedance spectroscopy as an SoC evaluation method in Battery Management Systems (BMS) allows systems to avoid the use of Coulomb counting or Open Circuit Voltage. The latter methods, by their nature, tend to diverge over time or be less reliable during battery charge or discharge phases. However, impedance spectroscopy has a major limitation represented by the time required to make a measurement. This is mainly due to the low inspection frequencies that

result in measurement times on the order of minutes. Such long times may not be compatible with some applications. To reduce the measurement time without significantly compromising performance in SoC estimation, a data-driven methodology for SoC estimation is proposed. In detail, the proposed method addresses the problem of designing a system for real–time SoC estimation using the EIS technique. Depending on the requirements of different applications, such as target Accuracy to be achieved and measurement time required to perform SoC estimation, the proposed method allows optimizing the choice of frequencies to be used in EIS in order to meet the target constraints imposed. In this method, optimal frequencies are selected using feature selection approaches based on search algorithms and different fitness functions.

The proposed method follows the scheme shown in Figure 5.16. The first step is to define the design parameters and constraints listed below.

- Resolution of SoC output estimation: the higher the resolution the longer the characterization step and in general the greater the number of classes and the greater the classification error.
- Target measurement time: the target measurement time is the maximum acceptable time to obtain the measurement.
- Accuracy target: is the minimum acceptable Accuracy of the classifier for SoC estimation.
- The type of batteries: the characteristics of the batteries you want to use (chemistry, nominal capacity, nominal voltage etc.).
- Classifier: the proposed method uses a classifier built into the search algorithms to estimate the population. The choice of classifier is critical because the next optimization step will be stitched around the classifier, in other words, the features selected also depend on the classifier adopted.

The classifier is sometimes defined by design parameters related to hardware requirements, while at other times the choice of the classifier is related solely to classification performance. The proposed method addresses both of these needs; the classifier can be an input parameter or a preliminary step will be used to identify the most suitable classifier for the dataset. The adopted classifier will constitute the data–driven method used by the BMS to estimate the SoC.

The second step is the characterization of the devices under test, in the characterization the tougher parameters possible are adopted, so the resolution in SoC must be the best possible, as well as the number of features measured. The batteries that make up the database must be chosen according to the future application

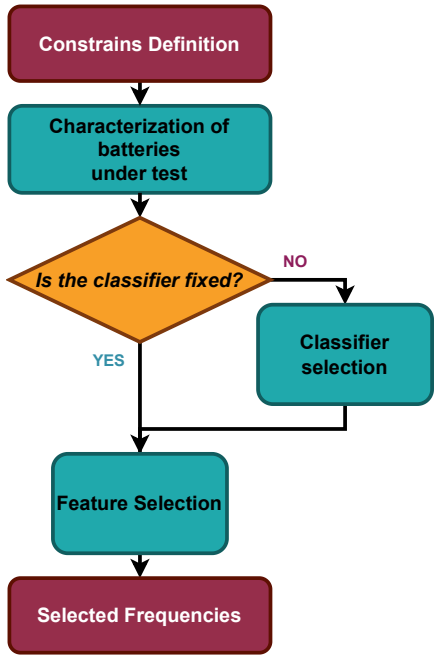


Figure 5.16: The proposed method workflow.

of the BMS. Using different batteries of the same model allows generality to the different batteries, just as adopting batteries at different states of Health (SoH) allows the system to be proof to different SoH. If the classifier does not fit within the design constraints at this stage, the classifier best suited to the data set acquired in the previous Section is evaluated. The classifier that provides better Accuracy among a set of selected classifiers will then be integrated into the feature selection algorithm.

The last stage of the proposed method is feature selection, where search algorithms can be used to select features with the dual goals of decreasing the measurement time to a value equal to and minor to the target time and, at the same time, having an Accuracy greater than the required target.

5.3.2 The proposed method: characterization of batteries under test

This Section discusses the experimental approach used to obtain the dataset. An experimental campaign was conducted to evaluate the impedance of LFP batteries

during their discharge. EIS technique was used to perform the impedance measurement while the reference SoC was estimated by the Coulomb counting method based on integrating discharge current throughout the battery discharge process.

Experimental set–up

From the authors' experience in designing experimental systems [161, 162], Figure 5.17 shows the schematic diagram of the developed experimental set–up to experimentally characterize the impedance of LFP batteries as the SoC varies during the discharge process. In detail, the experimental set–up consists of the following components:

- a Hioki BT4560 battery impedance meter to measure battery impedance;
- a ZKEtech EBD–A20H electronic load used to discharge the battery with a controlled current;
- a climatic chamber based on Peltier modules to control and set the environmental temperature around the battery;
- a PC to manage the process by controlling the instruments via USB interfaces.

Finally, the impedance meter–battery and the electronic load–battery connections are made with the four–wire technique to reduce the impact of parasitic phenomena.

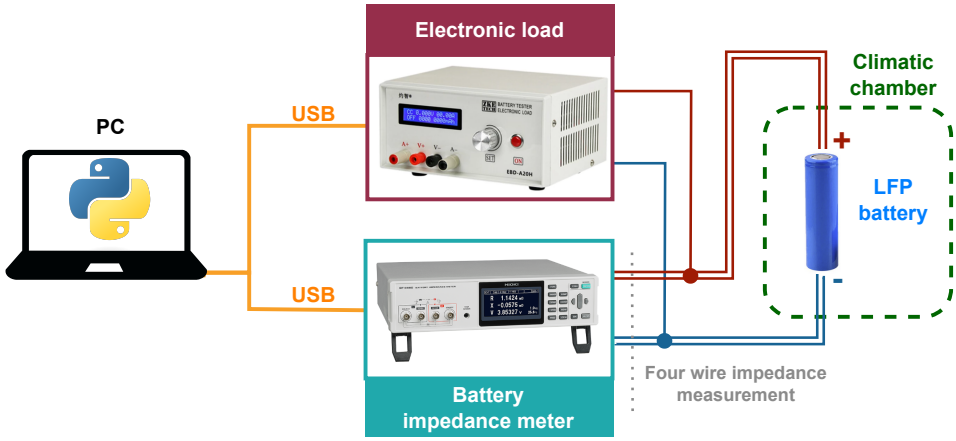


Figure 5.17: Implemented experimental set–up.

The implemented experimental set–up was used to characterize the behavior of 7 LFP batteries during their discharge process. The main characteristics of the considered batteries are summarized in Table 5.2.

Table 5.2: Main characteristics of the considered batteries.

Battery features	Values
Chemistry	Lithium Iron Phosphate (LFP)
Nominal capacity	600 mAh @ 0.2C Discharge
Nominal voltage	3.2 V
Minimum voltage	2.0 V
Charging voltage	3.65 V
Maximum continuous C-rate Discharge	2C
Maximum operating temperature	-10°C – 45°C (Charging) -20°C – 60°C (Discharging)

Measurement protocol

To characterize the behavior of the batteries through impedance measurement during their discharge process, a measurement protocol schematized in Figure 5.18 was implemented. This procedure was performed for all seven considered batteries. A detailed description of the measurement protocol is given below.

i) **Initialization**: initially, all the necessary conditions for performing battery characterization are set. These settings concern the characterization temperature, the relaxation time required for electrochemical recombination, the impedance measurement (the range and number of frequencies to be analyzed), the current used during battery discharge, and the SoC variation (ΔSoC) with which to perform the characterization. These conditions are summarized in Table 5.3 and detailed in the description of the measurement protocol.

ii) **Place the battery in the climatic chamber**: after reaching its thermal regime at a temperature of 20 °C, the battery is placed inside the climatic chamber.

iii) **Battery charging with CC–CV protocol**: at this stage, the battery is charged until it reaches the fully charged state (SoC = 100 %). This operation follows the standard Constant Current–Constant Voltage (CC–CV) protocol [163].

Table 5.3: Adopted conditions for the experimental characterization.

Settings	**Values**
Characterization temperature	20 °C
Relaxation time	15 min
Frequency range	$[10mHz; 1kHz]$
Number of adopted frequencies within the considered frequency range	28
Discharge current	120 mA
ΔSoC	5%

iv) **Waiting for the relaxation time**: to enable electrochemical recombination processes, it is important to add an interval between two consecutive discharge phases during the characterization phase. The relaxation time is the time required to reach equilibrium. The relaxation time is not unique but depends on the particular technology used. The adopted relaxation time for characterization was set to 15 minutes, as shown in Table 5.3. This result was achieved by performing several EIS experiments after discharging the battery and evaluating the measuring compatibility.

v) **Impedance measurement**: battery impedance measurement using EIS is performed with the settings shown in Table 5.3.

vi) **Battery discharge of ΔSoC**: the battery is discharged with a constant current of 120 mA until the SoC variation (ΔSoC) of 5 % is achieved.

vii) $\mathbf{V_B < V_{min}}$: here it is checked whether the battery voltage (V_B) has reached its minimum value (V_{min}). If not, the steps iv)–vii) are repeated; however, if the condition is met the battery is completely discharged. The manufacturer provides the minimum voltage value and is 2 V, as shown in Table 5.2.

viii) **End**: the characterization process is complete.

As an example, Figure 5.19 shows the obtained result of the experimental characterization for a single battery. In particular, the Real and Imaginary parts of the EIS measurements obtained for all the analyzed SoC are shown. As can be seen from the figure, the measurement with SoC equal to 100 % has a very different behavior from the other measurements. This can be explained by analyzing what happens at an electrochemical level. With high values of SoC (close to 100 %), the concentration of lithium ions in the positive electrode (cathode) is small [164].

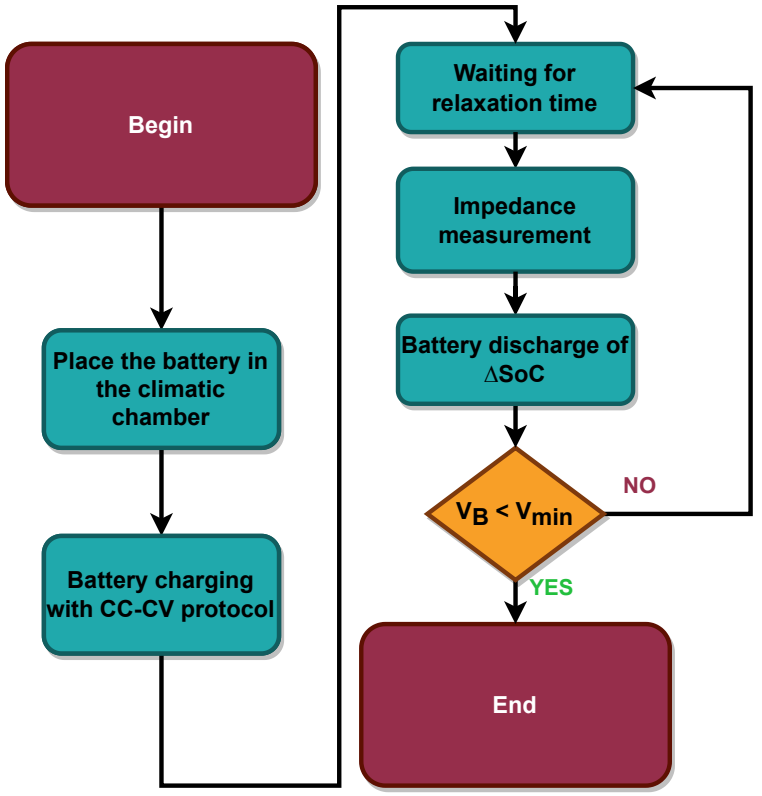


Figure 5.18: Adopted protocol for batteries characterization.

Since there are no lithium ions, the characteristic diffusion times tend to infinity, and the impedance is higher than the impedances of other SoCs.

Finally, Figure 5.20 shows the structure of the entire dataset obtained from our experimental campaign. For each battery tested, 20 SoC levels were considered, and 28 complex impedance measurements were obtained for each SoC level. Each impedance measurement comprises two real numbers (Real and Imaginary parts) to obtain 56 real features corresponding to a single SoC level.

5.3.3 The proposed method: Classifier Selection

As described in Section 5.3.1 the classifier could be defined from the project, but if it is not in this Section it showed how to select the best classifier for the acquired data. In this case to evaluate the data's validity and identify the best performing

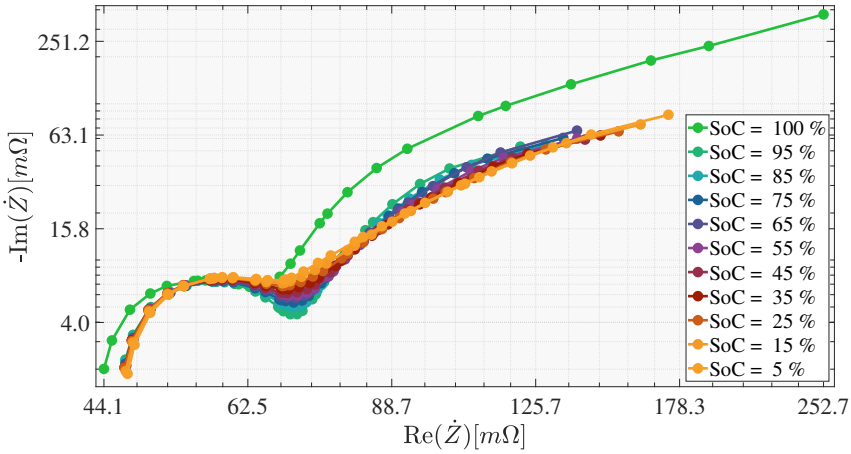


Figure 5.19: Obtained Nyquist plot of a single battery at different State of Charge.

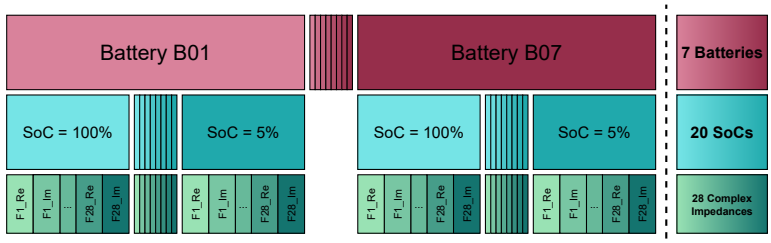


Figure 5.20: Structure of the entire obtained experimental dataset.

Machine Learning (ML) models for SoC estimation for the subsequent Feature Selection (FS) phase, preliminary tests were carried out in which all available features were considered. The performed experiments can be divided into two distinct steps. In the first phase, multiple ML models were analyzed to identify a subset of algorithms most representative of the problem under consideration, both in terms of achieved performance and computational complexity. In the second phase, the resulting models were used as the basis for in–depth analysis to evaluate their validity in real applications. The SoC estimation problem was represented through 10–class classification models, where each of the classes represents a 10 % SoC interval, thus producing estimation intervals $[0-10], (10-20], \ldots, (90-100]$. As already mentioned, all available features were used, with 28 impedances for each SoC (Real and Imaginary part) acquired at different frequencies constituting the 56 features used. In addition, no normalization techniques or more general pre-processing methods

were applied, since standard techniques do not lead to significant improvements in the performance of the various models. In the various experiments, the principal metrics related to multiclass classification problems, as highlighted by Grandini et al. in [165], were used for performance evaluation. Accuracy (Acc), Matthews Correlation Coefficient (MCC), Precision (P), Recall (R), and F1 Score (F1) were used in this work. These are mainly defined for binary classification problems and were used in a multiclass context averaging the unweighted mean per label.

Preliminary analysis

The dataset composed of seven LFP batteries was divided into two sub–datasets; the first one equal to 80 % of the data used to build the models (Training set) and the second one equal to 20 % of the data used to evaluate their performances (Test set). To properly analyze the various algorithms, it was also necessary to evaluate their response to the variation of the respective hyperparameters. Consequently, the portion of the dataset used for training the models was further randomly divided into five folds, applying the k–fold cross–validation method. The analyzed models' related metrics and the best obtained hyperparameters are shown in Table 5.4.

From this analysis, three models were extracted to investigate their performance further. The chosen models are as follows:

- **K–Nearest Neighbors**, mainly characterized by negligible computational complexity for training and not very high performance.
- **Random Forest**, as a good performance/complexity trade-off.
- **Support Vector Machine**, to achieve the best performance.

5.3.4 In–depth analysis

The selected models in the previous phase were used to conduct further tests. In particular, the main purpose is to act in a controlled manner on the separation of data used for training and testing, thus ensuring that portions of measurements obtained from the batteries are not present in both Train and Test data. The experiments were conducted always following the k–fold method, using six folds in which measurements from two batteries were used as Tests and the remainder for Training. We then turned the batteries used in such a way as to make maximum use of the dataset, obtaining six trained models for each type of classification algorithm. The obtained results are shown in Table 5.5. The considered metrics confirm that the best-performing model is based on the Support Vector Machine (SVM) with a mean accuracy of 0.83 and a standard deviation of 0.04. Figure 5.21 shows the resulting confusion matrix.

Table 5.4: Average values of the considered evaluation metrics and main best obtained hyperparameters for the ML models.

Model	**Acc**	**MCC**	**P**	**R**	**F1**	**Hyper–parameter**	**Value**
AdaBoost	0.34	0.31	0.32	0.36	0.27	Base estimator	Decision Tree
						Number of estimators	50
						Learning rate	0.1
Logistic Regression	0.45	0.39	0.43	0.45	0.43	Regularization param	10000
						Norm of the penalty	l2
Multi–Layer Perceptron	0.55	0.52	0.52	0.57	0.51	Activation function	ReLu
						Hidden layer sizes	100, 200, 100
						Learning rate	Constant
						Solver	Adam
K–Nearest Neighbor	0.57	0.53	0.59	0.57	0.57	Number of neighbors	3
						Minkowski metrics	Euclidean
						Weight function	Distance
Gaussian Naive Bayes	0.68	0.65	0.72	0.68	0.67	Variance smoothing	1e-9
Bagging	0.75	0.72	0.77	0.74	0.74	Base estimator	Decision Tree
						Number of estimators	50
Random Forest	0.75	0.73	0.78	0.74	0.74	Criterion	log_loss
						Number of estimators	100
Support Vector Machine	0.89	0.88	0.92	0.89	0.89	Regularization param	10000
						Kernel type	RBF
						Kernel coefficient	1
						Decision function	one-vs-one

The final result of these preliminary classification tests allows the SVM to be identified as the best ML model among those considered. Hereafter, within the proposed FS methodologies involving the use of a classification model, the SVM will be the adopted model.

5.3.5 The proposed method: feature selection

Feature Selection (FS) aims to decrease the number of used features by removing nonessential or noisy ones. Consider a set of data samples to be classified $\mathcal{Z}$, represented by a set Y of N features. The FS problem involves finding the best subset $X^* \in Y$ of M ($M < N$) features using an objective function J. The function $J(X)$ evaluates the discriminatory power of the feature subspace represented by the subset X, when the samples in $\mathcal{Z}$ are projected onto it. Given an evaluation function, the optimal subset can be obtained by exhaustively evaluating all pos-

Table 5.5: Average values and standard deviation of the considered evaluation metrics examined, where Train and Test datasets are divided by batteries.

	k-NN	Random Forest	Support Vector Machine
Acc	0.62 ± 0.12	0.79 ± 0.05	0.83 ± 0.04
MCC	0.58 ± 0.13	0.77 ± 0.05	0.82 ± 0.04
P	0.65 ± 0.11	0.81 ± 0.05	0.85 ± 0.04
R	0.63 ± 0.12	0.79 ± 0.05	0.84 ± 0.04
F1	0.62 ± 0.13	0.79 ± 0.05	0.84 ± 0.04

sible solutions. However, this exhaustive search is often impracticable due to the exponential increase in the number of solutions (2^N, where N is the number of available features). Therefore, many search techniques have been developed for FS to address this challenge, including complete search, greedy search, and heuristic search. However, since these algorithms do not effectively consider complex interactions among features, they often encounter problems such as being trapped in local optima or incurring high computational costs. Evolutionary Computation (EC) techniques have been widely used in this context as they are well known for their global search ability. Furthermore, EC techniques do not need domain knowledge and do not make any assumptions about the evaluation function, such for example linearity or differentiability.

In this work, we addressed the problem of identifying the optimal set of frequencies for impedance measurement by EIS for battery SoC estimation. To achieve this result, we used two optimization algorithms as search strategies; the former is based on the Genetic Algorithm (GA), and the latter is based on Particle Swarm Optimization (PSO). Furthermore, for both considered search algorithms, two different fitness functions are implemented: the first based on a supervised learning model and the second based on a correlation analysis. The proposed approaches regarding search algorithms and implemented fitness functions will be described below.

Search algorithms

Figure 5.22 shows the schematic structure of both used search algorithms. Both the GA and the PSO are discussed below.

Genetic Algorithm

A Genetic Algorithm (GA) simulates the natural process of evolution by creating a population of potential solutions, represented as binary strings, and modifying

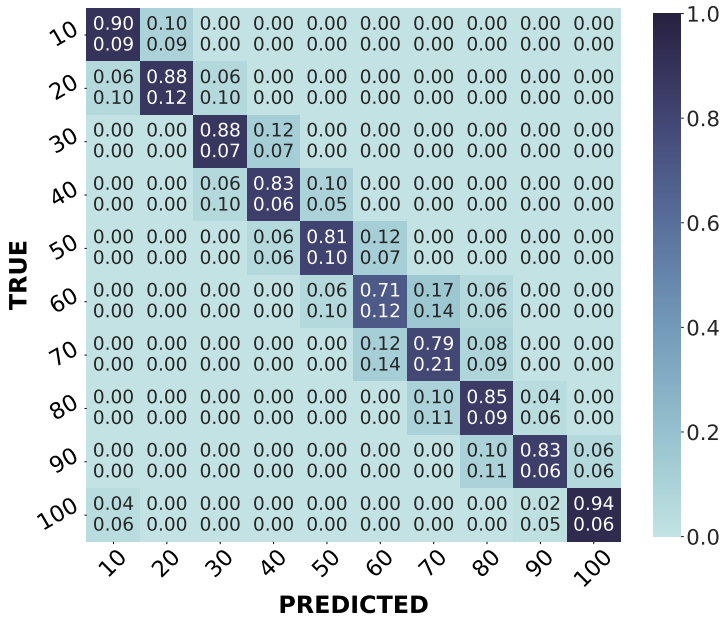


Figure 5.21: Obtained confusion matrix of the Support Vector Machine model, with mean value (top) and standard deviation (bottom) for each class.

them using crossover and mutation operators to explore the search space [166]. The implemented method utilized a generational GA, where individuals were represented by binary vectors that encoded a subset of features. Each individual is a binary vector with a constant length equal to N, where the feature inclusion or exclusion is coded respectively with 1 or 0. The algorithm flow chart is shown on the left side of Figure 5.22. To start the algorithm, a population of individuals P is created where each individual's values are randomly assigned as 1 or 0. Next, a fitness function is used to evaluate each individual's fitness, considering the set frequencies encoded by that individual.

Following this evaluation phase, a new population is generated by first implementing an elitist strategy that involves copying the best individuals from the current population to preserve the best elements in the current population. The remaining $(P-e)/2$ couples of individuals, referred to as parents, are selected using

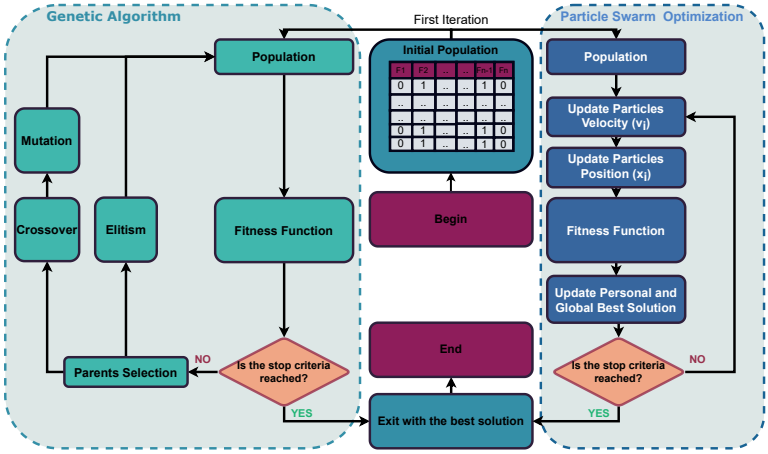


Figure 5.22: Genetic Algorithm (left) and Particle Swarm Optimization (right) flow charts.

a tournament selection method of size t, where an individual from the population is chosen by selecting the best from t randomly selected individuals. The uniform crossover operator is applied to each selected couple, followed by the mutation operator, which is applied with probabilities equal to pc and pm, respectively. The resulting offspring are then evaluated and added to the new population. This process is repeated for Ng generations. All the setting parameter values of pc, pm, Ng are in Table 5.6.

Table 5.6: Genetic Algorithm parameters.

Parameter	**Symbol**	**Value**
population size	P	100
selection type	-	tournament
tournament size	t	2
elitism	e	2
crossover probability	p_c	0.6
mutation type	-	random
mutation number of genes	m_{size}	1 bit
generation number	N_g	250

Particle Swarm Optimization

The Particle Swarm Optimization (PSO) algorithm [167] is a global optimization algorithm particularly well suited for solving problems in which the optimal solution is a point in a multidimensional space and can be used for both real-valued and discrete–data optimization. These swarm–based algorithms are a family of nature–inspired, population–based algorithms capable of producing robust solutions to various complex problems. A swarm consists of a population of simple, homogeneous agents that perform elementary tasks by interacting with each other and the environment.

From a mathematical point of view, as stated in (5.4), assuming that we want to solve a maximization problem and denote by f the objective function, the problem is defined as finding the $\boldsymbol{x}_{best}$ coordinate point, where the domain $\mathbb{R}^D$ is called the search space and the value of $f(\boldsymbol{x})$ is commonly referred as *fitness*. In PSO, each candidate solution ($\boldsymbol{x}_i$) is called a particle and represents a point in the search space.

$$\begin{aligned} &f\colon \mathbb{R}^D \longrightarrow \mathbb{R} \\ &\boldsymbol{x}_{best} | f(\boldsymbol{x}_{best}) \geq f(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \mathbb{R}^D \end{aligned} \tag{5.4}$$

The set of these elements constitutes the swarm, which in search of the solution to the problem, iteratively updates its position based on the equation of motion shown in (5.5), where v_i is a D–dimensional vector representing the components of the velocity of the $i-th$ particle. The time instants t and $t+1$ represent two successive iterations of the algorithm [168]. From the algorithmic point of view, the iterative execution of the process is represented in Figure 5.22.

$$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1) \tag{5.5}$$

For solving the problem under consideration, the optimization algorithm was applied to discrete data [169], where each particle is represented by a binary vector that describes a subset of features. Table 5.7 shows the parameters used in the following experiments.

Fitness functions

As discussed above, two fitness functions were implemented for each considered serch algorithm. Fitness functions allow selecting which features (and thus which frequencies) to use for SoC estimation. The idea behind the definition of both fitnesses is to consider two aspects: maximizing the information content of features and minimizing the overall measurement time. The measurement time is calculated as the sum of all selected stimulus periods. Since the measured impedances are

Table 5.7: Particle Swarm Optimization parameters.

Parameter	Symbol	Value
optimizer	-	Binary PSO
cognitive parameter	c1	2
social parameter	c2	2
inertia parameter	w	0.8
iterations	i	250
number of particles	n	100

complex numbers, that is, composed of a real and an imaginary part, two features are generated for each stimulus frequency. The instrument used for impedance measurement cannot measure only one of the two impedance components but outputs them simultaneously every time. For this reason, the measurement time is computed by counting a measurement period in case one or both the features at the same frequency are considered.

Classification–based fitness

We developed and tested a fitness function based on a SVM classifier because, as explained in Section 5.3.3, it is the most performative algorithm for classifying the State of Charge over the acquired data. The structure of the fitness function is shown in (5.6). The SVM is a supervised classifier that divides a projected space of features with hyperplanes. The optimal hyperplane is selected based on its ability to divide different classes on the test dataset. In the case of FS, the feature space changes at each iteration; let Ch be one of the Chromosome containing M features, our algorithm uses the M *dimensional* space on the training sub–dataset for selecting the best hyperplane. After the test is performed on the test sub–dataset with the same M *dimensional* space, the resulting SoC estimation accuracy is evaluated as shown in (5.7), where the A parameter is the fraction between the correct predictions CP over the total predictions TP. The focus of the FS in this application is reducing the measurement time, saving the SoC estimation accuracy, or at least finding a good compromise. For this purpose, the measurement time is considered with the B parameter reported in (5.8). It has an inverse relationship with the duration of the measurement process, the term T_{meas} is the measurement time, and it is computed as the summation of the durations of the selected features (frequencies). On the other hand, the maximum measurement time T_{max} is determined by summing the durations of all features. In our case, T_{max} is around 4 minutes. Using measurement periods instead of the number of features enables the algorithm to minimize the number of features and prioritize higher

frequencies over lower frequencies because the impact of lower frequencies on the measurement time is higher than the impact of higher frequencies. Furthermore, the terms A and B lie within the range of [0, 1], while the α weight coefficient is a weight between the SoC estimation accuracy and measurement time contributions.

$$S = \alpha \cdot A + (1 - \alpha) \cdot B \tag{5.6}$$

$$A = \frac{CP}{TP} \tag{5.7}$$

$$B = 1 - \frac{T_{meas}}{T_{max}} \tag{5.8}$$

Correlation–based fitness

The fitness function based on Pearson's Correlation Coefficient (PCC), defined in (5.9) for a paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, aims to allow the identification of a subset of features such that the redundancy of information in the dataset is minimized. For each solution the algorithms return, the correlation matrix between the selected features is calculated, from which an index associated with the individual is obtained through the arithmetic mean of the unique elements in the matrix. Measurement time is also taken into account in the resulting optimization problem. The obtained function is structured as a maximization problem (5.10) and comprises two elements weighted using an α weight coefficient. The first element C shown in (5.11) is the term that represents the correlation index of the solution under consideration, while the second element B is inversely proportional to the measurement time as discussed above (5.8).

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.9}$$

$$S = \alpha \cdot C + (1 - \alpha) \cdot B \tag{5.10}$$

$$C = 1 - \frac{1}{n}\sum_{i=1}^{n} |r_{xy}(i)| \tag{5.11}$$

5.3.6 Performance evaluation

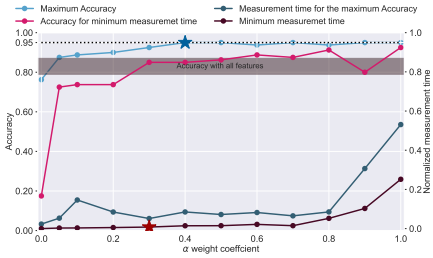
In this Section, we will discuss the results of the FS by applying the search algorithms and the fitness functions just discussed to the experimental dataset described in Section 5.3.2. Specifically, we implemented the two fitness functions based on

SVM and PCC on the GA and PSO search algorithms, obtaining four possible combination schemes: GA+SVM, PSO+SVM, GA+PCC, and PSO+PCC. The experimental parameters used for the GA and PSO in all experiments described in the following were set after some preliminary trials (see Table 5.6 and Table 5.7). Furthermore, to evaluate the performance of the tested systems, we performed 50 independent runs, each with a different initial population, randomly generated. At the end of each run, the best individual found was selected as the output solution for that run. The results reported in the following were computed by taking the best result in terms of Accuracy over the 50 runs performed. The hyperparameters used for SVM are the same as those shown in Table 5.4. For each of the runs, the model construction follows the structure described in section 5.3.4, randomly selecting one of the available folds. Finally, the obtained performances were evaluated by considering several α weight coefficients within the implemented fitness functions (5.6), (5.10).

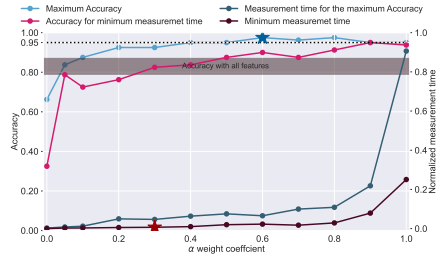
We imaged two different application scenarios; in the first one, the goal is to select all the solutions with a classification Accuracy better than a target, whatever the time. The second scenario, instead, has multiple goals: select the solution with the minimum measurement time but without classification performance degradation for the case with all the features. With this purpose, Figure 5.23 shows the trends of the maximum Accuracy and measurement time obtained for each value of α, considering all combinations of fitness functions and search algorithms.

5.3.7 Case study A: Accuracy target

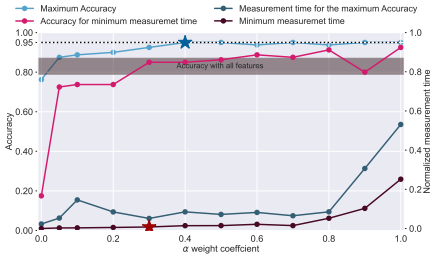
In this applied case study, we set a target Accuracy of 0.95 without considering the measurement time, the threshold should be defined into the requirements in Section 5.3.1. Figure 5.23 reports the maximum Accuracy obtained through the 50 runs with its related measurement time as a function of the α weight coefficient. The dot marked with a blue star indicates the solution with maximum accuracy. Lastly, the band represents the SoC estimation Accuracy obtained considering all features with the SVM classifier, along with the relative uncertainty, as reported in Table 5.5. For Accuracy, the higher is the value and the better is the solution, instead for measurement time is the opposite, lower is the value, and better is the solution. Regardless of the fitness function and search algorithms combination, Accuracy increases as alpha increases. This phenomenon occurs because, with low alpha values, search algorithms tend to generate solutions that minimize measurement time rather than maximize classifier accuracy. Conversely, high alpha values lead the search algorithms to place greater emphasis on Accuracy at the expense of minimizing measurement time. To assess the comparative solution searching performances of GA and PSO, not notable disparities between the obtained solutions'



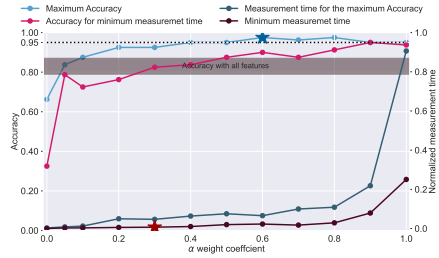
(a) Accuracy and measurement time as a function of α weight coefficient for GA+SVM combination.



(b) Accuracy and measurement time as a function of α weight coefficient for PSO+SVM combination.



(c) Accuracy and measurement time as a function of α weight coefficient for GA+PCC combination.



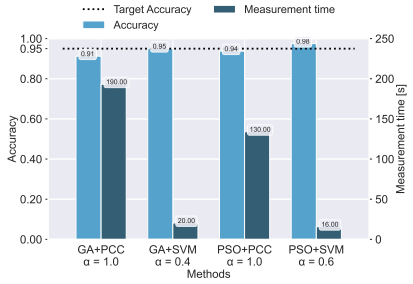
(d) Accuracy and measurement time as a function of α weight coefficient for PSO+PCC combination.

Figure 5.23: Analysis of Accuracy and measurement time as a function of alpha weighting coefficient for different configurations of search algorithms and fitness functions.

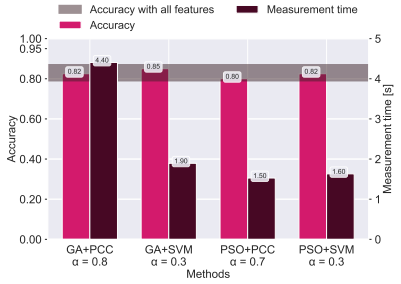
Accuracy can be observed in Figures figs. 5.23a and 5.23b, this suggests that their performances are comparable. This observation holds true when comparing the solutions' Accuracy depicted in Figures figs. 5.23c and 5.23d.

Regarding the fitness functions, it is clear that the one based on the SVM classifier achieves on average superior performance than the PCC algorithm in Accuracy. The target Accuracy is reached only when the fitness function includes the SVM classifier. An interesting result is that the feature selection helps to improve SoC evaluation performances. There are several cases where the accuracy is better than the one reached with all features. This fact could be explained by considering the presence of ambiguous features that trick the classification algorithm. Thus, even if there is no necessity of lowering the measurement time this method could exclude the bad feature and increase the overall classification Accuracy.

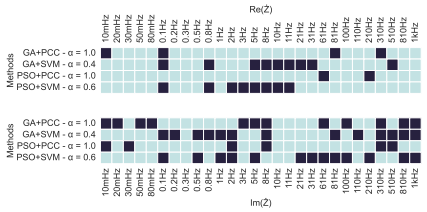
Figure 5.24a expand the star-marked points from Figures figs. 5.23a to 5.23d;



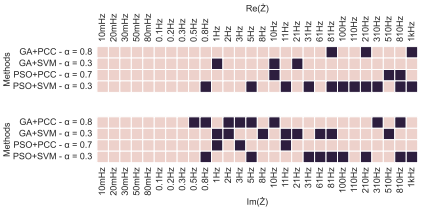
(a) Accuracy and time performances of the solutions with the higher SoC classification Accuracy.



(b) Accuracy and time performances of the solutions with two constraints on time and Accuracy.



(c) Distribution of selected features through the different combinations of searching algorithms and fitness functions for the scenario A.



(d) Distribution of selected features through the different combinations of searching algorithms and fitness functions for scenario B.

Figure 5.24: Performance analysis and distribution of selected features using different combinations of search algorithms and fitness functions for SoC estimation in the two scenarios A and B, considering the constraints imposed.

the left axis reports the Accuracy while the right axis the measurement time for all the combinations of searching algorithms and fitness functions. From this, it's clear that the better combination for the acquired dataset is the PSO+SVM for both the accuracy that reaches 0.98 and the measurement time of 16 seconds. In this case, even if the optimization in time was not considered, the time saved is over 93%, considering a total measuring time of 240 seconds.

Figure 5.24c shows the selected features for the real and imaginary parts of the impedance for all the stimulus frequencies. Comparing the real and imaginary parts is clear that for all the combinations the imaginary part looks more important for the classifier than the real part, because it is selected more times. Furthermore, the PCC tends to select fewer features than the SVM but more on the lower frequencies with a higher impact on the measurement time.

5.3.8 Case study B: Accuracy target and minimum measurement time

In this case study, we have established two constraints: one about the evaluation SoC Accuracy and the other concerning the measurement time. Specifically, our objective is to identify solutions that do not affect Accuracy performance compared to the classification Accuracy achieved using all features outlined in Section 5.3.1, minimizing the measurement time. Figures figs. 5.23a to 5.23d present the trends of the minimum measurement time for various values of the α weight coefficient, along with their associated Accuracies. The red star denotes the solution that exhibits the minimum measurement time while maintaining the Accuracy performances of the tests with all features. The measurement time increases as α approaches 1 due to the multiplication of the variable B in (5.6) and (5.10) with a coefficient $(1-\alpha)$ close to 0. It is worth noting that the Accuracy related to SVM classification displays a significant increase as α transitions from 0 to 0.05, followed by a gradual incline beyond $\alpha = 0.05$.

Figure 5.24b illustrates the solutions for minimum measurement time that also satisfy the Accuracy requirement. The lower measurement time is achieved using the PSO search algorithm with the fitness function based on the PCC. For all the selected frequencies, the system requires only 1.5 seconds, which accounts for less than 1 % of the measurement time when utilizing all features. A more detailed depiction of the selected feature for the second scenario is presented in Figure 5.24d. Due to the time constraint, all combinations of search algorithms and fitness functions tend to exclude lower frequencies, as they substantially impact the measurement time.

5.3.9 Conclusions

In this work, we propose a measurement method to minimize measurement time in SoC estimation given the target Accuracy needed for the classification. It is based on selecting the most useful feature for a data–driven State of Charge (SoC) estimation. The aim is given to the Battery Management System (BMS) designers a workflow for optimizing the SoC estimation through a data–driven approach, instead simply presenting a result on a specific dataset. We think that is a winning approach because the Feature Selection (FS) operation strongly depends on the acquired data, the chosen classifier, and other parameters.

The proposed FS is based on multidimensional search algorithms, specifically the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) for selecting the most relevant features in the Electrochemical Impedance Spectroscopy (EIS) signals. Two different fitness functions are considered, one based on a supervised

learning model, specifically Support Vector Machine (SVM), and the other based on feature correlation through Pearson's Correlation Coefficient (PCC). As a data source for the FS algorithms, we acquired an experimental EIS-based dataset consisting of seven Lithium Iron Phosphate batteries.

Table 5.8: Comparison of the analyzed scenarios based on some application examples. The first scenario (A) emphasizes accuracy, neglecting measurement time, while the second (B) achieves a use–case–dependent trade–off.

Scenarios	Target	Accuracy	Measurement Time [s]
A	Accuracy	0.98	16
B	Accuracy and Measurement Time	0.80	1.5

To create a reference by selecting all available features, the obtained classification Accuracy was 0.82 with a corresponding measurement time of 240 s. To show the potential of the proposed method, we explore two distinct application scenarios. In Table 5.8 the obtained results are reported for both scenarios. In scenario A, the primary requirement is achieving a SoC classification Accuracy of 0.95, without taking into account the measurement time. The optimal solution achieves a classification Accuracy of 0.98, with a measurement time of 16 seconds, which is lower than the measurement time for all the stimulus frequencies. On the other hand, scenario B introduces two constraints: one on the classification Accuracy and another on the measurement time. In this case, the best solution identified by the search algorithms attains an Accuracy of 0.82, with a remarkable measurement time of 1.5 seconds. This impressive time–saving of up to 99% highlights the efficiency gained by utilizing the selected features instead the all measured features. Furthermore, it is crucial to acknowledge the impact of misleading features on the classification algorithms. Eliminating those features improves the SoC estimation Accuracy over 10% and demonstrates the importance of FS in optimizing the SoC evaluation performance even if saving time is not a priority.

These findings highlight the significance of careful FS for efficient and reliable of SoC estimation performance. The proposed method aims to give the BMS designers a new instrument that includes a multi–optimization fitness function that allows them to choose the weight of each aspect and consequentially focus the feature selection more on reducing the measuring time, increasing the classification Accuracy, or optimizing the number of classes. A Further step could include the adoption of a regression approach inside the proposed method for feature selection

to obtain a continuous SoC estimation.

Summary and conclusions

This thesis explores the deployment of Artificial Intelligence (AI) on edge devices for environmental monitoring and battery management applications, focusing on the use of the SENSIPLUS platform developed by Sensichips s.r.l. Unlike traditional cloud-based AI, edge AI processes data locally on devices such as IoT sensors, enabling lower latency, reduced bandwidth, enhanced privacy, and improved reliability. These features are critical for real-time applications such as autonomous vehicles, industrial automation, and smart city systems. However, implementing AI on edge devices is challenging due to limitations in processing power, memory, and battery life, requiring optimization methods such as model compression and quantization. The research specifically addresses environmental monitoring and intelligent battery management. For environmental monitoring, edge AI enables rapid sensor data analysis to detect water pollutants, helping to respond to environmental changes in real–time. For smart batteries, AI models optimize battery performance in electric vehicles and renewable energy storage, improving energy efficiency and extending battery life. Through applications of the SENSIPLUS sensor platform, this research demonstrates the potential of edge AI to create scalable and responsive solutions across multiple industries.

In chapter 2, the SENSIPLUS platform, developed by Sensichips s.r.l., was presented as the sensor suite for this research. It supports Electrochemical Impedance Spectroscopy (EIS) and is adaptable to different applications. The platform's architecture allows for optimized deployment on edge devices and ensures compatibility with Arduino-compliant architectures. This chapter highlights the versatility of SENSIPLUS, especially for edge AI tasks, by facilitating local processing, which is essential for Internet of Things and real-time monitoring.

An open-source toolchain for the SENSIPLUS platform is presented in chapter 3, focusing on firmware generation and deployment of Machine Learning (ML) models on microcontrollers. The toolchain supports end-to-end model deployment, which is demonstrated through a case study on contaminant detection in wastewater. Experiments with K-fold cross-validation yielded the same accuracy as the PC

counterpart, demonstrating its high detection capability. This toolchain increases the deployment efficiency of edge ML applications and is available for further customization.

For the in-water monitoring activities described in chapter 4, the thesis developed an AI-driven wastewater pollutant detection system using the SENSIPLUS platform. This chapter includes the design of an IoT-based platform that achieved 93% accuracy in pollutant detection. The system's two-phase detection approach, which includes anomaly detection followed by multi-class classification, significantly reduced false positives. By combining machine learning techniques, this chapter demonstrates the capability of low-cost sensors in challenging environments, highlighting real-time response potential and reliability. A Natural Language Processing (NLP) approach to detecting contaminants in water is also presented in the same chapter. Although not readily applicable to devices with limited resources, these experiments demonstrate how such systems can significantly improve model performance.

For the battery management activities presented in chapter 5, this research focuses on a novel State of Charge (SoC) estimation method using feature selection to optimize both accuracy and inference time. The proposed method achieves high efficiency in measurement time reduction and improvement in SoC accuracy. Through this multi-objective optimization, the system can prioritize speed, accuracy, or other factors as needed. This work opens the way for intelligent power management in Internet of Things applications. In addition, this chapter has evaluated SoC estimation systems that are highly compatible with resource-constrained devices, showing how systems such as the Battery Cell Management Unit (CMU) can be used to correctly estimate state of charge through pre-characterization and LUTs. Statistical correlations between SoC and the peaks and amplitudes of the PDFs obtained with the Kernel Density Estimation (KDE) method from the EIS measurements were also identified. Future research will focus on several key aspects. For the contaminant detection system, the goal is to test it in real-world scenarios and solve challenges related to solid waste and contaminant injection time management. It will also be important to explore synthetic data generation and Transfer Learning (TF) techniques to improve the training of Deep Learning (DL) models. In the area of battery state of charge estimation, future research will focus on integrating temperature effects and using EIS information to generate LUTs for OCV and ESR. Further tests will be conducted on different batteries and with different current profiles. Finally, to improve edge AI in general, it will be critical to focus on improving sensor quality, optimizing power management, and improving model update efficiency. The results presented in this thesis demonstrate that edge AI when deployed with optimized hardware and intelligent software, provides

a viable solution for real-time monitoring and responsive decision–making in environments where latency, energy efficiency, and device autonomy are critical. This research highlights the transformative potential of edge AI in environmental and energy applications by achieving the low latency and high reliability essential for real-time autonomous systems. While promising, further improvements in sensor quality, power management, and model update efficiency are needed to fully realize the potential of edge AI. The achievement of this study suggests that further advances in edge AI could lead to innovative solutions that improve environmental sustainability and energy optimization across a wide range of industries.

Bibliography

[1] *CN0510 User Guide [Analog Devices Wiki] — wiki.analog.com*. https://wiki.analog.com/resources/eval/user-guides/circuits-from-the-lab/cn0510. [Accessed 05-8-2024].

[2] Parul Datta and Bhisham Sharma. "A survey on IoT architectures, protocols, security and smart city based applications". In: *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. 2017, pp. 1–5. DOI: 10.1109/ICCCNT.2017.8203943.

[3] Nunzio Cennamo et al. "Water monitoring in smart cities exploiting plastic optical fibers and molecularly imprinted polymers. The case of PFBS detection". In: *2019 IEEE International Symposium on Measurements & Networking (M&N)*. 2019, pp. 1–6. DOI: 10.1109/IWMN.2019.8805049.

[4] Gerhard P. Hancke, Bruno De Carvalho e Silva, and Gerhard P. Hancke Jr. "The Role of Advanced Sensing in Smart Cities". In: *Sensors* 13.1 (2013), pp. 393–425. ISSN: 1424-8220. DOI: 10.3390/s130100393. URL: https://www.mdpi.com/1424-8220/13/1/393.

[5] Marco Carminati et al. "Miniaturized Pervasive Sensors for Indoor Health Monitoring in Smart Cities". In: *Smart Cities* 4.1 (2021), pp. 146–155. ISSN: 2624-6511. DOI: 10.3390/smartcities4010008. URL: https://www.mdpi.com/2624-6511/4/1/8.

[6] Wei Yu et al. "A Survey on the Edge Computing for the Internet of Things". In: *IEEE Access* 6 (2018), pp. 6900–6919. DOI: 10.1109/ACCESS.2017.2778504.

[7] Zhi Zhou et al. "Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing". In: *Proceedings of the IEEE* 107.8 (2019), pp. 1738–1762. DOI: 10.1109/JPROC.2019.2918951.

[8] Kuljeet Kaur et al. "Edge Computing in the Industrial Internet of Things Environment: Software-Defined-Networks-Based Edge-Cloud Interplay". In: *IEEE Communications Magazine* 56.2 (2018), pp. 44–51. DOI: 10.1109/MCOM.2018.1700622.

[9] Michele Vitelli et al. “SENSIPLUS-LM: A Low-Cost EIS-Enabled Microchip Enhanced with an Open-Source Tiny Machine Learning Toolchain”. In: *Computers* 12.2 (2023). ISSN: 2073-431X. DOI: 10.3390/computers12020023. URL: https://www.mdpi.com/2073-431X/12/2/23.

[10] Mario Molinara et al. “An End to End Indoor Air Monitoring System Based on Machine Learning and SENSIPLUS Platform”. In: *IEEE Access* 8 (2020), pp. 72204–72215. DOI: 10.1109/ACCESS.2020.2987756.

[11] Giovanni Betta et al. “Contaminants detection and classification through a customized IoT-based platform: A case study”. In: *IEEE Instrumentation & Measurement Magazine* 22.6 (2019), pp. 35–44. DOI: 10.1109/MIM.2019.8917902.

[12] Luca Gerevini et al. “An end-to-end real-time pollutants spilling recognition in wastewater based on the IoT-ready SENSIPLUS platform”. In: *Journal of King Saud University - Computer and Information Sciences* (2023). ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2022.12.018. URL: https://www.sciencedirect.com/science/article/pii/S1319157822004360.

[13] Shuiguang Deng et al. “Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence”. In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 7457–7469. DOI: 10.1109/JIOT.2020.2984887.

[14] Partha Pratim Ray. “A review on TinyML: State-of-the-art and prospects”. In: *Journal of King Saud University - Computer and Information Sciences* 34.4 (2022), pp. 1595–1623. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.11.019. URL: https://www.sciencedirect.com/science/article/pii/S1319157821003335.

[15] Dr. Lachit Dutta and Swapna Bharali. “TinyML Meets IoT: A Comprehensive Survey”. In: *Internet of Things* 16 (2021), p. 100461. ISSN: 2542-6605. DOI: https://doi.org/10.1016/j.iot.2021.100461. URL: https://www.sciencedirect.com/science/article/pii/S2542660521001025.

[16] Swapnil Sayan Saha, Sandeep Singh Sandha, and Mani Srivastava. “Machine Learning for Microcontroller-Class Hardware: A Review”. In: *IEEE Sensors Journal* 22.22 (2022), pp. 21362–21390. DOI: 10.1109/JSEN.2022.3210773.

[17] *TensorFlow Lite for Microcontrollers — tensorflow.org*. https://www.tensorflow.org/lite/microcontrollers. [Accessed 15-Sep-2022].

[18] S. Soro. “TinyML for Ubiquitous Edge AI”. In: *ArXiv* abs/2102.01255 (2021).

[19] Hadis Karimipour and Franaz Derakhshan. "Artificial Intelligence for Threat Detection and Analysis in Industrial IoT: Applications and Challenges". In: *AI-Enabled Threat Detection and Security Analysis for Industrial IoT*. Ed. by Hadis Karimipour and Farnaz Derakhshan. Cham: Springer International Publishing, 2021, pp. 1–6. ISBN: 978-3-030-76613-9. DOI: 10.1007/978-3-030-76613-9_1. URL: https://doi.org/10.1007/978-3-030-76613-9_1.

[20] Muhammad Bilal et al. "Smart Cities Data: Framework, Applications, and Challenges". In: *Handbook of Smart Cities*. Ed. by Juan Carlos Augusto. Cham: Springer International Publishing, 2020, pp. 1–29. ISBN: 978-3-030-15145-4. DOI: 10.1007/978-3-030-15145-4_6-1. URL: https://doi.org/10.1007/978-3-030-15145-4_6-1.

[21] *uTensor*. http://utensor.ai. [Accessed 15-Sep-2022].

[22] *NanoEdge AI Studio*. https://cartesiam-neai-docs.readthedocs-hosted.com/. [Accessed 15-Sep-2022].

[23] *Edge Impulse*. https://www.edgeimpulse.com/. [Accessed 15-Sep-2022].

[24] *STM32Cube.AI*. https://www.st.com/content/st_com/en/ecosystems/stm32-ann.html. [Accessed 15-Sep-2022].

[25] Xiaying Wang et al. "FANN-on-MCU: An Open-Source Toolkit for Energy-Efficient Neural Network Inference at the Edge of the Internet of Things". In: *IEEE Internet of Things Journal* 7 (2020), pp. 4403–4417.

[26] Benjamin Sliwa, Nico Piatkowski, and Christian Wietfeld. "LIMITS: Lightweight Machine Learning for IoT Systems with Resource Limitations". In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. 2020, pp. 1–7. DOI: 10.1109/ICC40277.2020.9149180.

[27] *SPC5-STUDIO - STMicroelectronics — st.com*. https://www.st.com/en/development-tools/spc5-studio.html. [Accessed 15-Sep-2022].

[28] *BME688 Software — bosch-sensortec.com*. https://www.bosch-sensortec.com/software-tools/software/bme688-software. [Accessed 15-Sep-2022].

[29] *SENSIPLUSToolchain · GitLab — gitlab.com*. https://gitlab.com/t8664. [Accessed 15-Sep-2022].

[30] E. Frank et al. "Weka: A machine learning workbench for data mining." In: *Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Ed. by O. Maimon and L. Rokach. Berlin: Springer, 2005, pp. 1305–1314. URL: http://researchcommons.waikato.ac.nz/handle/10289/1497.

[31] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. 2nd ed. Springer, 2009. URL: http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

[32] Antonello Rosato, Rosa Altilio, and Massimo Panella. "On-line Learning of RVFL Neural Networks on Finite Precision Hardware". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351399.

[33] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[34] *Espressif company*. https://www.espressif.com/en/company/about-espressif. [Accessed 15-Oct-2024].

[35] Bharath Sudharsan et al. "TinyML Benchmark: Executing Fully Connected Neural Networks on Commodity Microcontrollers". In: *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*. 2021, pp. 883–884. DOI: 10.1109/WF-IoT51360.2021.9595024.

[36] *D32 Pro; WEMOS documentation — docs.wemos.cc*. https://docs.wemos.cc/en/latest/d32/d32_pro.html. [Accessed 15-Sep-2022].

[37] A. Bria et al. "An IoT-ready solution for automated recognition of water contaminants". In: *Pattern Recognition Letters* 135 (2020). ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2020.04.019. URL: https://www.sciencedirect.com/science/article/pii/S0167865520301410.

[38] *Public link for downloading the acquired dataset*. https://aida.unicas.it/data/JKSU_2022.zip. [Online: accessed 11-November-2022]. 2022.

[39] Youdi Gong et al. "A survey on dataset quality in machine learning". In: *Information and Software Technology* 162 (2023), p. 107268. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2023.107268. URL: https://www.sciencedirect.com/science/article/pii/S0950584923001222.

[40] Marco Ferdinandi et al. "A Novel Smart System for Contaminants Detection and Recognition in Water". In: *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2019, pp. 186–191. DOI: 10.1109/SMARTCOMP.2019.00051.

[41] C. Bourelly et al. "A Preliminary Solution for Anomaly Detection in Water Quality Monitoring". In: *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2020, pp. 410–415. DOI: 10.1109/SMARTCOMP50058.2020.00086.

[42] Alessandro Bria et al. "An IoT-ready solution for automated recognition of water contaminants". In: *Pattern Recognition Letters* 135 (2020), pp. 188–195.

[43] Mario Molinara et al. "An End to End Indoor Air Monitoring System Based on Machine Learning and SENSIPLUS Platform". In: *IEEE Access* 8 (2020), pp. 72204–72215. DOI: 10.1109/ACCESS.2020.2987756.

[44] J.S. Lim. *Mobile Sensor Network to Monitor Wastewater Collection Pipelines*. https://escholarship.org/uc/item/0d9813bn. [Online: accessed 11-November-2022]. 2012.

[45] Mathieu Lepot, Konstantinos F. Makris, and François H.L.R. Clemens. "Detection and quantification of lateral, illicit connections and infiltration in sewers with Infra-Red camera: Conclusions after a wide experimental plan". In: *Water Research* 122 (2017), pp. 678–691. ISSN: 0043-1354. DOI: https://doi.org/10.1016/j.watres.2017.06.030. URL: https://www.sciencedirect.com/science/article/pii/S0043135417305067.

[46] Hyon Wook Ji et al. "Measurement of Wastewater Discharge in Sewer Pipes Using Image Analysis". In: *Water* 12.6 (2020). ISSN: 2073-4441. DOI: 10.3390/w12061771. URL: https://www.mdpi.com/2073-4441/12/6/1771.

[47] Ivan Pisa et al. "ANN-Based Soft Sensor to Predict Effluent Violations in Wastewater Treatment Plants". In: *Sensors* 19.6 (2019). ISSN: 1424-8220. DOI: 10.3390/s19061280. URL: https://www.mdpi.com/1424-8220/19/6/1280.

[48] Adam Drenoyanis et al. "Implementation of an IoT Based Radar Sensor Network for Wastewater Management". In: *Sensors* 19.2 (2019). ISSN: 1424-8220. DOI: 10.3390/s19020254. URL: https://www.mdpi.com/1424-8220/19/2/254.

[49] Mihai Horia Popescu et al. "Automatic Assignment of ICD-10 Codes to Diagnostic Texts using Transformers Based Techniques". In: *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*. IEEE. 2021, pp. 188–192.

[50] Kevin Roitero et al. "DiLBERT: Cheap Embeddings for Disease Related Medical NLP". In: *IEEE Access* 9 (2021), pp. 159714–159723.

[51] Vincenzo Della Mea, Mihai Horia Popescu, and Kevin Roitero. "Underlying cause of death identification from death certificates using reverse coding to text and a NLP based deep learning approach". In: *Informatics in Medicine Unlocked* 21 (2020), p. 100456.

[52] Mihai Horia Popescu, Kevin Roitero, and Vincenzo Della Mea. "Explainable Classification of Medical Documents Through a Text-to-Text Transformer". In: *HC@AIxIA 2022: 1st AIxIA Workshop on Artificial Intelligence For Healthcare*. 2022.

[53] Hao Xue et al. "Translating Human Mobility Forecasting through Natural Language Generation". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 1224–1233. DOI: 10.1145/3488560.3498387.

[54] Hao Xue, Bhanu Prakash Voutharoj, and Flora D Salim. "Leveraging Language Foundation Models for Human Mobility Forecasting". In: *arXiv preprint arXiv:2209.05479* (2022).

[55] Alexey Dosovitskiy et al. "An image is worth 16x16 words: Transformers for image recognition at scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[56] Haiping Wu et al. “Cvt: Introducing convolutions to vision transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 22–31.

[57] Reza Azad et al. “Transnorm: Transformer provides a strong spatial normalization mechanism for a deep segmentation model”. In: *IEEE Access* 10 (2022), pp. 108205–108215.

[58] Yuqing Wang et al. “End-to-end video instance segmentation with transformers”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8741–8750.

[59] Hongchun Yuan et al. “Transanomaly: Video anomaly detection using video vision transformer”. In: *IEEE Access* 9 (2021), pp. 123977–123986.

[60] Zhengkun Tian et al. “Synchronous transformers for end-to-end speech recognition”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 7884–7888.

[61] Cem Subakan et al. “Attention is all you need in speech separation”. In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 21–25.

[62] Liu Yang et al. “IART: Intent-aware response ranking with transformers in information-seeking conversation systems”. In: *Proceedings of The Web Conference 2020*. 2020, pp. 2592–2598.

[63] Rafael Ferreira et al. “Open-domain conversational search assistants: the Transformer is all you need”. In: *Information Retrieval Journal* 25.2 (2022), pp. 123–148.

[64] Fei Sun et al. “BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer”. In: *Proceedings of the 28th ACM international conference on information and knowledge management*. 2019, pp. 1441–1450.

[65] Yulong Gu et al. “Deep multifaceted transformers for multi-objective ranking in large-scale e-commerce recommender systems”. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 2493–2500.

[66] Emilio Parisotto et al. “Stabilizing transformers for reinforcement learning”. In: *International conference on machine learning*. PMLR. 2020, pp. 7487–7498.

[67] Lili Chen et al. “Decision transformer: Reinforcement learning via sequence modeling”. In: *Advances in neural information processing systems* 34 (2021), pp. 15084–15097.

[68] Vijay Prakash Dwivedi and Xavier Bresson. “A generalization of transformer networks to graphs”. In: *arXiv preprint arXiv:2012.09699* (2020).

[69] Jinwoo Kim et al. "Pure transformers are powerful graph learners". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14582–14595.

[70] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589.

[71] Rianon Zaman et al. "Constraint Guided Neighbor Generation for Protein Structure Prediction". In: *IEEE Access* 10 (2022), pp. 54991–55001.

[72] Jinyang Wang, Xiao Lin, and Hongying Yu. "POAT-Net: Parallel Offset-attention Assisted Transformer for 3D Object Detection for Autonomous Driving". In: *IEEE Access* 9 (2021), pp. 151110–151117.

[73] Xiaobo Chen et al. "Vehicle Trajectory Prediction Based on Intention-Aware Non-Autoregressive Transformer With Multi-Attention Learning for Internet of Vehicles". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–12.

[74] David Martín-Gutiérrez et al. "A deep learning approach for robust detection of bots in twitter using transformers". In: *IEEE Access* 9 (2021), pp. 54591–54601.

[75] Yunus Emre Seyyar, Ali Gökhan Yavuz, and Halil Murat Ünver. "An attack detection framework based on BERT and deep learning". In: *IEEE Access* 10 (2022), pp. 68633–68644.

[76] In: (). URL: `https://aida.unicas.it/icprchallenge2022/`.

[77] G. Betta et al. "Contaminants detection and classification through a customized IoT-based platform: A case study". In: *IEEE Instrumentation Measurement Magazine* 22.6 (Dec. 2019), pp. 35–44. ISSN: 1941-0123.

[78] Andrea Ria et al. "The SENSIPLUS: A Single-Chip Fully Programmable Sensor Interface". In: *Applications in Electronics Pervading Industry, Environment and Society*. Ed. by Sergio Saponara and Alessandro De Gloria. Cham: Springer International Publishing, 2022, pp. 256–261. ISBN: 978-3-030-95498-7.

[79] Ingmar Nopens, Carlo Capalozza, and Peter A Vanrolleghem. "Stability analysis of a synthetic municipal wastewater". In: *Department of Applied Mathematics Biometrics and Process Control, University of Gent, Belgium* (2001).

[80] Hussein Janna. "Characterisation of raw sewage and performance evaluation of al-diwaniyah sewage treatment work, iraq". In: *World Journal of Engineering and Technology* 4.2 (2016), pp. 296–304.

[81] `https://aida.unicas.it/data/JKSU_2022.zip`. [Accessed 15-10-2024].

[82] Yue Zhao, Zain Nasrullah, and Zheng Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7. URL: http://jmlr.org/papers/v20/19-011.html.

[83] Susmita Ray. "An Analysis of Computational Complexity and Accuracy of Two Supervised Machine Learning Algorithms—K-Nearest Neighbor and Support Vector Machine". In: *Data Management, Analytics and Innovation*. Ed. by Neha Sharma et al. Singapore: Springer Singapore, 2021, pp. 335–347.

[84] *Azienda Comunale Energia e Ambiente*. https://www.acea.it/. [Online: accessed 11-November-2022]. 2022.

[85] Petr Vanýsek. "ELECTROCHEMICAL SERIES". In: 2010.

[86] Yann LeCun, Y. Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (May 2015), pp. 436–44. DOI: 10.1038/nature14539.

[87] Ashish Vaswani et al. "Attention Is All You Need". In: *Proceedings of NIPS*. 2017.

[88] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21 (2020), pp. 1–67.

[89] M. Molinara et al. "A new Dataset for Detection of Illegal or Suspicious Spilling in Wastewater through Low-cost Real-time Sensors". In: *2022 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2022, pp. 293–298. DOI: 10.1109/SMARTCOMP55677.2022.00070.

[90] Yoav Freund and Robert E. Schapire. "Experiments with a New Boosting Algorithm". In: *ICML*. 1996.

[91] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32.

[92] Leo Breiman. "Bagging predictors". In: *Machine Learning* 24.2 (Aug. 1996), pp. 123–140.

[93] Eibe Frank and Bernhard Pfahringer. "Improving on bagging with input smearing". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2006, pp. 97–106.

[94] T. Cover and P. Hart. "Nearest neighbor pattern classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27. DOI: 10.1109/TIT.1967.1053964.

[95] Corinna Cortes and Vladimir Vapnik. "Support-vector networks". In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297.

[96] Ian H. Witten, Eibe Frank, and Mark A. Hall. "Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)". In: *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*. 3rd ed. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2011, pp. 1–621. ISBN: 978-0-12-374856-0. DOI: https://doi.org/10.1016/C2009-0-19715-5.

[97] Mary Phuong and Marcus Hutter. "Formal algorithms for transformers". In: *arXiv preprint arXiv:2207.09238* (2022).

[98] Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. "The devil is in the detail: Simple tricks improve systematic generalization of transformers". In: *arXiv preprint arXiv:2108.12284* (2021).

[99] Mengdi Xu et al. "Prompting Decision Transformer for Few-Shot Policy Generalization". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 17–23 Jul 2022, pp. 24631–24645. URL: https://proceedings.mlr.press/v162/xu22g.html.

[100] Yuxuan Li and James L McClelland. "Systematic Generalization and Emergent Structures in Transformers Trained on Structured Tasks". In: *arXiv preprint arXiv:2210.00400* (2022).

[101] Hyung Won Chung et al. "Scaling instruction-finetuned language models". In: *arXiv preprint arXiv:2210.11416* (2022).

[102] Ghufron Fathoni et al. "Comparison of State-of-Charge (SOC) estimation performance based on three popular methods: Coulomb counting, open circuit voltage, and Kalman filter". In: *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*. 2017, pp. 70–74. DOI: 10.1109/ICACOMIT.2017.8253389.

[103] G. Cerro et al. "Probe Localization by Magnetic Measurements in Eddy-Current Nondestructive Testing Environment". In: *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. 2018, pp. 45–49. DOI: 10.1109/MetroAeroSpace.2018.8453551.

[104] G. Cerro et al. "On a finite domain magnetic localization by means of TMR triaxial sensors". In: *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. 2020, pp. 1–6. DOI: 10.1109/I2MTC43012.2020.9129628.

[105] Nina Meddings et al. "Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review". In: *Journal of Power Sources* 480 (2020), p. 228742.

[106] Qunming Zhang et al. "Electrochemical Impedance Spectroscopy based State of Health Estimation for Lithium-ion Battery Considering Temperature and State of Charge Effect". In: *IEEE Transactions on Transportation Electrification* (2022), pp. 1–1. DOI: 10.1109/TTE.2022.3160021.

[107] Fan Yu et al. "Design of Multi-channel EIS Measurement System for Lithium-ion Batteries". In: *2020 11th International Conference on Prognostics and System Health Management (PHM-2020 Jinan)*. 2020, pp. 179–185. DOI: 10.1109/PHM-Jinan48558.2020.00039.

[108] Shriram Santhanagopalan and Ralph E White. "Online estimation of the state of charge of a lithium ion cell". In: *Journal of power sources* 161.2 (2006), pp. 1346–1355.

[109] Rui Xiong et al. "Critical Review on the Battery State of Charge Estimation Methods for Electric Vehicles". In: *IEEE Access* 6 (2018), pp. 1832–1843. DOI: 10.1109/ACCESS.2017.2780258.

[110] Dickson N. T. How et al. "State of Charge Estimation for Lithium-Ion Batteries Using Model-Based and Data-Driven Methods: A Review". In: *IEEE Access* 7 (2019), pp. 136116–136136. DOI: 10.1109/ACCESS.2019.2942213.

[111] Qian-Kun Wang et al. "State of Charge-Dependent Polynomial Equivalent Circuit Modeling for Electrochemical Impedance Spectroscopy of Lithium-Ion Batteries". In: *IEEE Transactions on Power Electronics* 33.10 (2018), pp. 8449–8460. DOI: 10.1109/TPEL.2017.2780184.

[112] Ji'ang Zhang et al. "Variable-Order Equivalent Circuit Modeling and State of Charge Estimation of Lithium-Ion Battery Based on Electrochemical Impedance Spectroscopy". In: *Energies* 14.3 (2021). ISSN: 1996-1073. DOI: 10.3390/en14030769. URL: https://www.mdpi.com/1996-1073/14/3/769.

[113] Iman Babaeiyazdi, Afshin Rezaei-Zare, and Shahab Shokrzadeh. "State of charge prediction of EV Li-ion batteries using EIS: A machine learning approach". In: *Energy* 223 (2021), p. 120116.

[114] G. Di Capua et al. "Using Genetic Programming to Learn Behavioral Models of Lithium Batteries". In: *Applications of Evolutionary Computation*. Ed. by João Correia, Stephen Smith, and Raneem Qaddoura. Cham: Springer Nature Switzerland, 2023, pp. 461–474. ISBN: 978-3-031-30229-9.

[115] C. Bourelly et al. "GA-Based Features Selection for Electro-chemical Impedance Spectroscopy on Lithium Iron Phosphate Batteries". In: *presented at 2023 International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles & International Transportation Electrification Conference (ESARS-ITEC)*. 2023.

[116] Emanuel Parzen. "On Estimation of a Probability Density Function and Mode". In: *The Annals of Mathematical Statistics* 33.3 (1962), pp. 1065–1076. DOI: 10.1214/aoms/1177704472. URL: https://doi.org/10.1214/aoms/1177704472.

[117] Leo Wildfeuer, Philipp Gieler, and Alexander Karger. "Combining the distribution of relaxation times from EIS and time-domain data for parameterizing equivalent circuit models of lithium-ion batteries". In: *Batteries* 7.3 (2021), p. 52.

[118] Md Sazzad Hosen et al. "Impact of relaxation time on electrochemical impedance spectroscopy characterization of the most common lithium battery technologies-experimental study and chemistry-neutral modeling". In: *World Electric Vehicle Journal* 12.2 (2021), p. 77.

[119] Muhammad Sharif Uddin et al. "Online bad data detection using kernel density estimation". In: *2015 IEEE Power & Energy Society General Meeting*. 2015, pp. 1–5. DOI: 10.1109/PESGM.2015.7286013.

[120] Simon J. Sheather. "Density Estimation". In: *Statistical Science* 19.4 (2004), pp. 588–597. ISSN: 08834237. URL: http://www.jstor.org/stable/4144429 (visited on 04/12/2023).

[121] Long Zhou et al. "State Estimation Models of Lithium-Ion Batteries for Battery Management System: Status, Challenges, and Future Trends". In: *Batteries* 9.2 (2023). ISSN: 2313-0105. DOI: 10.3390/batteries9020131. URL: https://www.mdpi.com/2313-0105/9/2/131.

[122] Wen-Yeau Chang. "The State of Charge Estimating Methods for Battery: A Review". In: *International Scholarly Research Notices* 2013.1 (2013), p. 953792. DOI: https://doi.org/10.1155/2013/953792. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2013/953792. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2013/953792.

[123] Mona Faraji Niri et al. "Remaining energy estimation for lithium-ion batteries via Gaussian mixture and Markov models for future load prediction". In: *Journal of Energy Storage* 28 (2020), p. 101271. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2020.101271. URL: https://www.sciencedirect.com/science/article/pii/S2352152X19312502.

[124] Mona Faraji Niri et al. "State of Power Prediction for Lithium-Ion Batteries in Electric Vehicles via Wavelet-Markov Load Analysis". In: *IEEE Transactions on Intelligent Transportation Systems* 22.9 (2021), pp. 5833–5848. DOI: 10.1109/TITS.2020.3028024.

[125] Thomas Gallien et al. "Magnetism Versus LiFePO4 Battery's State of Charge: A Feasibility Study for Magnetic-Based Charge Monitoring". In: *IEEE Transactions on Instrumentation and Measurement* 64.11 (2015), pp. 2959–2964. DOI: 10.1109/TIM.2015.2437634.

[126] Zichuan Ni and Ying Yang. "A Combined Data-Model Method for State-of-Charge Estimation of Lithium-Ion Batteries". In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–11. DOI: 10.1109/TIM.2021.3137550.

[127] Kiarash Movassagh et al. "A Critical Look at Coulomb Counting Approach for State of Charge Estimation in Batteries". In: *Energies* 14.14 (2021). ISSN: 1996-1073. DOI: 10.3390/en14144074. URL: https://www.mdpi.com/1996-1073/14/14/4074.

[128] Kiarash Movassagh, Sheikh Arif Raihan, and Balakumar Balasingam. "Performance Analysis of Coulomb Counting Approach for State of Charge Estimation". In: *2019 IEEE Electrical Power and Energy Conference (EPEC)*. 2019, pp. 1–6. DOI: 10.1109/EPEC47565.2019.9074781.

[129] Ines Baccouche et al. "Implementation of an Improved Coulomb-Counting Algorithm Based on a Piecewise SOC-OCV Relationship for SOC Estimation of Li-IonBattery". In: *CoRR* abs/1803.10654 (2018). arXiv: 1803.10654. URL: http://arxiv.org/abs/1803.10654.

[130] Zhongxiao Liu et al. "Accurate and Efficient Estimation of Lithium-Ion Battery State of Charge with Alternate Adaptive Extended Kalman Filter and Ampere-Hour Counting Methods". In: *Energies* 12.4 (2019). ISSN: 1996-1073. DOI: 10.3390/en12040757. URL: https://www.mdpi.com/1996-1073/12/4/757.

[131] Wentao Wang et al. "End-Cloud Collaboration Approach for State-of-Charge Estimation in Lithium Batteries Using CNN-LSTM and UKF". In: *Batteries* 9.2 (2023). ISSN: 2313-0105. DOI: 10.3390/batteries9020114. URL: https://www.mdpi.com/2313-0105/9/2/114.

[132] Jiazhi Miao et al. "State of Charge Estimation of Lithium-Ion Battery for Electric Vehicles under Extreme Operating Temperatures Based on an Adaptive Temporal Convolutional Network". In: *Batteries* 8.10 (2022). ISSN: 2313-0105. DOI: 10.3390/batteries8100145. URL: https://www.mdpi.com/2313-0105/8/10/145.

[133] Peter Fundaro. "Impedance Track™ Based Fuel Gauging". In: *Texas Instruments* (2007), pp. 1–11.

[134] Ke Zhu, Xuyang Liu, and Philip W. T. Pong. "Performance Study on Commercial Magnetic Sensors for Measuring Current of Unmanned Aerial Vehicles". In: *IEEE Transactions on Instrumentation and Measurement* 69.4 (2020), pp. 1397–1407. DOI: 10.1109/TIM.2019.2910339.

[135] Chang Yoon Chun et al. "Current sensor-less state-of-charge estimation algorithm for lithium-ion batteries utilizing filtered terminal voltage". In: *Journal of Power Sources* 273 (2015), pp. 255–263. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2014.08.121. URL: https://www.sciencedirect.com/science/article/pii/S0378775314013962.

[136] Marco Crescentini et al. "Online EIS and Diagnostics on Lithium-Ion Batteries by Means of Low-Power Integrated Sensing and Parametric Modeling". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–11. DOI: 10.1109/TIM.2020.3031185.

[137] Giuseppe Manfredini et al. "An ASIC-Based Miniaturized System for Online Multi-Measurand Monitoring of Lithium-Ion Batteries". In: *Batteries* 7.3 (2021). ISSN: 2313-0105. DOI: 10.3390/batteries7030045. URL: https://www.mdpi.com/2313-0105/7/3/45.

[138] *Li-Polymer Battery 3.7V 2000mAh — mikroe.com*. https://www.mikroe.com/li-polymer-battery-37v-2000mah. [Accessed 18-10-2024].

[139] IEC IEC. "62660-2: 2018; Secondary Lithium-Ion Cells for the Propulsion of Electric Road Vehicles—Part 2: Reliability and Abuse Testing". In: *International Electrotechnical Commission: Geneva, Switzerland* (2018).

[140] Hans-Georg Schweiger et al. "Comparison of Several Methods for Determining the Internal Resistance of Lithium Ion Cells". In: *Sensors* 10.6 (2010), pp. 5604–5625. ISSN: 1424-8220. DOI: 10.3390/s100605604. URL: https://www.mdpi.com/1424-8220/10/6/5604.

[141] United Nations. *Addendum 15: Global Technical Regulation No. 15—Worldwide Harmonized Light Vehicles Test Procedure*. 2014.

[142] Szabolcs Kocsis Szürke, Adrienn Dineva, and Bence Csomós. *Complex testing procedure for 18650 batteries*. 2021. DOI: 10.21227/gg24-sj16. URL: https://dx.doi.org/10.21227/gg24-sj16.

[143] Adrienn Dineva et al. "Investigation of the performance of direct forecasting strategy using machine learning in State-of-Charge prediction of Li-ion batteries exposed to dynamic loads". In: *Journal of Energy Storage* 36 (2021), p. 102351. ISSN: 2352-152X. DOI: https://doi.org/10.1016/j.est.2021.102351. URL: https://www.sciencedirect.com/science/article/pii/S2352152X21001122.

[144] BIPM JCGM. *International vocabulary of metrology—Basic and general concepts and associated terms (VIM)*. 2012.

[145] S.S. Zhang, K. Xu, and T.R. Jow. "The low temperature performance of Li-ion batteries". In: *Journal of Power Sources* 115.1 (2003), pp. 137–140. ISSN: 0378-7753. DOI: https://doi.org/10.1016/S0378-7753(02)00618-3. URL: https://www.sciencedirect.com/science/article/pii/S0378775302006183.

[146] Foad H. Gandoman et al. "Understanding Voltage Behavior of Lithium-Ion Batteries in Electric Vehicles Applications". In: *Batteries* 8.10 (2022). ISSN: 2313-0105. DOI: 10.3390/batteries8100130. URL: https://www.mdpi.com/2313-0105/8/10/130.

[147] Chi Nguyen Van. "Optimal Control of Active Cell Balancing for Lithium-Ion Battery Pack With Constraints on Cells' Current and Temperature". In: *Journal of Electrochemical Energy Conversion and Storage* 20.1 (2023), p. 011009.

[148] Arzu Turksoy and Ahmet Teke. "A fast and energy-efficient nonnegative least square-based optimal active battery balancing control strategy for electric vehicle applications". In: *Energy* 262 (2023), p. 125409.

[149] Zachary Bosire Omariba, Lijun Zhang, and Dongbai Sun. "Review of Battery Cell Balancing Methodologies for Optimizing Battery Pack Performance in Electric Vehicles". In: *IEEE Access* 7 (2019), pp. 129335–129352. DOI: 10.1109/ACCESS.2019.2940090.

[150] Ghufron Fathoni et al. "Comparison of State-of-Charge (SOC) estimation performance based on three popular methods: Coulomb counting, open circuit voltage, and Kalman filter". In: *2017 2nd International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*. 2017, pp. 70–74. DOI: 10.1109/ICACOMIT.2017.8253389.

[151] Evgenij Barsoukov and J. Ross Macdonald, eds. *Impedance Spectroscopy: Theory, Experiment, and Applications*. en. 1st ed. Wiley, Apr. 2005. ISBN: 9780471647492 9780471716242. DOI: 10.1002/0471716243. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/0471716243 (visited on 12/17/2021).

[152] Nina Meddings et al. "Application of electrochemical impedance spectroscopy to commercial Li-ion cells: A review". In: *Journal of Power Sources* 480 (2020), p. 228742. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2020.228742.

[153] Markos Koseoglou et al. "A novel on-board electrochemical impedance spectroscopy system for real-time battery impedance estimation". In: *IEEE Transactions on Power Electronics* 36.9 (2021), pp. 10776–10787.

[154] Andreas Mertens et al. "Quantitative analysis of time-domain supported electrochemical impedance spectroscopy data of Li-ion batteries: reliable activation energy determination at low frequencies". In: *Journal of the Electrochemical Society* 163.7 (2016), H521.

[155] Thomas F Landinger, Guenter Schwarzberger, and Andreas Jossen. "A novel method for high frequency battery impedance measurements". In: *2019 IEEE International Symposium on Electromagnetic Compatibility, Signal & Power Integrity (EMC+ SIPI)*. IEEE. 2019, pp. 106–110.

[156] Jundong Li et al. "Feature Selection: A Data Perspective". In: *ACM Comput. Surv.* 50.6 (Dec. 2017).

[157] Iman Babaeiyazdi, Afshin Rezaei-Zare, and Shahab Shokrzadeh. "State of charge prediction of EV Li-ion batteries using EIS: A machine learning approach". In: *Energy* 223 (2021), p. 120116.

[158] Yuanyuan Li et al. "On the feature selection for battery state of health estimation based on charging–discharging profiles". In: *Journal of Energy Storage* 33 (2021), p. 102122.

[159] Tareq M. Shami et al. "Particle Swarm Optimization: A Comprehensive Survey". In: *IEEE Access* 10 (2022), pp. 10031–10061. DOI: 10.1109/ACCESS.2022.3142859.

[160] G. Di Capua et al. "Using Genetic Programming to Learn Behavioral Models of Lithium Batteries". In: *Applications of Evolutionary Computation*. Ed. by João Correia, Stephen Smith, and Raneem Qaddoura. Cham: Springer Nature Switzerland, 2023, pp. 461–474. ISBN: 978-3-031-30229-9.

[161] Gianni Cerro et al. "Probe localization by magnetic measurements in eddy-current nondestructive testing environment". In: *2018 5th IEEE International Workshop on Metrology for AeroSpace (MetroAeroSpace)*. IEEE. 2018, pp. 45–49.

[162] Gianni Cerro et al. "On a finite domain magnetic localization by means of TMR triaxial sensors". In: *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*. IEEE. 2020, pp. 1–6.

[163] Ali Abdollahi et al. "Optimal battery charging, Part I: Minimizing time-to-charge, energy loss, and temperature rise for OCV-resistance battery model". In: *Journal of Power Sources* 303 (2016), pp. 388–398.

[164] Qiu-An Huang et al. "Impedance characteristics and diagnoses of automotive lithium-ion batteries at 7.5% to 93.0% state of charge". In: *Electrochimica Acta* 219 (2016), pp. 751–765.

[165] Margherita Grandini, Enrico Bagli, and Giorgio Visani. "Metrics for multi-class classification: an overview". In: *arXiv preprint arXiv:2008.05756* (2020).

[166] Mandavilli Srinivas and Lalit M Patnaik. "Genetic algorithms: A survey". In: *computer* 27.6 (1994), pp. 17–26.

[167] J. Kennedy and R. Eberhart. "Particle swarm optimization". In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. Vol. 4. 1995, 1942–1948 vol.4. DOI: 10.1109/ICNN.1995.488968.

[168] Federico Marini and Beata Walczak. "Particle swarm optimization (PSO). A tutorial". In: *Chemometrics and Intelligent Laboratory Systems* 149 (2015), pp. 153–165. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2015.08.020. URL: https://www.sciencedirect.com/science/article/pii/S0169743915002117.

[169] Mojtaba Ahmadieh Khanesar, Mohammad Teshnehlab, and Mahdi Aliyari Shoorehdeli. "A novel binary particle swarm optimization". In: *2007 Mediterranean Conference on Control & Automation*. 2007, pp. 1–6. DOI: 10.1109/MED.2007.4433821.