

Democratic and Popular Republic of Algeria
Ministry of Higher Education and Scientific Research

Abbes Laghrour University -Khenchela-

Faculty of Sciences and Technologies

Department of Mathematics and Computer Science

Order number:
Serial number:



A Machine Learning-Based Approach for Managing Cyber-Physical Systems

Thesis presented by

Nabila AZERI

To obtain the degree of

DOCTOR IN COMPUTER SCIENCE

(Option: Software Engineering and Distributed Systems)

Presented publicly on: December 19th, 2024

Composition of the jury:

N°	First & Family name	Grade	Establishment	Quality
01	Zianou AHMED SEGHIR	PROF	Abbes Laghrour University, Khenchela	President
02	Ouassila HIOUAL	PROF	Abbes Laghrour University, Khenchela	Supervisor
03	Ouided HIOUAL	MCA	Abbes Laghrour University, Khenchela	Co-Supervisor
04	Nabil MESSAOUDI	MCA	Abbes Laghrour University, Khenchela	Examiner
05	Said BRAHIMI	MCA	08 MAI 1945 University, Guelma	Examiner
06	Tahar GUERRAM	MCA	Larbi Ben M'hidi University, Oum El Bouaghi	Examiner



A Machine Learning-Based Approach for Managing Cyber-Physical Systems

By:

Nabila Azeri

Supervisor:

Pr Ouassila Hioual

Co-Supervisor:

Dr Ouided Hioual

This thesis proposal was submitted in Partial Fulfillment of the
Requirements for the PhD's Degree
In Computer Science
Abbes Laghrour University

19/12/ 2024

Dedication

This thesis is dedicated to my beloved family:

To my dear parents, whose unconditional love, support, and prayers have always been my guiding light.

To my wonderful sister and brothers, for their constant encouragement and belief in me.

To my husband, whose unwavering support and patience have been my greatest strength.

And to my precious children, who are my source of joy and inspiration.

This work would not have been possible without each of you.

Acknowledgments

I would like to express my deepest gratitude to my thesis supervisor, **Prof. Hioual Ouassila**, for her invaluable guidance, support, and encouragement throughout this journey. Her expertise and dedication have been instrumental in shaping the direction of my research.

I am also profoundly thankful to my co-supervisor, **Dr. Hioual Ouided**, for her insightful feedback, constant support, and constructive advice, which greatly enriched the quality of this thesis.

I extend my sincere appreciation to the distinguished members of my thesis jury, **Prof. Ahmed Seghir Zianou**, **Dr. Messaoudi Nabil**, **Dr. Brahimi Said**, and **Dr. Guerram Tahar**, for kindly accepting to participate in the evaluation of my work.

Finally, I would like to thank the faculty and staff of **Université Abbès Laghrour, Khenchela**, for their academic and administrative support. To my family and friends, your unwavering encouragement and patience have been my greatest source of strength.

List of Abbreviations

COAP Constrained Application Protocol. 8

CPS Cyber-Physical Systems. 1

FedAvg Federated Averaging algorithm. 81

FML Federated Machine Learning. 3

KNN K-Nearest Neighbors. 77

ML Machine Learning. 1, 2

MQTT Message Queuing Telemetry Transport. 8

OPC-UA Open Platform Communications Unified Architecture. 8

RF Random Forest. 77

SML Split Machine Learning. 3

SVM Support Vector Machine. 77

TFF Federated TensorFlow. 81

Table of Contents

Dedication	i
Acknowledgments	ii
Table of Contents	iv
List of Figures	viii
List of Tables	ix
Abstract	x
Résumé	xi
Abstract in arabic	xii
Introduction	1
1 Research context and problem statement	1
2 Research objectives	2
3 Research contributions	2
4 Thesis organization	4
1 Cyber-Physical Systems and Self-Adaptive Systems	6
1 Fundamental concepts of CPS	6
1.1 Definitions of CPS	6
1.2 CPS Components	7
1.3 Characteristics of CPS	9
1.4 Applications of CPS	9
2 Self-Adaptive Systems	12
2.1 Self-adaptation	14
2.2 Taxonomy for self-adaptation	14
2.3 Adaptation process	16
2.4 Structure of self-adaptive systems	17
3 Conclusion	19

2	Machine Learning and Collaborative Machine Learning	20
1	Machine Learning	20
1.1	Definition of Machine Learning	20
1.2	ML versus Traditional programming	21
1.3	Types of ML	21
1.4	Machine learning pipeline	26
2	Federated Machine Learning	30
2.1	Definition and Relevance of Federated Machine Learning	30
2.2	Principle of Operation of FML	31
2.3	Main Algorithms for Model Aggregation	34
3	Split Machine Learning	36
3.1	Definition and Relevance of SML	36
3.2	Split Learning Configurations	36
4	Conclusion	37
3	Machine Learning for Enhancing Adaptability in CPS : A Literature Review	38
1	Adaptability Challenges in CPS	38
2	ML Advantages for Enhanced CPS Adaptability	40
3	Review of ML Approaches for Adaptability in CPS	40
3.1	Related Work	41
3.2	Discussion	48
4	Conclusion	49
4	Proposed Framework for Adaptive Cyber-Physical Systems	51
1	Meta-Model for Adaptive CPS	51
2	Centralized ML-Based Architecture for CPS	55
2.1	Physical layer	56
2.2	Data/Resource Processing Layer	57
2.3	Data Storage Layer	58
2.4	Learning Application Layer	58
3	Limitations of our Centralized ML-Based Architecture	59
3.1	Data Security and Privacy Concerns	59
3.2	Scalability Concerns	59
3.3	Communication Overhead	60
4	Towards a Privacy-Enhanced Architecture Empowered by Collaborative ML	60

4.1	The Proposed FML-Based Architecture for Enhancing Data Privacy in CPS	60
4.2	The Proposed SML-Based Architecture for Resource-Constrained CPS	64
5	Conclusion	69
5	Case Study - Fault Prediction in CPS Using Centralized/Decentralized ML	70
1	Fault Prediction Module Structure	70
2	Dataset Description	72
3	Implementation with Centralized Architecture	72
3.1	Data preparation	73
3.2	Choosing ML Algorithms for Fault Prediction	76
3.3	Deploying the Trained Model for Real-world Predictions	80
4	Implementation with Decentralized Architecture	81
4.1	Data Preparation	82
4.2	Training process in FML-based architecture	82
4.3	Training process in SML-based architecture	86
4.4	Comparative Analysis	88
5	Conclusion	90
	Conclusion	92
1	Research summary	92
2	Future work	93
	Annex 1 : Implementation Details of the Centralized Architecture	94
1	Libraries Used for Implementation	94
1.1	Scikit-learn	94
1.2	Pandas	94
1.3	NumPy	94
1.4	Matplotlib	95
1.5	Seaborn	95
2	Work Environment	95
2.1	Software Environment	95
2.2	Hardware Environment	96
	Annex 2 : Implementation Details of the Decentralized Architecture	97
1	FML-Based Platforms	97

2	Comparative Studies of Collaborative-Based Platforms	98
3	Implementation of the FML-based architecture with TFF	100
4	Vanilla SML Implementation with TensorFlow Keras	101
Publications During the Thesis		103
Bibliography		104

List of Figures

1.1	CPS Components	7
1.2	Some applications of CPS [2]	10
1.3	Three-dimensional model of the RAMI 4.0. [85]	13
1.4	IIRA function domains and control flows [18]	13
1.5	Adaptation taxonomy [60]	15
1.6	MAPE-K Architecture [87]	16
1.7	Generic conceptual architecture of a self-adaptive system [1]	18
2.1	ML versus Traditional Programming [23]	22
2.2	Classification [102]	23
2.3	Regression [103]	23
2.4	Overview of reinforcement learning	25
2.5	ML pipeline	26
2.6	Basic principle of FML [128]	32
2.7	Types of FML [128]	33
4.1	A meta-model for CPS	52
4.2	The proposed architecture	56
4.3	The proposed FML-based architecture	61
4.4	The proposed SML-based architecture	66
5.1	Fault Prediction Operating Principle	71
5.2	Correlation matrix between the different features	74
5.3	Bivariate relationship between each pair of features	75
5.4	Confusion Matrix	78
5.5	Elbow method curve	80
5.6	Clustering result in 3D	81
5.7	Data visualization	83
5.8	Loss values vs number of rounds	84
5.9	Accuracy vs number of rounds	85
5.10	Loss and accuracy convergence for each client	87
5.11	Total learning time for each client in FML and SML	89

List of Tables

1.1	Relation of the taxonomy dimensions and the 5W+1H questions	16
2.1	Comparison of ML and Traditional Programming	21
2.2	Commonly Used Machine Learning Algorithms	28
3.1	Comparative overview of ML applications in CPS	41
5.1	Scores of each features	76
5.2	The obtained accuracy of each classifier	77
5.3	Performance Metrics for Classification Using RF	79
5.4	Comparison of accuracy metrics across different ML approaches	88
5.5	Comparison of learning time between FML and SML for each client	89
5.6	Comparison of FML Platforms	99

Abstract

The increasing prevalence of Cyber-Physical Systems (CPS) demands adaptive systems capable of responding to dynamic environmental changes. Traditionally, ensuring quality of service in CPS involves pre-planned activities at design time, posing challenges in highly dynamic contexts. This thesis investigates the integration of Machine Learning (ML) to bolster CPS adaptability, emphasizing security and privacy.

The primary objective is to employ ML algorithms for real-time monitoring and dynamic adaptation of CPS behavior in response to environmental changes. Key contributions of this thesis encompass a meta-model for adaptive CPS, providing a conceptual foundation to comprehend and model interactions within CPS. In the second contribution, a ML-based architecture is proposed, enhancing CPS functionalities, including data and resource management, fault prediction, and proactive maintenance. A novel Collaborative ML-based architecture, founded on the second contribution, is introduced, highlighting scalability, security, heightened data privacy, and its suitability for resource-constrained environments. This Collaborative ML approach encompasses Federated Machine Learning (FML) and Split Machine Learning (SML). Validation is conducted through a case study on fault prediction in industrial CPS, and the experimentation results showcase promising outcomes.

Keywords: CPS, Adaptability, ML, FML, SML, Meta-model, Multi-layer Architecture.

Résumé

Les systèmes cyber-physiques (CPS) sont de plus en plus répandus et exigent des systèmes adaptables capables de répondre aux changements dynamiques de l'environnement. Traditionnellement, la garantie de la qualité de service dans les CPS repose sur des activités planifiées à l'avance, ce qui pose des défis dans des contextes hautement dynamiques. Cette thèse explore l'intégration de Machine Learning (ML) pour renforcer l'adaptabilité des CPS, en mettant l'accent sur la sécurité et la confidentialité.

L'objectif principal est d'utiliser des algorithmes de ML pour la surveillance en temps réel et l'adaptation dynamique du comportement des CPS en réponse aux changements environnementaux. Les principales contributions de cette thèse comprennent un méta-modèle pour les CPS adaptatifs, fournissant une base conceptuelle pour comprendre et modéliser les interactions au sein des CPS. Dans la deuxième contribution, une architecture basée sur le ML est proposée, améliorant les fonctionnalités des CPS, notamment la gestion des données et des ressources, la prédiction des défaillances et la maintenance proactive. Ensuite, une nouvelle architecture basée sur le ML collaboratif, fondée sur la deuxième contribution, est introduite, mettant en avant la scalabilité, la sécurité, la confidentialité accrue des données et son adéquation aux environnements à ressources limitées. Cette approche de ML collaboratif englobe l'apprentissage fédéré (FML) et l'apprentissage par découpage (SML). La validation est effectuée à travers une étude de cas sur la prédiction des défaillances dans les CPS industriels, et les résultats des expérimentations présentent des résultats prometteurs.

Mots clés: CPS, Adaptation, ML, FML, SML, Meta-model, Architecture multi-couche.

الملخص

الملخص :

تزايد انتشار الأنظمة السيبرانية الفيزيائية (CPS) يتطلب أنظمة قادرة على التكيف مع التغيرات البيئية الديناميكية. سابقاً، كان ضمان جودة الخدمة في أنظمة CPS يتطلب أنشطة مخططة مسبقاً أثناء وقت التصميم، مما يطرح تحديات في السياقات العالية الديناميكية. تتناول هذه الرسالة استعمالات تعلم الآلة (ML) لتعزيز قابلية تكيف أنظمة CPS ، مع التركيز على الأمان والخصوصية.

الهدف الرئيسي هو استخدام خوارزميات تعلم الآلة للمراقبة في الوقت الفعلي والتكيف الديناميكي لسلوك أنظمة CPS استجابةً لتغيرات البيئة. تشمل المساهمات الرئيسية لهذه الرسالة نموذجاً شاملاً للتكيف في أنظمة CPS ، مما يوفر أساساً مفاهيمياً لتصميم ونمذجة التفاعلات ضمن أنظمة CPS . في المساهمة الثانية، تم اقتراح بنية متعددة المستويات تعتمد على تعلم الآلة، مما يعزز وظائف أنظمة CPS ، بما في ذلك إدارة البيانات والموارد، وتنبؤ بالأعطال، والصيانة الاستباقية. في المساهمة الثالثة يتم ادماج تعلم الآلة التعاوني والمتمثل في التعلم الآلي الموحد (FML) و التعلم الآلي المقسم (SML) . هذه البنية والتي تعتمد على المساهمة الثانية، تسلط الضوء على قابلية التوسع، والأمان، وتعزيز خصوصية البيانات، وملاءمتها للبيئات ذات الموارد المحدودة. يتم التحقق من الصحة من خلال دراسة حالة حول التنبؤ بالأعطال في أنظمة CPS الصناعية، وتظهر نتائج التجارب نتائج واعدة.

الكلمات المفتاحية : الأنظمة السيبرانية الفيزيائية (CPS) ، التكيف ، التعلم الآلي (ML) ، التعلم الآلي الموحد (FML) ، التعلم الآلي المقسم (SML) ، بنية متعددة المستويات

Introduction

1 Research context and problem statement

In the ever-evolving landscape of technology, Cyber-Physical Systems (CPS) have emerged as the linchpin of our interconnected world, seamlessly merging the physical and digital realms [89]. These systems, which encompass everything from autonomous vehicles and industrial automation to smart cities and healthcare applications, are characterized by their ability to interact with the physical environment, process data, and make intelligent decisions in real-time [5][93].

Nowadays, CPS have evolved from stand-alone computers to highly dynamic and complex systems with access to rich context information. In such systems, it is critical to guarantee the quality of service, according to the requirements of the application domain. Traditionally, this is achieved through a variety of activities and processes performed at design time, such as requirements analysis and testing. However, in the context of CPS, some of quality of service satisfaction requirements may need to be performed at runtime. The latter requires the design of systems that ensure compliance with requirements through a sustainable process that continuously provides new solutions. This can be achieved by combining system-driven and human-driven activities to deal with the highly dynamic and uncertainties in CPS.

In fact, in a highly dynamic environment like that of a CPS, the complexity that matters is not the huge size of the code through which such systems are programmed but it lies in how they adapt their behavior to respond to the changes in the environment in which they will be involved. In several studies [22] [44], it is stated that the major motivation for dynamic adaptation is the uncertainty of the execution environment. In fact it is almost impossible to pre-plan at design time all possible combinations of environmental conditions that will be encountered.

To support developing such new functionalities, Machine Learning (ML) algorithms were merged. These latter allow to monitor the execution of systems. Then, they adapt, dynamically, their behaviour in response to environmental conditions changes. Nowadays, ML algorithms have achieved impressive results providing solutions to practical large-scale problems. ML offers a powerful set of approaches, techniques, and predictive models that can process a large amount of information and help solve many use cases efficiently [11][21]. In addition, ML identifies patterns automatically and provides predictive information to systems to make intelligent decisions in real time. It solves a variety of problems such as analysis and planning, decision making, predictive maintenance, and anomaly detection. Numerous studies have explored the utilization of ML techniques to develop flexible and adaptive CPS capable of responding to changing environmental conditions. These techniques often involve a central storage of data,

followed by the training of a model on this centralized data repository. This approach facilitates comprehensive analysis and decision-making on a global scale. However, this centralized approach gives rise to potential challenges concerning data privacy and security, as all the information is consolidated in a single location.

2 Research objectives

The main objective of this thesis is to utilize ML techniques to implement and improve adaptability in CPS while taking into account security and privacy aspects. To achieve this objective, the research focuses on the following specific objectives:

- Explore the application of ML algorithms to monitor and adapt CPS behavior in response to changing environmental conditions. This involves exploring how ML techniques can continuously analyze real-time data streams generated by CPS to identify patterns, anomalies, and potential threats. The insights extracted from this analysis will be utilized to trigger proactive adaptations, enabling CPS to respond effectively to dynamic environmental changes.
- Explore collaborative ML approaches to address scalability challenges, with a particular emphasis on data privacy. Centralized ML architectures offer a streamlined approach for processing large volumes of data, but they may raise concerns about data privacy and centralization. Conversely, collaborative ML architectures distribute ML computations across multiple nodes, potentially alleviating privacy concerns but introducing new challenges in terms of coordination and communication overhead. The research will investigate the trade-offs between these approaches and develop strategies for optimizing scalability while ensuring data privacy.

3 Research contributions

This thesis proposes a new approach for the development of self-adaptive CPS. The approach that we consider relies on the ML paradigm, which allows CPS to modify their behaviors at runtime in response to changes in the environment in order to ensure a certain quality level. More precisely, our main contributions are:

- **Proposition of a Meta-Model for Adaptive CPS:** Our thesis introduces a comprehensive meta-model for adaptive CPS that captures their essential characteristics and behaviors.

This meta-model provides a conceptual foundation for understanding and modeling the complex interactions between physical and digital components within CPS. It facilitates the design, implementation, and analysis of adaptive CPS by providing a unified framework for representing their structure, behavior, and adaptability mechanisms.

- **Development of a Centralized ML-Based Architecture for CPS:** We develop an architecture, which ensures the various functionalities of a CPS such as: data management, resources management, fault prediction, predictive maintenance, etc. To ensure these functionalities, our architecture uses advanced techniques such as ML and Cloud/Edge Computing. Through the integration of ML algorithms, the system gains the ability to analyze and extract meaningful insights from diverse data sources, facilitating functionalities like fault prediction and proactive maintenance. Furthermore, the utilization of Cloud/Edge Computing enhances the system's scalability and responsiveness, ensuring efficient resource management.
- **Development of a Privacy-Enhanced Architecture Empowered by Collaborative ML:** Building upon the second contribution, our third contribution introduces two novel architectures based on Federated Machine Learning (FML) and Split Machine Learning (SML). The first architecture, based on FML, addresses scalability challenges and enhances security and data privacy by distributing ML computations across multiple nodes. The second architecture, based on SML, tackles the problem of resource constraints in CPS by partitioning data and computation tasks, ensuring efficient use of limited resources while maintaining data privacy and security.
- **Validation of the Proposed Architectures: A Case Study on Fault Prediction in Industrial CPS Environments:** To validate our approach, we conduct a series of experiments using a real-world dataset, with a specific focus on fault prediction within industrial CPS. The fault prediction module, situated in the learning application layer of our architecture, is implemented using both centralized and decentralized ML approaches. Through a detailed comparison of results from both implementations, we gain valuable insights into the performance and suitability of each architecture for fault prediction within CPS.

4 Thesis organization

This thesis is structured into five chapters, with the first three chapters forming the state-of-the-art part.

Chapter 1: CPS and Adaptive Systems

This chapter provides an overview of Cyber-Physical Systems (CPS) and adaptive systems. It discusses the fundamental concepts, architectures, and challenges of CPS, as well as the different approaches to achieving adaptability in CPS.

Chapter 2: ML and Collaborative ML

This chapter focuses on machine learning (ML) and collaborative ML. It introduces the main concepts, algorithms, and applications of ML in CPS. It also discusses the challenges and opportunities of using collaborative ML for CPS applications.

Chapter 3: ML for Enhancing Adaptability in CPS : A Literature Review

This chapter presents an in-depth exploration of ML techniques aimed at enhancing adaptability in CPS. First, we present the main challenges impeding seamless adaptability in CPS, including dynamic environments, uncertainty, evolving requirements, fault tolerance, interoperability, and security. Next, we examine the advantages of ML for bolstering adaptability in CPS and review existing literature on ML approaches tailored for CPS environments.

Chapter 4: Proposed Framework for Adaptive Cyber-Physical Systems

This chapter presents the proposed framework for adaptive CPS. The framework consists of three main components:

- A meta-model for understanding and modeling CPS
- A centralized ML-based architecture to enhance adaptability and efficiency
- An enhancement to the centralized architecture using FML to address data privacy, scalability, and communication overhead, and SML to tackle resource constraints in CPS.

The chapter provides a detailed description of each component and explains how they work together to achieve the overall goal of adaptability.

Chapter 5: Case Study - Fault Prediction in CPS Using Centralized/Decentralized ML

This chapter focuses on the fault prediction module within the learning application layer of the proposed architecture. It describes the implementation of the fault prediction module using

both centralized and decentralized ML approaches. The chapter also presents a comparison of the results obtained from both implementations, offering insights into the performance and effectiveness of each architecture in the context of fault prediction within CPS.

1. Cyber-Physical Systems and Self-Adaptive Systems

Cyber-Physical Systems (CPS) represent a paradigm shift, bridging the virtual and physical domains. This chapter lays the groundwork for comprehending these intricate systems, first by meticulously defining and dissecting their key components. We then investigate the unique characteristics that set CPS apart, particularly their inherent dynamism and tight coupling with the physical world.

Recognizing the crucial role of adaptability in CPS, the latter section delves into the realm of Self-Adaptive Systems. We introduce the principle of self-adaptation, highlighting its various forms and the intricate processes that enable systems to adjust to changing environments in real-time.

1 Fundamental concepts of CPS

1.1 Definitions of CPS

In the literature, the term CPS has been defined in various ways. This section aims to provide an understanding of CPS by presenting key definitions and fundamental concepts associated with the term.

- Definition 1: According to NIST¹[38], CPS encompass a spectrum of components, including digital, analog, physical, and human-interactive elements. These components are intentionally designed to operate through embedded physics and logic. This definition underscores the holistic nature of CPS, emphasizing the integration of computational elements with the physical world. The inclusion of human-interactive components highlights the collaborative synergy between technology and human operators within the CPS framework.
- Definition 2: According to NSF²[32], CPS are engineered systems that are intricately built from, and reliant upon, the seamless integration of computation and physical components. They emerge through the synergy of three fundamental elements: (1) physical components, representing the tangible entities in the system; (2) virtual components,

¹National Institute of Standards and Technology

²The National Science Foundation

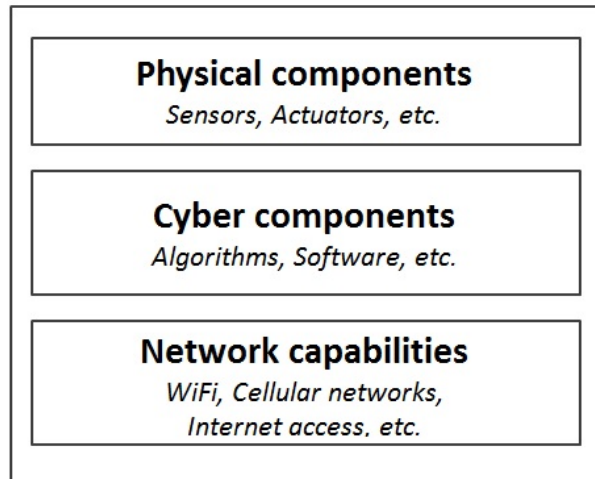


Figure 1.1: CPS Components

signifying the computational and algorithmic aspects; and (3) connectivity to digital networks, highlighting the role of communication and connectivity in the CPS ecosystem. This definition emphasizes the interdependence and collaboration among these three elements to create a cohesive and functional system.

Both the NIST and NSF definitions of CPS emphasize the integration of computation and physical components as a defining characteristic of these systems. The NIST definition highlights the holistic nature of CPS, emphasizing the integration of computational elements with the physical world. The NSF definition, on the other hand, emphasizes the interdependence and collaboration among three fundamental elements: physical components, virtual (or cyber) components, and connectivity to digital networks. This definition highlights the fact that CPS are complex systems that require the interaction of these three elements to function properly.

1.2 CPS Components

From the definitions discussed earlier, CPS comprise three foundational components: physical systems, cyber systems, and communication networks [7][48]. Figure 1.1 illustrates these three components, each with distinct functions that collectively enable CPS to sense, analyze, and react to the real-world environment in real-time.

1.2.1 Physical Systems

Physical systems represent the real-world components of a CPS, which can include sensors, actuators, and other physical devices. Physical systems interact with the environment and generate

data that is crucial for the functioning of the CPS.

- **Sensors:** are devices that detect and respond to changes in the environment. They capture data about the physical world and convert it into measurable electrical signals [98]. These signals are then processed and analyzed by the cyber system, enabling CPS to make real-time decisions and control the physical world.
- **Actuators:** Actuators are devices that convert electrical signals into physical actions, enabling CPS to control and manipulate the physical world [74]. They receive commands from the cyber system and translate them into physical actions, such as opening valves, starting motors, or adjusting positions.

Beyond sensors and actuators, the physical systems may include various physical entities such as industrial machines, appliances, infrastructure elements, and other tangible objects that interact with the environment [10].

1.2.2 Cyber Systems

The cyber component involves the computational and algorithmic aspects of the system. This includes the software, control algorithms, and processing units that analyze data from the physical systems and make decisions or generate responses.

1.2.3 Communication Networks

The communication component of CPS serves as the backbone of data exchange between the physical and cyber realms, enabling seamless coordination and synchronization of CPS operations. The selection of communication networks and protocols is guided by factors such as application requirements, data transmission rates, reliability, energy consumption, and security. Prevalent communication networks include wired (Ethernet, RS-485, and CAN bus) and wireless options (Wi-Fi, Bluetooth, and Zigbee), while common protocols encompass TCP/IP, UDP, MQTT³, COAP⁴, and the widely adopted OPC-UA⁵, which is specifically designed for CPS communication.

³Message Queuing Telemetry Transport (MQTT)

⁴Constrained Application Protocol (COAP)

⁵Open Platform Communications Unified Architecture (OPC-UA)

1.3 Characteristics of CPS

CPS systems are characterized by their ability to sense, process, and actuate on physical data in real-time [100]. They exhibit several key characteristics that distinguish them from traditional systems. These characteristics highlight the integrated nature of CPS, where the cyber and physical components collaborate to achieve specific functionalities. Some key characteristics of CPS include:

- **Real-time sensing, processing, and actuation:** CPS continuously gather data from the physical world through sensors, analyze it in real-time using computational algorithms, and trigger physical actions through actuators based on the analysis. This closed-loop operation allows for immediate and dynamic responses to changing environments.
- **Adaptability and flexibility:** CPS can adjust their behavior based on new information or changing contexts. This enables them to handle unforeseen situations and optimize performance in diverse environments.
- **Robust security:** CPS are designed with security in mind, incorporating features like encryption and authentication to protect against cyberattacks and unauthorized access.
- **Interconnectivity and communication:** CPS can communicate with other devices and systems, sharing data and collaborating to achieve shared goals. This interconnectedness enables efficient coordination and information sharing, leading to improved system-level performance.
- **Learning capabilities through machine learning:** CPS systems can be programmed to learn from their environment, employing machine learning algorithms to continuously enhance their performance over time. This learning capability contributes to ongoing system improvement. In this thesis, our focus revolves around exploring and detailing the learning capabilities of CPS through Machine Learning. Chapter 2 will comprehensively investigate the intricacies of Machine Learning and its application in enhancing the adaptability and performance of CPS.

1.4 Applications of CPS

In recent years, CPS have experienced remarkable expansion, gaining widespread applications in diverse engineering and industrial sectors. Notably, CPS has played a significant role in the

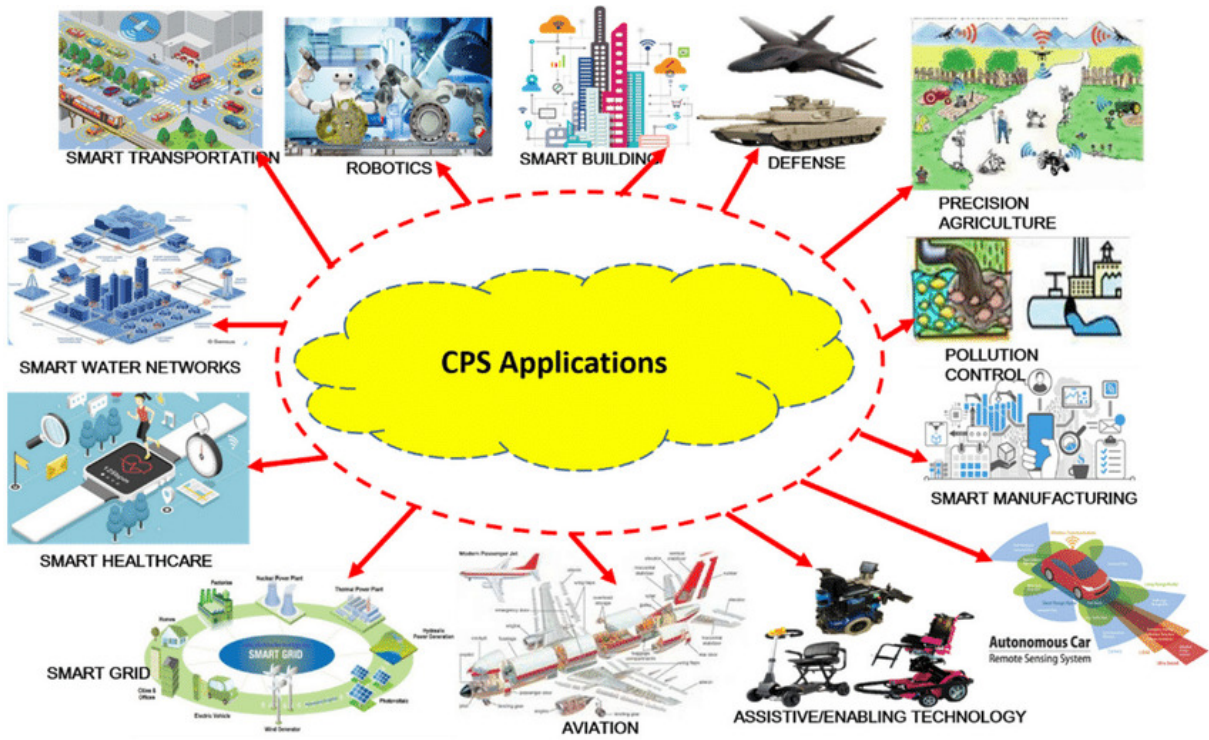


Figure 1.2: Some applications of CPS [2]

health sector, contributing to the development of intelligent healthcare systems [2]. Additionally, its influence extends to the production and manufacturing domain, where CPS is integral to the establishment of smart factories (or Industry 4.0). Figure 1.2 illustrates the extensive application of CPS in the transportation and vehicular sector, facilitating the realization of autonomous vehicles and intelligent transportation systems. This section unfolds a panoramic view of CPS applications focusing on the transformative impact of CPS within the industrial domain.

1.4.1 Transportation

The integration of CPS has revolutionized the transportation sector, enhancing efficiency, safety, and innovation. Intelligent transportation systems leverage CPS to create smart and adaptive solutions, leading to the development of autonomous vehicles, efficient traffic management, and advanced vehicle-to-vehicle communication. Works such as [126], [76], and [27] delve into the applications of CPS in transportation, highlighting the transformative impact on modern mobility solutions.

1.4.2 Healthcare sector

The integration of CPS has brought about transformative advancements in the healthcare sector, reshaping the delivery of medical services. Employing CPS, healthcare systems have witnessed the emergence of smart healthcare solutions, remote patient monitoring, and data-driven decision-making. Noteworthy research such as [30], [20], and [131] explores the applications of CPS in healthcare, shedding light on its potential to revolutionize patient care and medical practices.

1.4.3 Defense/military systems

The infusion of CPS has catalyzed transformative advancements within the defense and military sectors, revolutionizing strategic capabilities and operational methodologies. In the realm of military innovation, CPS is instrumental in fostering intelligent and adaptive solutions, leading to the development of autonomous defense systems, real-time threat analysis, and enhanced situational awareness. Insightful studies, exemplified by [54], [117], and [80], delve into the nuanced applications of CPS in defense, offering profound insights into its role in shaping modern military landscapes.

1.4.4 Industry 4.0

In the Manufacturing area, CPS are used for the smart manufacturing, smart production, robotics, and process control [85]. Among the various architectural frameworks proposed for CPS, the 5C Architecture [63] stands out as a prominent reference model widely employed in the development of such systems. With the fourth industrial revolution and the increase in data and industrial devices, other reference architectures were created with proposals related to CPS, such as RAMI 4.0 [26] in the manufacturing sector for device virtualization in the value chain, and IIRA [77] for integration and cooperation among industries.

- **RAMI 4.0 :** is an Architecture Reference Model for Industry 4.0. It introduces a vocabulary, semantics, and syntax that facilitate the integration of Internet of Things and services in the Industry 4.0 context. This Service Oriented Architecture amalgamates IT components to support key aspects of Industry 4.0, including horizontal integration among factory networks and plants, as well as vertical integration within a factory from products to the Cloud [96].

Represented as a three-dimensional map with axes for Hierarchy Levels, Product Lifecycle, and Architecture Layers, RAMI 4.0 addresses Industry 4.0 challenges systemati-

cally. Figure 1.3 shows the three-dimensional model of the RAMI 4.0. The first axis redefines the concept of Industry 3.0 by promoting flexible machines and systems, distributed functions over the network, and interconnected products. The second axis, focusing on the Product Life-cycle, delineates assets in the value chain from design to production and maintenance.

The third axis, central to CPS, encapsulates the Architecture Layers of RAMI 4.0 [66]. It vertically describes the physical entities of the industrial network, such as devices and machines, mapping them to virtual representations known as Industry 4.0 Components. These components provide a detailed description of the properties of a CPS, contributing to the comprehensive framework of RAMI 4.0.

- **IIRA** : is an open architecture for interoperability among industries [18]. It is structured into four viewpoints : Business, Usage, Implementation, and Functional. The Business Viewpoint focuses on identifying participants, their business views, values, and objectives in CPS systems. The Usage Viewpoint outlines the expectations of the CPS system in delivering intended business objectives. The Implementation Viewpoint specifies the technologies required for functional components, their communication schemes, and life-cycle procedures.

The Functional Viewpoint delves into functional components and their interrelation and interaction with external elements in the environment. Figure 1.4 illustrates the architecture of the Functional Viewpoint, along with Crosscutting Functions and Systems Characteristics. The Functional Viewpoint is further categorized into five domains—control, operation, information, application, and business, each responsible for describing essential system functions.

2 Self-Adaptive Systems

Unlike traditional systems, CPS exhibit a high degree of heterogeneity, involving diverse components that operate at different levels of abstraction. Moreover, the dynamic and often unpredictable nature of the physical world adds an additional layer of complexity. As a result, the design and development of CPS that can effectively navigate this complexity and remain in compliance with requirements have become a principal concern [35] [78].

In this section, we will focus on self-adaptive systems, exploring their design principles,

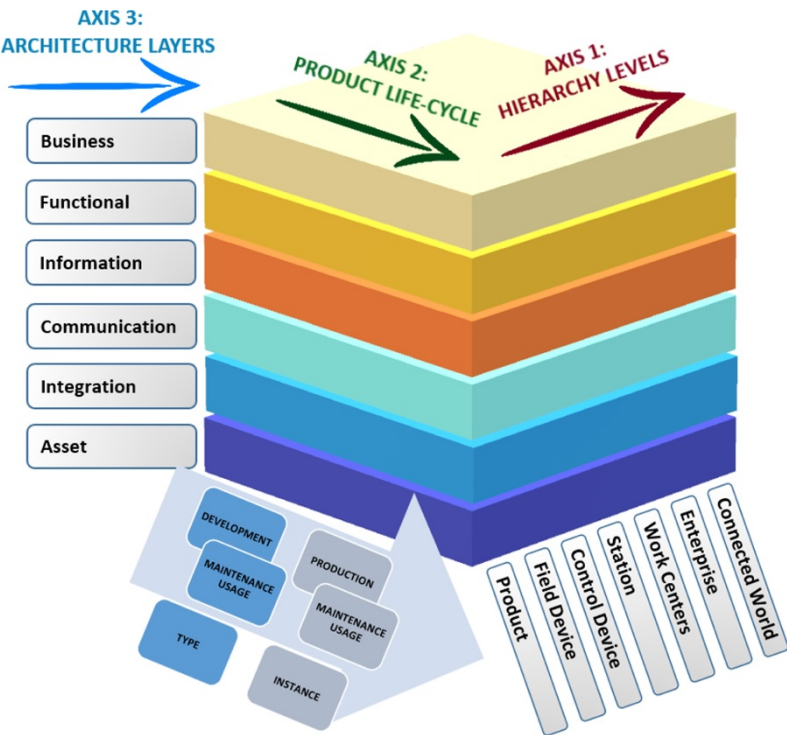


Figure 1.3: Three-dimensional model of the RAMI 4.0. [85]

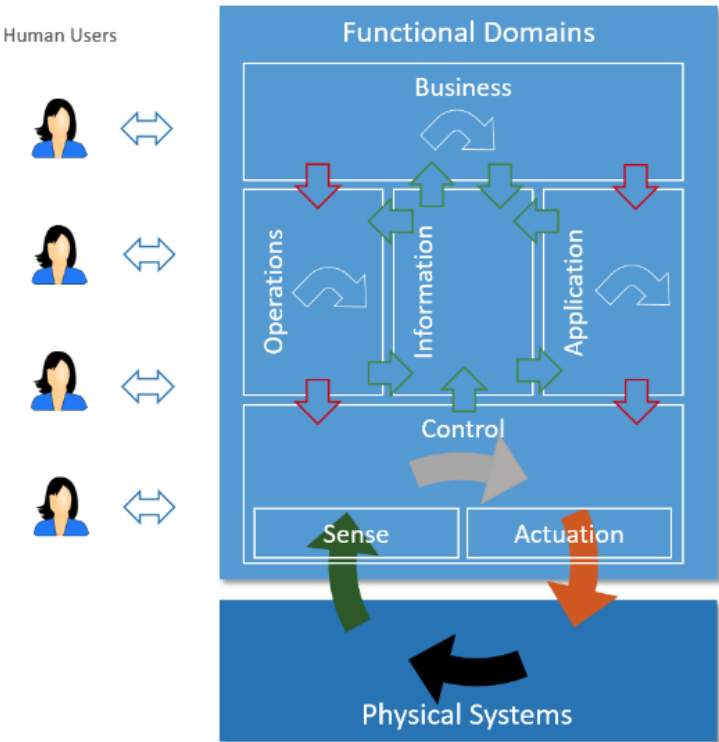


Figure 1.4: IIRA function domains and control flows [18]

architectural frameworks, and practical applications, with a particular emphasis on their indispensable role in meeting the dynamic challenges posed by CPS.

2.1 Self-adaptation

The concept of self-adaptation is a fundamental mechanism that enables systems to autonomously adjust and optimize their behavior based on changes in the environment or evolving requirements [122][121]. In a self-adaptive system, two integral parts collaborate to ensure a dynamic and responsive operation:

- **Managed System:** The managed system constitutes the primary interface with the environment and is responsible for addressing domain concerns. These concerns encompass the specific needs and requirements of users for whom the system is designed. The managed system continuously interacts with its surroundings, collecting data, processing information, and taking actions to fulfill its intended functionalities within the given domain.
- **Managing System:** Complementing the managed system is the managing system, featuring a feedback loop designed for interaction with the managed system and continuous monitoring of its environment. The managing system takes on the responsibility of addressing adaptation concerns, focusing on optimizing and adjusting the behavior of the managed system. This adaptation aims to respond effectively to changes in the environment or evolving user requirements, ensuring the system remains resilient and responsive.

Self-adaptation is akin to an intelligent, dynamic decision-making process where the managing system learns from the behavior of the managed system and adapts its strategies accordingly. This adaptive capability proves crucial in scenarios where environmental conditions are unpredictable or subject to frequent changes.

2.2 Taxonomy for self-adaptation

A taxonomy for self-adaptation permits the systematic organization and comprehension of the multifaceted aspects inherent in adaptive systems. This structured classification is crucial for navigating the complexities associated with self-adaptation. Different taxonomies for self-adaptation or adaptation have been developed over the years. As one of the first ones, Rohr et al. describe a classification scheme for self-adaptation research in 2006 [88]. In their work, they classify self-adaptation among the dimensions origin, activation, system layer, controller

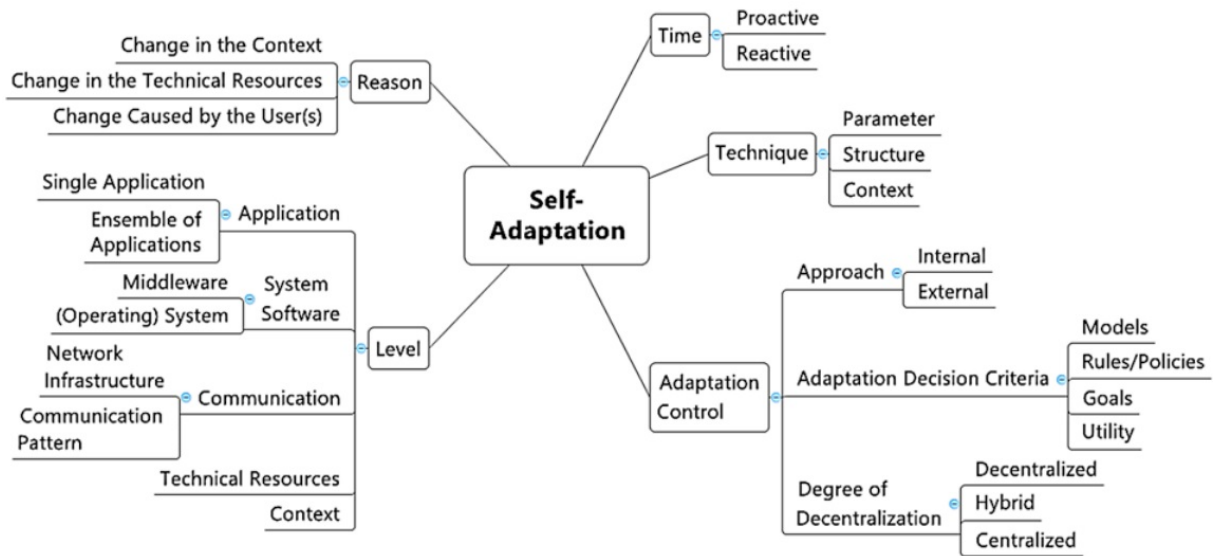


Figure 1.5: Adaptation taxonomy [60]

distribution, and operation. In [92], Salehie and Tahvildari introduce the 5W + 1H questions for eliciting adaptation requirements:

- When to adapt ?
- Why do we have to adapt ?
- Where do we have to implement change ?
- What kind of change is needed ?
- Who has to perform the adaptation ?
- How is the adaptation performed ?

In [60], authors further enrich the taxonomy by providing explicit answers to these questions. Table 1.1 succinctly illustrates the relation between the taxonomy dimensions and the adaptation requirements questions, serving as a guide for understanding and classifying the intricate dynamics of self-adaptation in systems. This taxonomy framework is designed to address the unique challenges presented by context-adaptive systems, particularly in the pervasive computing domain.

Figure 1.5 shows an overview of the proposed taxonomy. This taxonomy highlights key aspects of self-adaptation.

Table 1.1: Relation of the taxonomy dimensions and the 5W+1H questions

Question	Dimension of the Taxonomy
When?	Time (reactive vs. proactive)
Why?	Reason (context, technical resources, user)
Where?	Level (application, system software, communication, technical resources, context)
What?	Technique (parameter, structure, context)
Who?	N/A (nature of a SAS leads to an automatic type of adaptation)
How?	Adaptation control (approach, adaptation decision criteria, degree of decentralization)

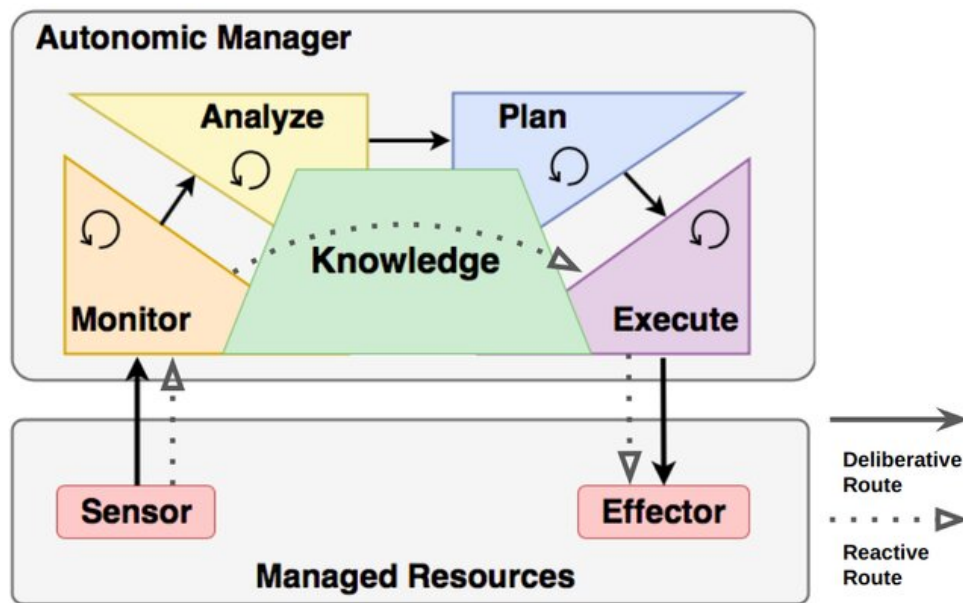


Figure 1.6: MAPE-K Architecture [87]

2.3 Adaptation process

A well recognized engineering approach to realize self-adaptation is by means of a feedback control loop called MAPE-K [52], [19] and conceived as a sequence of four computations Monitor-Analyze-Plan-Execute over a Knowledge base.

The adaptation loop within a self-adaptive system involves a sequence of interconnected processes designed to ensure a continuous and responsive evolution of the managed resources [24]. As shown in figure 1.6, this loop consists of distinct but interdependent phases: monitor, analyze, plan, and execute.

- **Monitor Process** The monitor process, as the initial phase of the adaptation loop, is responsible for collecting detailed information from managed resources. This information encompasses topology details and configuration property settings. The monitor process aggregates and filters this information until it detects a symptom that necessitates analysis.

- **Analyze Process** Upon identifying symptoms, the analyze process takes center stage, performing data analysis and reasoning. It leverages the symptoms provided by the monitor process and is influenced by stored knowledge data. This phase involves understanding the current state of the system and determining whether changes are required for optimal performance.
- **Plan Process** When changes are deemed necessary, the plan process comes into play. It structures actions required to achieve specific goals based on the analysis conducted. The plan process can take various forms, ranging from a single command to a complex workflow. It is responsible for creating or selecting a procedure to enact the desired alteration in the managed resources.
- **Execute Process** The final phase of the adaptation loop is the execute process, tasked with implementing changes in the behavior of the managed resources based on the actions recommended by the plan process. This phase ensures the seamless execution of the planned alterations, contributing to the system's adaptability.

All four processes within the adaptation loop share a knowledge base, encompassing data such as historical logs, symptoms, or policies. This knowledge base serves as a repository that may be updated by the processes depending on the outcomes of their respective tasks. The iterative nature of the adaptation loop allows the system to continually learn, analyze, and adapt, fostering a dynamic and responsive environment.

2.4 Structure of self-adaptive systems

A self-adaptive system is a dynamic entity capable of adjusting its behavior and structure in response to environmental changes, internal system dynamics, and defined goals [1][45]. The generic conceptual architecture of a self-adaptive system is illustrated in figure 1.7, delineating the roles and interactions between key components. As presented in section 2.1, self-adaptive system contains two main components which are : managed and managing subsystems besides the environment:

- **Managed Subsystem:** The managed subsystem abstracts the core application functionality of the system. It represents the part of the system that is subject to adaptation based on the perception of the environment and system goals.
- **Managing Subsystem:** The managing subsystem serves as an abstraction of the system's adaptation logic. This component identifies runtime changes and orchestrates adaptive

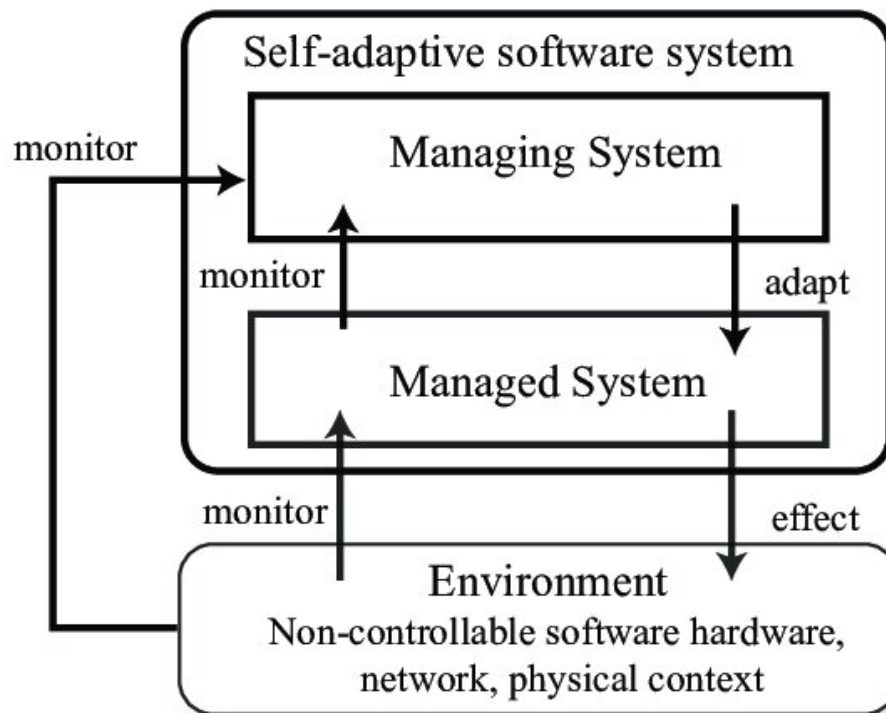


Figure 1.7: Generic conceptual architecture of a self-adaptive system [1]

actions on the managed subsystem to align with and maintain the overarching system goals.

- **Environment:** The environment refers to the external world with which the self-adaptive system interacts. Both the managed and managing subsystems operate within this environment, and changes in the environment can trigger adaptive responses.

Beyond the three core components, a network of crucial interfaces facilitates communication and exchange of information.

- **Monitor and Adapt Interfaces:** The managing subsystem employs the monitor and adapt interfaces to observe changes in the managed subsystem and enact adaptive actions, respectively. These interfaces facilitate the crucial functions of monitoring and adapting, allowing the system to stay attuned to its surroundings.
- **Monitor Interface (Managing Subsystem - Environment):** The monitor interface enables context-awareness by facilitating communication between the managing subsystem and the environment. It allows the managing subsystem to observe changes in the external world, providing valuable insights for adaptive decision-making.

- Adapt Interface (Managing Subsystem - Managed Subsystem): The adapt interface empowers the managing subsystem to perform adaptive actions on the managed subsystem. While the managing subsystem cannot directly affect the environment, it can influence the managed subsystem, potentially causing ripple effects in the broader context.

3 Conclusion

This chapter has provided a foundational exploration into the fundamental concepts of CPS. From delving into diverse definitions to dissecting key components and characteristics, we've sought to unravel the intricacies that define these systems at the intersection of the digital and physical realms.

As we navigate the dynamic landscape of CPS, the chapter underscores the inherent adaptability of these systems. Recognizing CPS as dynamic entities, we introduced the concept of self-adaptive systems, where autonomy and optimization in response to ever-changing environments take center stage.

Looking ahead to the next chapter, our focus will shift to the role of Machine Learning in ensuring adaptability within CPS. We will explore how ML algorithms contribute to the dynamic nature of these systems, enhancing their ability to autonomously adjust and optimize.

2. Machine Learning and Collaborative Machine Learning

This chapter delves into the pivotal role of Machine Learning (ML) and collaborative ML presented in Federated Machine Learning (FML) and Split Machine Learning (SML) in addressing the challenges posed by dynamic CPS. Beginning with an exploration of ML, we define its principles, types, and structured pipeline. Transitioning to FML, we highlight its relevance in distributed systems, emphasizing privacy considerations. We then explore SML, focusing on its ability to manage resource constraints in CPS by partitioning data and computation tasks.

1 Machine Learning

1.1 Definition of Machine Learning

In the literature, the term ML has been defined in various ways, reflecting the multidimensional nature of this evolving field. This section aims to provide an understanding of ML by presenting key definitions and fundamental concepts associated with the term.

- Definition 1: "Machine Learning is the science of getting computers to learn without being explicitly programmed" [94]. This definition underscores the autonomous learning capacity of ML systems, highlighting the departure from traditional programming paradigms. In essence, ML involves the creation of systems that can learn and adapt without explicit human intervention.
- Definition 2: "Machine Learning is the analysis of learning processes and the creation of systems that improve their performance on the basis of past experience" [83]. This definition emphasizes the analytical nature of ML, positioning it as a field that not only observes and analyzes learning processes but actively develops systems capable of enhancing performance through accumulated experience. It signifies the iterative and adaptive nature of ML systems in response to changing contexts.
- Definition 3 : According to Tom M. Mitchell's definition of ML: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E " [73][11]. A well-defined learning task is given by $\langle P, T, E \rangle$.

1.2 ML versus Traditional programming

While both traditional programming and machine learning (ML) involve creating code to perform tasks, they take fundamentally different approaches. In traditional programming, a programmer explicitly writes code with various conditionals and loops to handle different scenarios. The program follows these instructions to the letter, with no room for deviation or adaptation.

In contrast, ML algorithms learn the rules of the game by observing examples. They analyze large amounts of data to identify patterns and relationships, and then use these insights to make predictions or decisions on new data. Unlike traditional programs, ML algorithms can adapt and improve over time as they encounter more data.

As presented in figure 2.1 , automatically inferring the actions from data without manual intervention is the key differentiator between ML and traditional programming [23]. Also, ML algorithms can handle complex tasks that would be difficult or even impossible to program explicitly, such as recognizing faces in images or predicting customer behavior. Table 2.1 summarizing the key differences between ML and traditional programming:

Table 2.1: Comparison of ML and Traditional Programming

Feature	Traditional Programming	Machine Learning
Learning process	Explicitly programmed by humans	Learns from data by identifying patterns and relationships
Adaptability	Not adaptable to new data	Can adapt and improve over time with more data
Suitability for complex tasks	Less suitable for complex, unpredictable tasks	Well-suited for complex tasks that are difficult to program explicitly

1.3 Types of ML

ML involves showing a large volume of data to a machine so that it can learn and make predictions, find patterns, or classify data. The three machine learning types are supervised, unsupervised, and reinforcement learning [101].

1.3.1 Supervised learning

Supervised learning relies on datasets where each data point is paired with a corresponding target value or label. This labeling, the very essence of supervision, empowers the algorithm to uncover the underlying mapping function between the input variables and the desired output.

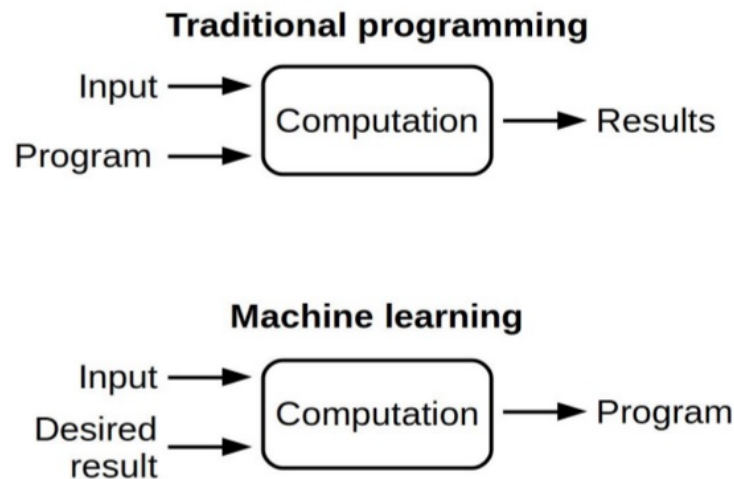


Figure 2.1: ML versus Traditional Programming [23]

Consequently, the algorithm learns to predict the appropriate output for unseen data, unlocking a vast array of applications [59].

While its effectiveness is undeniable, supervised learning bears a cost: the requirement of large-scale labeled data. Acquiring and labeling data can be expensive and time-consuming, particularly for tasks where data is scarce. Therefore, the suitability of supervised learning hinges on a delicate balance between the problem at hand and the availability of labeled data [91].

These approaches can be broadly divided into two main categories:

- **Classification:** In classification tasks, the output variable assumes one of a predetermined set of categories. For instance and like showed in figure 2.2, it may involve distinguishing between categories such as "cat" or "dog," or discerning sentiment as "positive" or "negative." The algorithm, informed by labeled training data, learns to generalize and make predictions on new, unseen data, assigning it to the appropriate category based on learned patterns.
- **Regression:** Another facet of supervised learning is Regression, where the output variable represents a real or continuous value. In this context, the algorithm aims to predict values that fall within a spectrum rather than distinct categories. Examples of regression tasks include predicting "price" based on various features, estimating "geographical location" from given parameters, or even predicting an individual's salary based on factors such as years of experience (see figure 2.3). For instance, a regression model could be trained on historical salary data to make predictions for future salary levels.

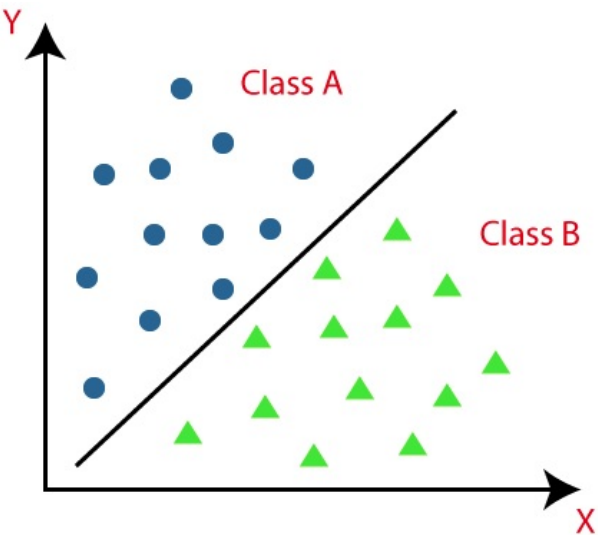


Figure 2.2: Classification [102]

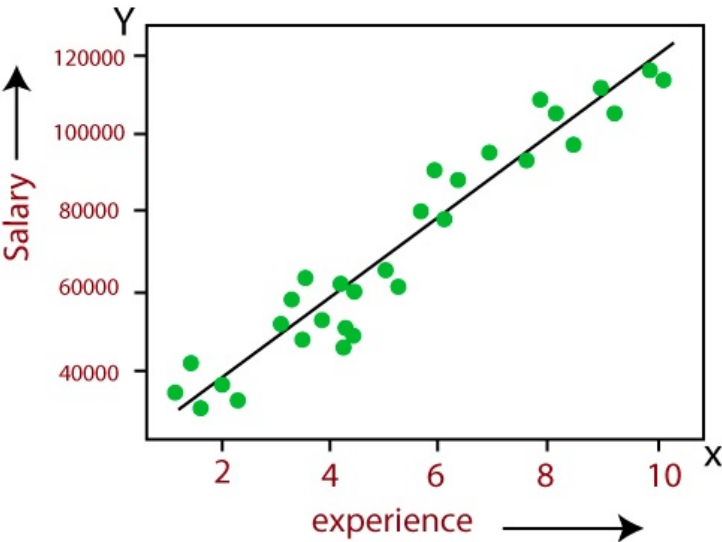


Figure 2.3: Regression [103]

1.3.2 Unsupervised learning

Unsupervised learning is a type of machine learning in which models are trained using unlabeled dataset and are allowed to act on that data without any supervision [104].

In these scenarios, algorithms within the unsupervised learning domain undertake the task of modeling the latent patterns inherent in the data. The primary objective is to unravel the intrinsic characteristics of the dataset without the explicit guidance provided by labeled output variables.

This learning approach is particularly valuable in situations where the inherent structure of the data needs to be uncovered, leading to a deeper understanding of its underlying patterns. Unsupervised learning algorithms operate on the premise of exploring and revealing the inherent relationships and structures within the data, allowing for valuable insights and knowledge discovery [91].

The unsupervised learning algorithm can be further categorized into three main types [109]:

- **Clustering:** Clustering stands out as a prominent type of unsupervised learning, focusing on grouping objects into clusters based on similarities. The fundamental premise is to form clusters where objects exhibit maximum similarity within a group and minimal or no similarities with objects from other clusters. Cluster analysis identifies commonalities between data objects and categorizes them based on the presence or absence of these commonalities. This approach facilitates the discovery of natural groupings within the dataset, aiding in the segmentation of data points with shared characteristics.
- **Association rules** An association rule represents a method guided by rules to uncover relationships among variables within a given dataset. This approach is commonly employed in market basket analysis, providing companies with insights into the intricate connections between various products. By comprehending customer consumption patterns, businesses can refine their cross-selling strategies and optimize recommendation engines. Illustrations of this practice are evident in features like Amazon's "Customers Who Bought This Item Also Bought," [46] showcasing the practical application of association rules in enhancing customer experience and driving sales.
- **Dimensionality Reduction:** While increased data typically leads to more precise outcomes, it can simultaneously affect the efficiency of machine learning algorithms, potentially leading to issues such as overfitting. Moreover, handling extensive datasets can pose challenges in terms of visualization. Dimensionality reduction comes into play when dealing with a dataset that exhibits an excessively high number of features or dimensions.

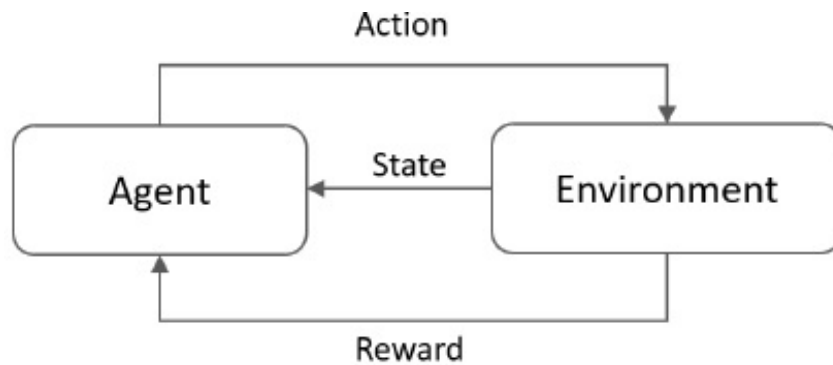


Figure 2.4: Overview of reinforcement learning

This technique aims to streamline the dataset by decreasing the number of data inputs to a manageable size, all the while endeavoring to retain the essential characteristics and integrity of the original dataset.

1.3.3 Reinforcement learning

Reinforcement learning takes center stage when the objective involves making a series of decisions leading to a final reward. In this paradigm, an artificial agent undergoes a learning process where it receives either rewards or penalties based on its actions. The overarching aim is to maximize the cumulative reward achieved over time. Instances where reinforcement learning excels include training agents to play computer games or execute robotics tasks with a defined end goal [34].

During the learning journey, the artificial agent dynamically adapts its decision-making strategies, driven by the continuous feedback loop of rewards and penalties. The agent strives to learn an optimal policy that guides its actions towards achieving the highest possible cumulative reward [91].

As presented in figure 2.4, reinforcement learning finds particular utility in scenarios where explicit programming of a solution is challenging or impractical. Instead, reinforcement learning enables agents to learn and evolve their behavior autonomously through interaction with their environment. The application domains range from training game-playing algorithms to developing robotic systems capable of performing intricate tasks with a focus on maximizing the overall reward.

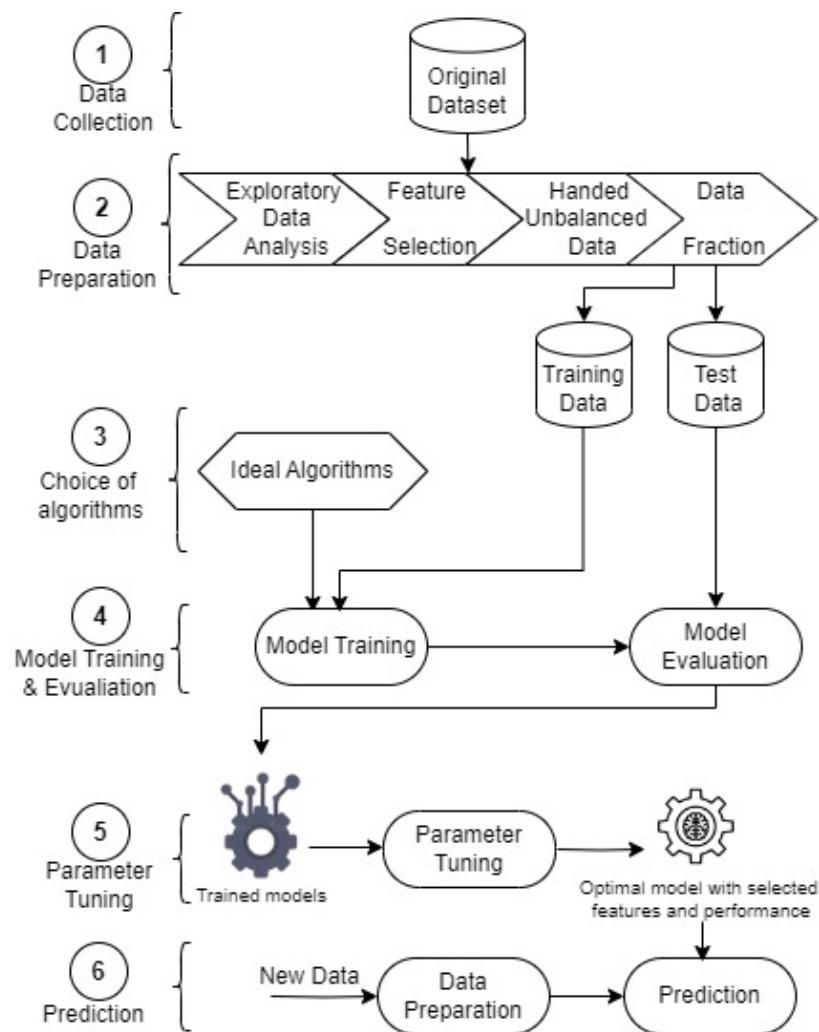


Figure 2.5: ML pipeline

1.4 Machine learning pipeline

A machine learning pipeline is a series of interconnected data processing and modeling steps designed to automate, standardize, and streamline the process of building, training, evaluating, and deploying machine learning models. By automating repetitive tasks and ensuring consistent execution, pipelines significantly increase efficiency, minimize errors, and ultimately lead to more robust and successful models [108]. Figure 2.6 depicts the key steps involved in the machine learning pipeline.

1.4.1 Data collection

The foundation of any successful ML endeavor lies in acquiring and curating reliable, relevant data. This step encompasses identifying appropriate data sources, employing ethical and

efficient data acquisition methods, and ensuring data quality through careful cleaning and pre-processing. Think of it as meticulously gathering the stones and mortar essential for building the ML edifice.

1.4.2 Data Preparation

Raw data rarely arrives ready for analysis. In this step, the researcher meticulously prepares the data for consumption by the chosen ML algorithm. Tasks like Exploratory Data Analysis, Feature Selection, Handling Unbalanced Data, and data fraction occur here, effectively shaping the raw materials into bricks suitable for the construction process.

- **Exploratory Data Analysis** : is a pivotal substep in the data preparation phase, playing a crucial role in analyzing and understanding the key characteristics of the dataset. This step is essential for uncovering patterns, identifying anomalies, and extracting valuable insights that contribute to informed decision-making throughout the ML pipeline. Exploratory Data Analysis involves employing a variety of techniques to analyze and understand the key characteristics of the dataset such as : Heatmap, Pair Plots, Box-and-Whisker Plots, and Histograms with Kernel Density Estimation.
- **Feature selection** : It stands as a pivotal substep within the data preparation phase, wielding significant influence over the efficacy of the ML model. This process is geared towards identifying and retaining the most pertinent variables, a task crucial for optimizing model performance, minimizing computational complexity, and mitigating the risk of overfitting. The most techniques used in feature selection are : Correlation Analysis, Feature Importance Scores, Recursive Feature Elimination, and Univariate Feature Selection.
- **Handling unbalanced data** : In scenarios where the distribution of classes is uneven, with one class significantly outnumbering the others, models may exhibit biases towards the majority class. This can lead to suboptimal performance, particularly for minority classes. Addressing this imbalance involves employing techniques such as oversampling the minority class, undersampling the majority class, or utilizing advanced methods like synthetic data generation. Striking a balance in class representation ensures that the machine learning model is trained to recognize patterns across all classes, promoting robust predictions in real-world scenarios.
- **Data fraction** : This substep involves partitioning the dataset into distinct subsets for training, validation, and testing purposes. This process is essential for evaluating the

model’s performance on unseen data and ensuring its ability to generalize effectively. The typical split involves allocating a majority of data for training to allow the model to learn patterns, a smaller portion for validation to fine-tune hyperparameters, and a separate set for testing to assess real-world predictive capabilities. A well-considered data fraction strategy contributes to model robustness and provides a reliable assessment of its effectiveness in diverse scenarios.

1.4.3 Choice of ML Algorithms

In the pivotal step of choosing ML algorithms, the selection process hinges on aligning the algorithms with the specific nature of the problem at hand. Key considerations include the problem type, whether it involves classification or regression, the inherent characteristics of the dataset, and the desired performance criteria. The diversity of ML algorithms offers a spectrum of approaches, each tailored to address distinct problem types and nuances. Table 2.2 provides a concise overview of some widely used ML algorithms categorized by the type of learning they employ.

Type of Learning	Algorithm
Supervised Learning	Linear Regression Decision Trees Random Forest Support Vector Machines k-Nearest Neighbors Neural Networks
Unsupervised Learning	K-Means Clustering Hierarchical Clustering Principal Component Analysis Gaussian Mixture Models
Reinforcement Learning	Q-Learning Deep Q Networks Policy Gradient Methods

Table 2.2: Commonly Used Machine Learning Algorithms

1.4.4 Model training and evaluation

Once the data is fully prepared and the model is well chosen,the next critical step in the pipeline is model training and evaluation. In this step, the chosen algorithm is fed with the training

dataset, allowing it to learn the patterns and relationships within the data. The trained model is then evaluated using a separate validation dataset to assess its performance and generalization capabilities.

- **Model Training** : It involves iteratively presenting the algorithm with input data and adjusting its internal parameters to minimize the difference between predicted outputs and actual outcomes. The training process continues until the model achieves satisfactory performance on the training dataset.
- **Model Evaluation** : Following training, the model is evaluated using a distinct validation dataset that it has not seen before. Evaluation metrics, such as accuracy, precision, recall, and F1 score for classification tasks, or mean squared error for regression tasks, provide quantitative insights into the model's performance.

1.4.5 Parameter tuning

Parameter tuning, also known as hyperparameter optimization. It aimed at fine-tuning the internal settings of the chosen algorithm to achieve optimal performance. While model training focuses on learning from the data, parameter tuning involves adjusting external configurations, such as learning rates or regularization parameters, to enhance the model's generalization and predictive accuracy.

They are many techniques and strategies used in this step such as :

- **Grid Search**: Systematically search through a predefined hyperparameter grid to identify the combination that yields the best model performance.
- **Random Search**: Randomly sample hyperparameter configurations, allowing for efficient exploration of the hyperparameter space.
- **Bayesian Optimization**: Utilize probabilistic models to predict the performance of different hyperparameter settings, guiding the search toward promising configurations.

1.4.6 Deploying the Trained Model for Real-world Predictions

The final stage of the machine learning pipeline involves deploying the trained model for real-world predictions. After the model has been trained, evaluated, and its parameters fine-tuned, it is ready to be utilized on previously unseen data to make accurate predictions and generate valuable insights.

While the transformative power of ML is undeniable, enabling systems to learn and make predictions from data across various domains, the conventional centralized training approach raises concerns regarding data privacy, security, and the need to transmit sensitive information to a central server. This is where Federated Machine Learning (FML) emerges as a solution, emphasizing a distributed learning paradigm.

FML extends the principles of ML into a collaborative, decentralized environment. In the subsequent section, we will delve into the intricacies of Federated Machine Learning, exploring its foundational concepts, methodologies, and real-world applications.

2 Federated Machine Learning

2.1 Definition and Relevance of Federated Machine Learning

The concept of Federated Machine Learning (FML) is proposed by Google recently [71, 57, 58]. Also known as collaborative learning, represents a decentralized paradigm for training machine learning models. In contrast to traditional approaches, FML eliminates the need for exchanging raw data from client devices to global servers. Instead, the model is trained locally on edge devices, preserving the privacy of sensitive information [99]. The main characteristics of Federated Learning are :

- **Decentralized Model Training:** FML distributes the training process across individual devices or servers, allowing local training without sharing raw data.
- **Enhanced Data Privacy:** By keeping data on edge devices, FML mitigates privacy concerns associated with centralized data repositories.
- **Shared Model Formation:** The final model is collaboratively formed by aggregating local updates, ensuring a collective representation of knowledge.

Numerous studies [68, 50, 67, 119] underscore the undeniable benefits and relevance of FML across diverse domains. The growing body of evidence highlights FML as a pivotal solution, offering innovative approaches to challenges prevalent in centralized machine learning models. Several key aspects illuminate the significance of FML, supported by empirical findings and real-world applications:

- **Privacy Preservation:** FML addresses growing concerns related to data privacy by allowing model training on local devices without the need to transmit raw data. This is

particularly critical in contexts where preserving the confidentiality of sensitive information is paramount.

- **Edge Computing Integration:** The integration of FML with edge computing heralds a paradigm shift, enabling on-device model training. This decentralized approach aligns seamlessly with the principles of edge computing, reducing latency and dependency on centralized servers.
- **Scalability in Edge Environments:** In environments with a multitude of edge devices, FML offers a scalable solution. It optimizes resource utilization by distributing the learning process, making it well-suited for scenarios where traditional centralized approaches might encounter scalability challenges.
- **Collaborative Knowledge Formation:** The collaborative nature of FML, where the final model is formed by aggregating local updates, promotes a collective representation of knowledge. This collaborative knowledge formation contributes to robust and adaptable models.
- **Mitigation of Communication Overhead:** By training models locally on edge devices, FML mitigates communication overhead associated with transmitting large volumes of raw data to centralized servers. This is particularly advantageous in resource-constrained or bandwidth-limited environments.

2.2 Principle of Operation of FML

FML operates through a series of well-defined steps that ensure model training without compromising data privacy. As presented in figure 2.6, the fundamental steps involved in the FML process are:

- **Step 1: Sending Encrypted Gradients (Client to Server) :** The FML process initiates with individual clients, each possessing local datasets. In Step 1, clients compute gradients based on their local data and encrypt these gradients before transmitting them to the central server. This cryptographic safeguard ensures that raw data remains secure during transmission.
- **Step 2: Secure Aggregation at Server :** Upon receiving the encrypted gradients from multiple clients, the central server performs secure aggregation in Step 2. This process involves aggregating the encrypted gradients from different clients while maintaining the

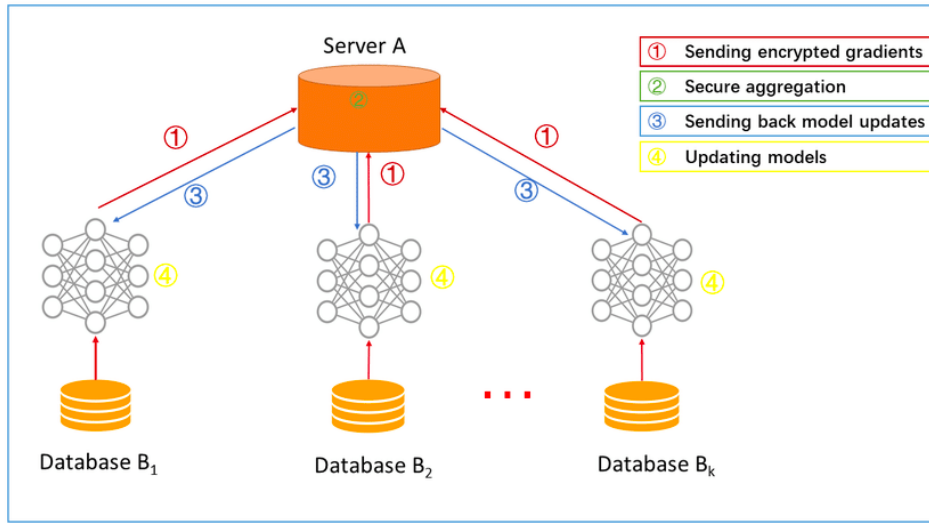


Figure 2.6: Basic principle of FML [128]

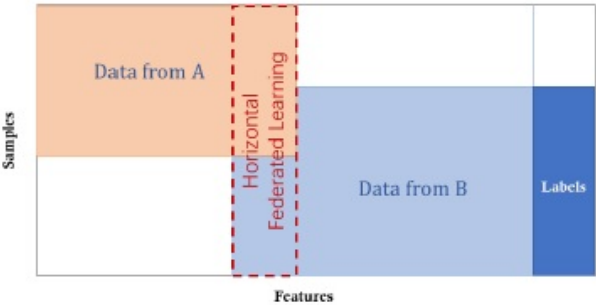
confidentiality of individual contributions. Secure aggregation methods, such as cryptographic techniques, play a pivotal role in preserving data privacy.

- **Step 3: Sending Back Model Updates (Server to Clients) :** Following secure aggregation, the central server sends back the aggregated model updates to the respective clients in Step 3. These updates represent the collective knowledge gleaned from the entire federated learning process. The use of encryption ensures the confidentiality of the model updates during transmission.
- **Step 4: Updating Models (Client Side) :** In the final step, clients receive the aggregated model updates and update their local models accordingly. This collaborative updating process ensures that each client benefits from the collective insights gained from the entire federated learning network. By repeating these steps iteratively, FML facilitates continuous model improvement across distributed devices.

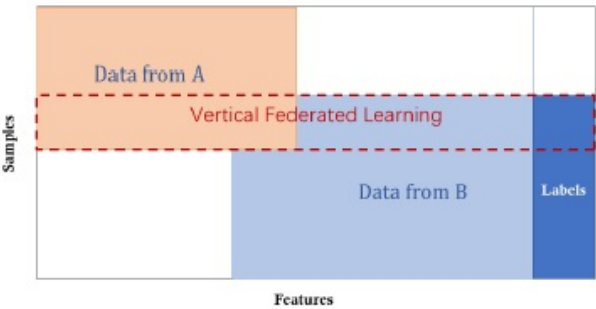
2.2.1 Types of FML

FML gracefully adapts to diverse data landscapes through distinct approaches, each tailored to specific data distribution patterns and privacy requirements. Figure 2.7 shows the various types of federated learning [128].

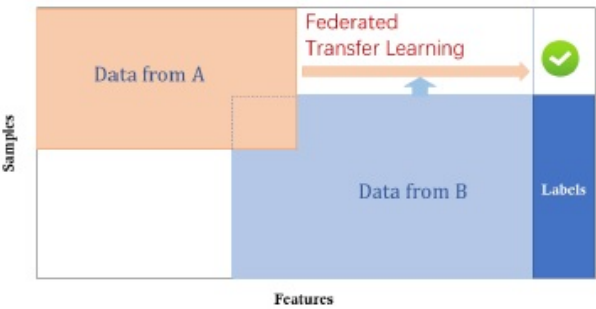
- **Horizontal Federated Learning (HFL) :** Horizontal Federated Learning is the preferred approach when the user characteristics of two datasets exhibit considerable conflict, but



(a) Horizontal Federated Learning



(b) Vertical Federated Learning



(c) Federated Transfer Learning

Figure 2.7: Types of FML [128]

the users themselves do not. In HFL, datasets are horizontally divided based on user dimensions. The goal is to extract data where user characteristics are similar but not identical for training. Alternatively, data in different rows share the same data properties, aligned by user features. HFL accommodates scenarios where there are distinct user bases within different locations or entities providing similar services.

- **Vertical Federated Learning (VFL) :** Vertical Federated Learning, or feature-based learning, is employed when two datasets share the same sample ID space but differ in the feature space. In VFL, multiple parties have access to the same data collection, but each party only has access to a distinct subset of features. This approach is particularly relevant in industries such as healthcare, finance, and banking, where privacy concerns are paramount. VFL focuses on preserving privacy by allowing collaborative learning without exposing sensitive feature details to all parties involved.
- **Federated Transfer Learning :** Federated Transfer Learning represents a unique form of federated learning, distinguished by differences in the feature space of two datasets. Unlike horizontal and vertical federations, federated transfer learning involves datasets acquired from various organizations with similar traits. The feature overlap between these datasets is minimal due to differences in the businesses they serve. Successful transfer learning in this context requires leveraging knowledge from other sections of the target domain. This approach is akin to teaching a system to use one product and applying the acquired knowledge to another, facilitating adaptability across diverse datasets.

2.3 Main Algorithms for Model Aggregation

In FML, model aggregation is a pivotal process that involves combining local models from multiple clients to generate a cohesive global model. Several algorithms have been proposed to enhance the effectiveness and efficiency of model aggregation in the context of federated learning. Here, we explore some of the main algorithms employed for model aggregation in FML [86]:

- **FedAvg :** FedAvg [71] stands as one of the earliest and widely used methods in Federated Learning (FL). In this approach, a group of clients is randomly selected during each round of training for aggregation. The parameters of each client are weighted and averaged to produce a global model, where the weight is determined by the proportion of the client's data volume. Additionally, FedAvg allows for the incorporation of more computation to each client by iterating the local update multiple times before the averaging step.

- **FedProx** : FedProx [65] serves as an enhancement to the FedAvg method, specifically addressing the challenge of local optimization inherent in Stochastic Gradient Descent (SGD)-based approaches. FedProx introduces a proximal term into the objective function to regulate the influence of local models, ensuring convergence guarantees. This addition helps prevent suboptimal convergence or model divergence by pulling the local model towards the global model through the constraints of the proximal term.
- **FedNova** : Proposed by Wang et al. [120], the FedNova algorithm enhances the model aggregation phase of FedAvg to address non-Independently and Identically Distributed (non-IID) data scenarios. It introduces a technique to normalize and scale local updates from each client based on its local iteration number before updating the global model, contributing to improved model convergence.
- **Scaffold** : To tackle the client-drift problem resulting from data heterogeneity, the Scaffold algorithm [51] suggests employing control variable technology, specifically variance reduction, in the local update process. By incorporating client and server control variables, Scaffold leverages the disparity between local and global control variables to rectify gradient updates during local training.
- **MOON** : MOON [64] aims to minimize the discrepancy between local models and the global model by incorporating a model contrastive loss as a regularization term. The model contrastive loss serves as a metric for quantifying the dissimilarity between the local model and the global model, enhancing model convergence in federated settings.
- **Per-FedAvg** : Per-FedAvg [31] combines model-agnostic meta-learning (MAML) with Federated Learning to produce personalized local models. MAML first trains the initial parameters of the model and then uses a small amount of data to perform one or more gradient descents on the new task, enabling the model to achieve good performance. In Per-FedAvg, the local model is updated using one-step gradient descent of the loss function with the objective of finding an initial model for each client, allowing for personalized learning experiences within the federated framework.

3 Split Machine Learning

3.1 Definition and Relevance of SML

SML is an advanced collaborative machine learning technique that partitions a neural network into distinct segments, allowing multiple parties to collaboratively train a model without sharing raw data [55]. SML is used for training Neural Networks among multiple data sources while mitigating the need to directly share raw labeled data with collaboration parties. The advantages of SML are multifold [118]:

- It allows users to train ML models without sharing their raw data with a server running part of the model, thus preserving user privacy,
- It protects both the client and the server from revealing their parts of the model, and
- It reduces the client's computational overhead by not running the entire model (i.e., utilizing a smaller number of layers).

By splitting the network, SML addresses key challenges in privacy, data security, and resource constraints, making it highly relevant for dynamic Cyber-Physical Systems (CPS). SML ensures that sensitive data remains within the local environment while enabling effective model training through distributed computation.

3.2 Split Learning Configurations

SML encompasses various configurations tailored to specific needs and constraints. Three common configurations include [118]:

- **Simple Vanilla Configuration:** In this configuration, each client trains a partial deep network up to a designated layer, known as the cut layer. The outputs from this layer are forwarded to a central server for further model training without exposing raw data. The server aggregates gradients and sends them back to clients for local backpropagation. This setup minimizes data exposure while facilitating collaborative model training.
- **U-shaped Configurations without Label Sharing:** These configurations mitigate the need for label sharing among clients while preserving data privacy. The network architecture is wrapped around at the end layers of the server's network, and outputs are sent back to clients. This approach maintains privacy by ensuring that sensitive labels are not shared.

- **Vertically Partitioned Data:** This configuration enables collaborative model training across multiple institutions with different data modalities without sharing raw data. Each institution trains a partial model up to its designated cut layer, and outputs are concatenated and forwarded to a central server for further training. This setup supports collaboration while maintaining the privacy of each institution's data.

4 Conclusion

This chapter has highlighted the crucial role of ML and collaborative ML in addressing the challenges of dynamic CPS. We've explored the principles of ML, including its types, and discussed the decentralized paradigm of collaborative ML, emphasizing its ability to handle sensitive data through collaborative, on-device training. By synthesizing these insights, we've underscored the transformative impact of ML and collaborative ML on CPS adaptability and resilience, particularly when dealing with sensitive data in industrial systems, where protecting proprietary information and intellectual property is critical. In the next chapter, we will present a comprehensive examination of existing literature and research efforts in utilizing ML to enhance adaptability within CPS. Through this review, we aim to identify key trends, challenges, and emerging methodologies in the field, in order to set the stage for introducing our novel contribution.

3. Machine Learning for Enhancing Adaptability in CPS : A Literature Review

In this chapter, we present an in-depth exploration of ML techniques aimed at enhancing adaptability in CPS. First, we present the main challenges impeding seamless adaptability in CPS, including dynamic environments, uncertainty, evolving requirements, fault tolerance, interoperability, and security. Next, we present the specific benefits ML brings to enhance adaptability within CPS, recognizing its transformative impact on resilience. Finally, we conduct a critical comparative analysis of diverse approaches in recent ML-based research for adaptive CPS. By highlighting their advantages and drawbacks, we were able to clarify our thesis's contribution compared to existing work and what new insights it brings to the field of using ML to improve adaptation in CPS.

1 Adaptability Challenges in CPS

The adaptability of CPS refers to their ability to dynamically adjust and evolve in response to changing conditions, emerging requirements, and unforeseen events. This section delves into the multifaceted challenges that impede the seamless adaptability of CPS, encompassing aspects such as dynamic environments, uncertainty, evolving requirements, fault tolerance, interoperability, and security.

- **Dynamic Environment and Uncertainty:** CPS are characterized by dynamics and uncertainty of operations and behavior due to exceptions in environmental conditions, system configurations, and user interactions. CPS are present in a variety of safety/mission critical domains [25][4]. Given the pervasiveness of CPS and their criticality to the daily functioning of society, it is vital for such systems to operate in a reliable manner. However, since they generally function in an inherently complex and unpredictable physical environment, a major difficulty with these systems is that they must be designed and operated in the presence of uncertainty. According to Zhang et al. [130], uncertainty is defined as the absence of certainty, indicating a lack of knowledge regarding the timing and nature of inputs, the current state of a system, anticipated future outcomes, and other pertinent factors. Uncertainty arises from various sources, including unpredictable environmental conditions such as weather fluctuations, dynamic changes in system configurations, and the diverse and sometimes unpredictable behaviors of users interacting with the system. As a result, CPS must be able to adapt to these dynamic and uncer-

tain conditions to maintain functionality and performance, making adaptability a critical requirement in CPS design and operation.

- **Evolving Requirements and Scalability:** The adaptability of CPS is further challenged by evolving requirements driven by technological advancements, user preferences, and regulatory changes [105][40]. Scalability becomes crucial as CPS systems expand in complexity, data volume, and network size over time. According to [40], scalability refers to the ability of a system to keep functioning well even in case of change in its size/increased workload, and take full advantage of it. Ensuring scalability while maintaining performance, reliability, and efficiency presents a formidable challenge, necessitating scalable architectures, distributed computing paradigms, and adaptive resource allocation strategies.
- **Fault Tolerance and Resilience:** Fault tolerance and resilience are paramount in safety-critical CPS applications [8][84], where system failures can have catastrophic consequences. Challenges abound in detecting, diagnosing, and recovering from faults in dynamic and heterogeneous CPS environments. Achieving fault tolerance requires the development of robust monitoring systems, fault detection algorithms, and automated recovery mechanisms capable of mitigating disruptions and maintaining system integrity.
- **Interoperability and Heterogeneity:** Interoperability challenges arise from the integration of diverse hardware and software components in CPS, including sensors, actuators, communication protocols, and legacy systems. Heterogeneity complicates system integration, data exchange, and interoperability standards across different CPS deployments, domains, and industries [69], [49]. Addressing interoperability and heterogeneity requires standardized interfaces, middleware frameworks, and protocol translation mechanisms to facilitate seamless communication and collaboration among disparate components.
- **Security and Privacy:**

Security and privacy in CPS are paramount due to their complex, interconnected components and middleware, which render them vulnerable to cyber-attacks while striving to maintain operational efficiency and performance integrity [6] [29]. Balancing security measures with system performance, usability, and accessibility is a delicate task, necessitating robust authentication mechanisms, encryption protocols, and access control policies to safeguard CPS assets and information.
- **Resource Constraints:** CPS often operate in environments with limited computational

resources, memory, and energy supply. These resource constraints pose significant challenges for the deployment and operation of sophisticated ML models and adaptive algorithms within CPS [13]. Ensuring efficient resource management and optimizing the performance of CPS under such constraints necessitate the development of lightweight algorithms, efficient data processing techniques, and adaptive resource allocation strategies.

2 ML Advantages for Enhanced CPS Adaptability

Driven by the need for adaptable CPS, the integration of ML has become a prominent focus within relevant research and development communities. The inherent benefits of ML, such as its ability to process vast amounts of data, learn from experience, and make real-time predictions, make it ideal for ensuring rapid adaptation in these dynamic systems [11][21]. Numerous studies have explored the utilization of ML techniques to develop flexible and adaptive CPS capable of responding to changing environmental conditions. ML empowers CPS with three key strengths for enhanced adaptability [39, 3]:

- **Real-time Learning:** Unlike static models, ML algorithms can continuously learn from new data, updating their understanding and adjusting their behavior on the fly.
- **Predictive Foresight:** ML algorithms can analyze data patterns to anticipate upcoming events and changing conditions. This foresight allows CPS to proactively adapt, preempting disruptions and maintaining optimal performance.
- **Autonomous Decision-Making:** ML algorithms can analyze complex situations and make informed decisions independently, reducing reliance on centralized control. This decentralized intelligence makes CPS more resilient to disruptions and adaptable to unforeseen circumstances.

3 Review of ML Approaches for Adaptability in CPS

ML techniques integrated into CPS have been instrumental in addressing challenges like self-adaptation, security, and data privacy, enabling CPS to autonomously adapt to dynamic environments, enhance security measures, and safeguard sensitive data [56, 81]. This section surveys existing research on utilizing ML within CPS environments, highlighting its benefits in facilitating self-adaptation, strengthening security mechanisms, and ensuring data privacy,

as demonstrated in Table 3.1. The table categorizes these works based on several key criteria, including the specific ML method employed, the learning process (centralized vs. distributed), and considerations for privacy, security, and communication overhead in CPS deployments. Following this overview, we transition to a subsection "Discussion," where we critically analyze the limitations of existing approaches and set the stage for introducing our novel contribution.

3.1 Related Work

Table 3.1: Comparative overview of ML applications in CPS

Papers	Goal	Method ML	Learning Process	Privacy Considerations	Security Considerations	Comm. Over-head
Diaz et al. [28]	Machine component knowledge discovery	Clustering algorithms based on real data	Centralized	No	No	High
Wu et al. [124]	Detecting Cyber Physical attacks	Image classification using ML algorithms	Centralized	No	No	High
Babouche et al. [16]	Self-adaptive and scalable prediction/detection mechanism	Multiple ML algorithms with statistical tests	Centralized	No	No	High
Grass et al. [37]	Anomaly detection for predictive maintenance	Historical sensor data and clustering algorithms	Centralized	No	No	High

Continued on next page

Table 3.1 – Table continuation 3.1

Papers	Goal	Method ML	Learning Process	Privacy Considerations	Security Considerations	Comm. Over-head
Rouzbahani et al. [75]	Anomaly detection in CPS using ML	Various ML techniques	Centralized	No	Yes	High
Sargolzaei et al. [95]	Fault detection in VCPSs using neural network-based technique	Neural Network	Centralized	Yes	Yes	High
Omprakash et al. [79]	Enhancing CPS Security in Water Distribution Systems	Fuzzy Logic-based Feature Selection	Centralized	No	Yes	High
Alshboul et al. [9]	Predictive maintenance using ML	Feature selection	Centralized	No	Yes	High
Li et al. [65]	FML for Intrusion Detection in CPS	Deep Learning (CNN and GRU)	Distributed	Yes	Yes	Low
Zhang et al. [129]	FML-based Edge Computing Platform for CPS	Federated Learning	Distributed	Yes	Yes	Low
Xu et al. [127]	FML for secure incentive mechanism in CPS	Federated Learning	Distributed	Yes	Yes	Low
Guo et al. [41]	Deep FML for secure microservices in CPS	Deep federated learning	Distributed	Yes	Yes	Low

In [28], authors propose a complete real-world CPS implementation cycle, ranging from machine data acquisition to processing and interpretation. In fact, they propose a CPS for ma-

chine component knowledge discovery based on clustering algorithms using real data from a machining process. More precisely, they propose the development of a component behavior multidimensional pattern using machine learning clustering techniques. Clustering algorithms aim to partition a centralized dataset into clusters, interpreted as behavior patterns. Some properties of these clusters like the mean can give insight into their interpretation.

In [124], physical data machine learning approaches are developed and integrated for detecting Cyber Physical attacks in Cyber Manufacturing system. Authors developed two examples with simulation and experimentation to test and demonstrate the physical data machine learning security approach. In the 3D printing process example, they used vision as physical data source for machine learning. Three different machine learning algorithms are implemented with image classification. The anomaly detection method returned the highest accuracy of 96.1% in detecting a malicious defect in printing process.

Authors in [16] provide a self-adaptive and scalable prediction/detection mechanism for CPS. They propose a framework called AAPF-CPS, which combines several machine learning algorithms with statistical tests. With multiple classification algorithms, AAPF-CPS analyzes CPS network logs simultaneously and in real-time. Friedman's test is also used to rank each classifier for each context in AAPF-CPS. In [37], authors present an approach for anomaly detection regarding predictive maintenance in an CPS. In particular, they are focusing on historical sensor data from a real reflow oven that is used for soldering surface mount electronic components to printed circuit boards. The sensor data comprises information about the heat and the power consumption of individual fans inside a reflow oven. The data set contains time annotated sensor measurements in combination with additional process information over a period of more than seven years.

Authors in [47] propose a two-level ensemble attack detection and attribution framework specifically designed for CPS, particularly in industrial control systems. Their approach utilizes a decision tree combined with a novel deep representation-learning model at the first level for anomaly detection in imbalanced industrial control systems environments. This is followed by an ensemble deep neural network at the second level for attack attribution. Evaluation using real-world data from gas pipelines and water treatment systems demonstrates their model's effectiveness in outperforming other approaches with similar computational complexity.

In [79], the authors address the growing vulnerability of CPS, particularly in critical infrastructures such as water distribution systems (WDS), to cyber-attacks. In their proposed approach, the authors leverage fog computing and employ fuzzy logic-based methods, specifically the Intuitionistic Fuzzy System, for feature selection. Furthermore, a voting classifier is applied to the selected features to detect anomalies within the WDS. The effectiveness of their

system is demonstrated through testing on the Water Distribution dataset, and the results are compared with state-of-the-art methods. This work contributes valuable insights into enhancing the security of CPS, especially in critical infrastructure scenarios.

In [61], the authors propose an intrusion detection system tailored for an IEEE 1815.1-based power system integrated with CPS. With the integration of CPS into power systems, new communication requirements emerge, necessitating a mapping between existing distributed network protocol networks and IEC 61850 networks, as defined by the IEEE 1815.1 standard. Addressing the evolving threat landscape of advanced CPS cyberattacks, the study conducts a comprehensive analysis of the IEEE 1815.1-based power system network. It identifies an optimal application method for an intrusion detection system and introduces a bidirectional recurrent neural network-based anomaly detection system tailored for the nuances of an IEEE 1815.1-based network.

In [43], the authors focus on bolstering the security of CPS, with a specific emphasis on direct current microgrids. Addressing the rising concern of cyber threats in diverse applications like renewable energy and power distribution, they propose an innovative AI-driven method. This method, employing time-series analysis and a tailored nonlinear auto-regressive exogenous model (NARX) neural network, aims at timely detection and identification of cyber-attacks within dc microgrids. Notably, the strategy accommodates the challenges posed by false data injection attacks targeting voltage and current sensors. Through meticulous phases, including a cyber-attack-free data collection phase and subsequent online deployment of trained NARX networks, the proposed method showcases effectiveness.

Author in [53] propose an approach to address the challenge of privacy preservation in CPS, which generate vast amounts of big data from the integration of physical and digital entities. This paper explores the application of Independent Component Analysis (ICA) as a technique for protecting sensitive information from unauthorized access. The ICA method is employed to transform raw CPS data into a new configuration while retaining its data utility. The effectiveness of this mechanism is assessed using a power CPS dataset, demonstrating superior privacy protection compared to four other privacy-preservation techniques. Furthermore, the data utility is evaluated through the utilization of three machine learning algorithms (Naive Bayes, Decision Tree, and Support Vector Machine) to gauge their effectiveness in identifying normal and attack patterns before and after transformation. The results highlight the enhanced privacy protection and utility preservation achieved through the proposed ICA-based approach.

Authors in [72] propose a novel contribution to the dynamic adaptation of service-oriented systems within the context of the Internet of Things (IoT). The work addresses the increasing complexity of modern software systems and the demand for flexibility and self-adaptation in

service-based IoT applications. The paper underscores the importance of predicting quality-of-service values, especially in scenarios where users interact with a limited subset of services, resulting in missing quality-of-service data. In response, the authors introduce the Auto-NF framework, which consists of three steps. The framework begins by proposing an extended similarity computation method, employing Euclidean distance to evaluate user similarity and identify comparable neighbors. The second step involves the formation of clusters of similar neighbors and subsequent partitioning of the initial matrix to alleviate data sparsity challenges. The third step introduces a straightforward neural network autoencoder, designed to learn deep features and select optimal latent factors to mitigate overfitting.

In [3], authors discuss the integration of ML techniques within CPS, focusing on Industry 4.0. CPS combines computation and physical processes and ML optimizes CPS functionalities such as domain adaptation, system fine-tuning, and vulnerability detection. ML enables CPS to learn from large-scale data, enhancing security and privacy in industrial settings. The paper highlights ML applications in predictive maintenance, quality assurance, and optimization of manufacturing processes and supply chains, emphasizing ML's transformative impact on CPS in Industry 4.0.

In their paper, Wickramasinghe et al. [123] propose a methodology for explainable unsupervised ML tailored for CPS. They address the challenge of the 'black-box' nature of complex ML models by introducing explainable unsupervised ML models, particularly suitable for safety-critical CPS applications. Their approach utilizes Self-Organizing Maps (SOMs) based clustering methodology to generate both global and local explanations, enhancing the interpretability of ML models within CPS contexts. Through feature perturbation techniques, they evaluate the fidelity of the generated explanations, demonstrating the effectiveness of their proposed method in identifying the most important features responsible for decision-making processes in CPS.

In [75], M. Rouzbahani et al. delve into anomaly detection in CPS using ML methods. The authors underscore the complexity of CPS, which integrate cyber components into the physical world, leading to diverse tasks and close interactions. With the proliferation of smart features and communication tools in CPS, new challenges related to security and privacy have arisen, particularly in systems like the smart grid. Anomaly detection emerges as a crucial strategy for enhancing CPS security, yet comparing various detection methods poses challenges due to their diversity. To address this, the chapter focuses on ML-based anomaly detection methods, highlighting their effectiveness through a case study on False Data Injection (FDI) attacks. Through their exploration, the authors contribute insights into the application of ML techniques for anomaly detection in CPS, emphasizing their potential in mitigating security threats.

In [95], authors propose a novel approach for detecting faults in vehicular cyber-physical

systems (VCPSs). They highlight the potential of VCPSs to enhance transportation safety, mobility, and sustainability through wireless communication. However, they also address the vulnerability of cloud-oriented architectures to cyber attacks, emphasizing the risks to safety, privacy, and property. Their proposed solution involves a neural network-based technique aimed at identifying and tracking fault data injection attacks in real time within a platoon of connected vehicles. They develop a decision support system to mitigate the probability and severity of potential accidents resulting from these attacks, ultimately enhancing system reliability, robustness, and safety.

In [9], authors propose an empirical exploration of predictive maintenance in concrete manufacturing. By harnessing machine learning techniques, they aim to enhance equipment reliability in construction project management. The study identifies key features, such as 24-hour mean voltage, crucial for predicting machinery failure within the concrete manufacturing framework. Insights gleaned from this empirical investigation underscore the significance of integrating machine learning methodologies into construction project management practices. This integration not only improves the accuracy of maintenance forecasts but also reinforces equipment dependability, ensuring optimal efficacy and benefit in the concrete manufacturing paradigm.

Authors in [65] propose DeepFed, a federated deep learning scheme for detecting cyber threats in industrial CPS. The scheme utilizes a convolutional neural network (CNN) and a gated recurrent unit (GRU) to develop an intrusion detection model tailored for industrial CPS. Furthermore, a federated learning framework is introduced, allowing multiple industrial CPS to collaboratively build an intrusion detection model while preserving data privacy. A Paillier cryptosystem-based secure communication protocol is also devised to ensure the security and privacy of model parameters during the training process. Experimental results on real industrial CPS datasets demonstrate the effectiveness of DeepFed in detecting various cyber threats and its superiority over existing schemes.

Authors in [129] propose a novel FML-based Edge Computing platform named "FengHuoLun" specifically designed for CPS. This platform aims to address the challenge of ensuring trustworthy smart services in Edge Computing environments by leveraging Federated Learning. With FengHuoLun, smart services can be implemented with machine learning models trained in a trusted FML framework, ensuring the trustworthiness of CPS behaviors through testing and monitoring.

In [127], authors introduce a novel approach employing multiagent federated reinforcement learning to devise a secure incentive mechanism in intelligent CPS. While federated learning addresses data privacy concerns in CPS, ensuring efficient incentive mechanisms remains crucial. Deep reinforcement learning is explored as a solution for long-term incentivization amidst

dynamic environments. However, the heterogeneity of CPS devices poses a challenge, affecting the convergence rate of existing single-agent reinforcement learning. The proposed multiagent learning-based mechanism addresses this issue by approximating stationarity in federated learning with heterogeneous CPS. The approach formulates the secure communication and data resource allocation problem as a Stackelberg game and models it as a partially observable Markov decision process to handle device heterogeneity. A multiagent federated reinforcement learning algorithm is devised to efficiently learn allocation policies, mitigating policy evaluation variances caused by device interactions without compromising privacy.

In [41], the authors propose a deep federated learning framework to enhance secure points of interest (POI) microservices in CPS. This framework aims to improve data security by isolating the cloud center from accessing user data on edge nodes. Through interactive training between the cloud center and edge nodes, reliable deep-learning-based models are pre-trained on edge nodes, and parameter updating is coordinated via federated learning. The proposed approach is evaluated using real-world POI-related datasets, demonstrating optimal scheduling performance and practical utility.

3.2 Discussion

While the works in the table 3.1 exhibit significant efforts in enhancing security and resilience in CPS, some common limitations can be identified. Most prominently, the majority of the approaches rely on centralized ML solutions, concentrating on global data processing and model training. This centralized paradigm presents challenges related to data privacy, security, and potential single points of failure. Additionally, centralized data storage raises scalability challenges as CPS systems grow in complexity. Moreover, continuous data transmission to a central server for model training and updates creates significant communication overhead, straining network resources, increasing vulnerability to interception, and potentially compromising real-time performance.

In contrast to the centralized approaches discussed earlier, decentralized methodologies, such as FML, emerge as promising alternatives that address some of the drawbacks associated with centralized models. Decentralized architectures, like FML, distribute the model training process across local devices or servers, eliminating the need for continuous data transmission to a central repository. This not only mitigates communication overhead but also safeguards against potential security and privacy concerns associated with centralized data storage. The decentralized nature of FML aligns with the distributed architecture of CPS, offering scalability as systems expand in complexity. Moreover, local training on diverse datasets allows for

adaptation to specific environmental conditions, promoting a more resilient and adaptable CPS framework.

However, all the works presented in table 3.1 and that are based on FML, like Li et al. [65], Zhang et al. [129], Xu et al. [127], and Guo et al. [41], do not respect resource constraints. These approaches often assume the availability of sufficient computational power and bandwidth, which may not be practical in resource-constrained environments typically encountered in CPS.

To address these limitations, SML has been proposed as a viable solution. SML divides the model training process between resource-constrained edge devices and more powerful central servers. This approach reduces the computational load on local devices while still benefiting from collaborative learning. By using SML, we propose a novel architecture in this thesis that is specifically designed to operate efficiently within the resource constraints of CPS. This architecture ensures enhanced scalability, security, and data privacy, while also optimizing resource utilization and maintaining high performance in real-time applications.

4 Conclusion

This chapter shed light on the multifaceted challenges hindering seamless adaptability in CPS. We explored how ML offers a compelling solution by empowering CPS with real-time learning, predictive foresight, and autonomous decision-making capabilities. We reviewed existing research on utilizing ML for self-adaptation, security, and data privacy within CPS environments. Finally, we identified limitations associated with current centralized approaches and highlighted the potential of FML and SML as a promising avenue for future research.

In the next chapter, we pivot towards our contribution. Our research, presented in the following chapter, seeks to provide novel insights and methodologies for leveraging ML in CPS adaptation. This work introduces three significant contributions that collectively enhance the landscape of adaptive CPS:

- Firstly, our thesis puts forth a comprehensive meta-model tailored for adaptive CPS. This meta-model encapsulates the essential characteristics and behaviors inherent in CPS, offering a conceptual foundation for understanding and modeling their intricate interactions between physical and digital components.
- Secondly, we unveil a centralized ML-based architecture designed specifically for CPS. This architecture not only encompasses critical functionalities such as data management,

resource optimization, fault prediction, and proactive maintenance but also integrates advanced technologies like Machine Learning and Cloud/Edge Computing.

- Thirdly, we present a groundbreaking Privacy-Enhanced Architecture empowered by FML and SML Expanding on the second contribution, this architecture addresses scalability challenges and introduces substantial improvements in terms of security and data privacy.

4. Proposed Framework for Adaptive Cyber-Physical Systems

Cyber-physical systems (CPS) have revolutionized various industries, seamlessly integrating physical and computational elements to enhance efficiency, reliability, and safety. However, the increasing complexity and dynamic nature of modern CPS necessitate adaptive architectures capable of responding to changing conditions and unforeseen events.

This chapter introduces a comprehensive framework for adaptive CPS. The framework consists of a meta-model, a centralized ML architecture, and enhancements to this architecture using both FML and SML. The meta-model provides a foundation for understanding and modeling CPS, while the centralized ML architecture leverages ML to enhance their adaptability and efficiency. However, the centralized architecture faces limitations in data privacy, scalability, and communication overhead. The FML-based architecture addresses these limitations by distributing ML computations across multiple nodes, while the SML-based architecture further tackles resource constraints in CPS, offering an efficient solution for environments with limited computational resources.

1 Meta-Model for Adaptive CPS

Figure 4.1 depicts the meta-model we have developed for addressing self-adaptive CPS. The proposed meta-model is based on advanced work about CPS and IoT [25] [17] and those related to designing and managing of self-adaptive systems [90]. Our model is composed of two parts: the first part includes all the generic elements of a CPS, whereas the second one holds the basic elements implementing the self-adaptation system. A CPS object is the composition of a virtual entity and a physical entity that is associated with, in order to highlight the fact that these two concepts belong together. The both present a state that may be perceived and changed during system running. Formally, we define a CPS as a non-empty set of intelligent components as follow:

$$CPS = \sum_{i=1}^n (comp_i) \quad (4.1)$$

In fact, a component *comp* of a CPS is the intersection between what is considered physical (PhyC) and what is virtual (VrtC). It is no longer enough to understand, develop, manage and maintain physical and virtual components separately. We will therefore note:

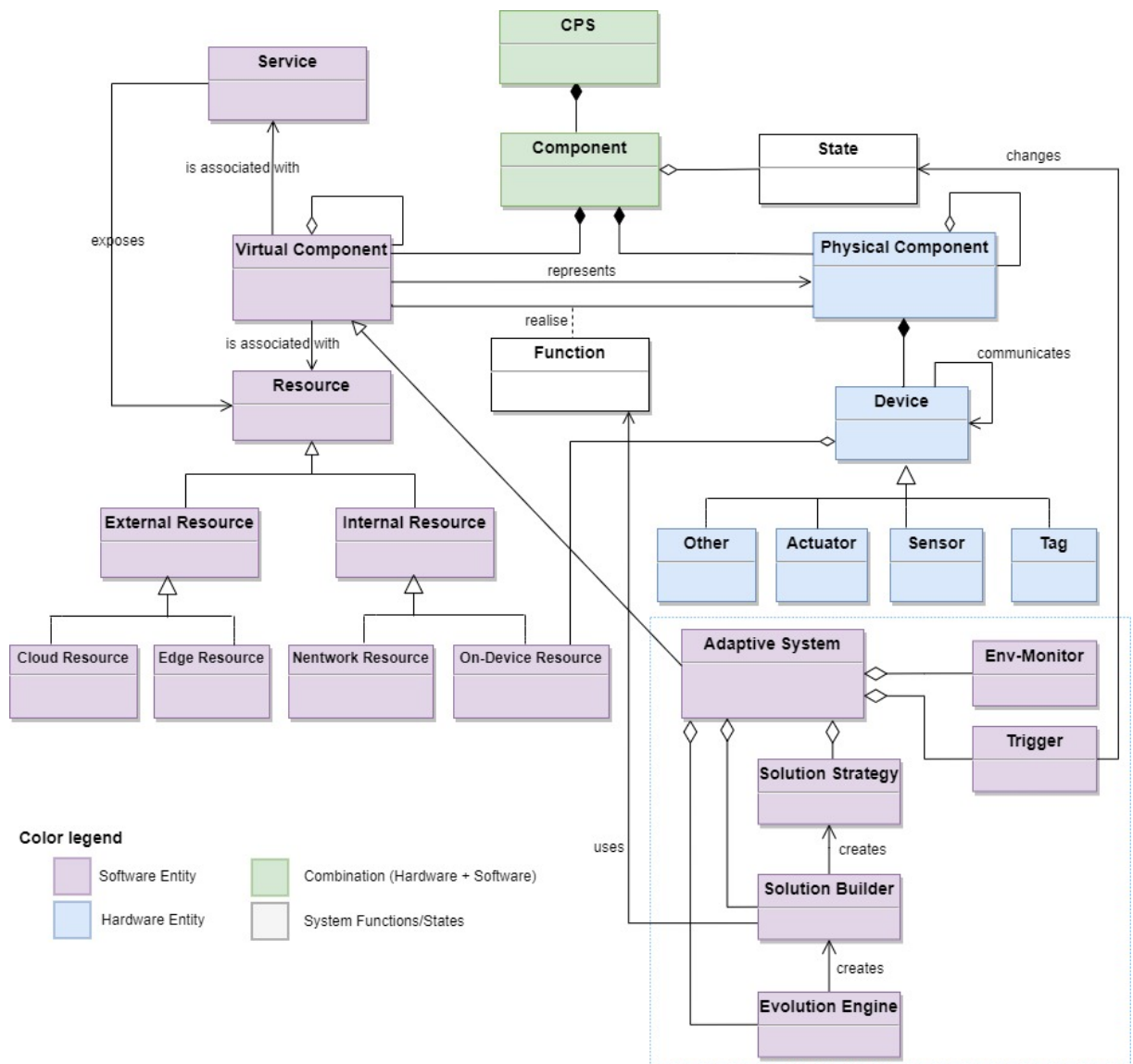


Figure 4.1: A meta-model for CPS

$$comp = PhyC \cap VrtC \quad (4.2)$$

Where:

$$PhyC = \sum_{i=1}^m (P_i) \quad (4.3)$$

and

$$VrtC = \sum_{i=1}^k (V_i) \quad (4.4)$$

Equation 4.2 defines a component (*comp*) of a CPS as the intersection between the set of physical entities (*PhyC*) and the set of virtual entities (*VrtC*), where *comp* represents the shared functionalities between the physical and virtual aspects of the system. *PhyC* and *VrtC* are further elaborated in Equations 4.3 and 4.4, respectively. *PhyC* is defined as the sum of physical components P_i with i ranging from 1 to m , encompassing entities such as sensors, actuators, tags, and other physical devices. Similarly, *VrtC* is expressed as the sum of virtual components V_i with i ranging from 1 to k , capturing various virtual entities in the CPS. Together, these equations establish the composition of a CPS component by combining its physical and virtual elements.

The result of the intersection between *PhyC* and *VrtC* gives the set of functions F performed by the CPS, where:

$$\forall P_i \in PhyC, \exists V_j \in VrtC, \exists f_k \in F : V_j \wedge P_i \Rightarrow f_k \quad (4.5)$$

Formula 4.5 expresses the functional relationship within a CPS by stating that for every physical component (P_i) in the set of physical entities (*PhyC*), there exists a virtual component (V_j) in the set of virtual entities (*VrtC*), and there exists a function (f_k) in the set of functions (F). The logical condition $V_j \wedge P_i \Rightarrow f_k$ signifies that the combination of a specific virtual entity V_j and a corresponding physical entity P_i implies the execution of a particular function f_k . In simpler terms, this formula formalizes the notion that the interaction between physical and virtual components in a CPS results in the performance of specific functions, contributing to the system's overall functionality.

Physicals Components (*PhyC*) are the recognizable part of the physical environment that interests the user to achieve his goal. They can be any physical device such as: robot arm, cars, storage elements or supply chain item, computers, electronics, ect. Like presented in formula

4.3, in our CPS model, PhyC is a set of physical devices:

$$P_i \in \{Sensor, Actuator, Tag, Other\} \quad (4.6)$$

A virtual entity (VrtC), representing the physical entity in the digital domain, maintains a one-to-one association with its physical counterpart. This digital representation serves as a key linchpin within our meta-model, encapsulating not only the essential features of the physical components but also acting as the focal point for enabling dynamic self-awareness, self-learning, and self-reconfiguring capabilities in CPS. Beyond mirroring the physical components, the virtual entity integrates a sophisticated layer of software, encompassing resources, services, and mechanisms dedicated to facilitating seamless self-adaptation. This holistic approach ensures that the virtual entity not only captures the state of the physical entity but also enhances it in the digital realm, contributing to the overall intelligence and resilience of the CPS.

Resources are software components that provide data or used in the operation of physical entities. In our model, we identify two types of resources: Internal and external resource. Concerning the first type, there is a distinction between resources on the device and network resources. As the name suggests, on-device resources are hosted on the devices. It is the software deployed on the device (sensor and actuator) associated with the physical entity.

On the other hand, network resources are locally deployed (in the LAN of the same organization), include executable code for accessing, processing, and storing sensor information, as well as the code for controlling actuators. External resources are the resources that are available somewhere in the Cloud/Edge, for example, a back-end or a Cloud-based database. As we mentioned above, our meta-model includes all the elements of self-adaptive systems which are the key enablers of smart environments such as CPSs. In such environments, the system is composed of the necessary elements to know the environment and to act on it. Thus, the system has Triggers (see Fig. 4.1) for changing the state of the environment. It has, also, perceptors (Environment Monitor) for obtaining the environment current state. In a CPS, adaptation is the higher level of a smart system that is able to inspect itself, learning from experience and self-modifying its specification. Consequently, in our meta-model we have the Evolution Engine that is qualified to know the model of the system at run-time, and to change or influence the solution strategy built at design time. In this case, the system is able to revise its run-time model, thus producing a new solution strategy. The Evolution Engine is able to set up a Solution Builder in which its role is to provide a completely new solution strategy by using a repository of established functionalities.

2 Centralized ML-Based Architecture for CPS

CPS are distributed and loosely coupled to cyber systems and physical systems that are monitored/controlled by user-defined semantic laws [62]. It is important to define the architecture of this paradigm as well as its components and their functionalities besides the used technology. The general objective of a CPS is to collect multiple data from various sources such as the data received from the sensors installed in the different production machines. Thereafter, convert them into information in the cyber world in order to process, to understand them, and then turn them into appropriate actions in the physical world. Therefore, our architecture [15] contains the main software modules that will ensure CPS functionalities such as: resource/data management, process planning, and the different modules which convert machines to be self-aware, self-learning and self-reconfiguring. As shown in Fig. 4.2, the proposed structure consists of the following layers:

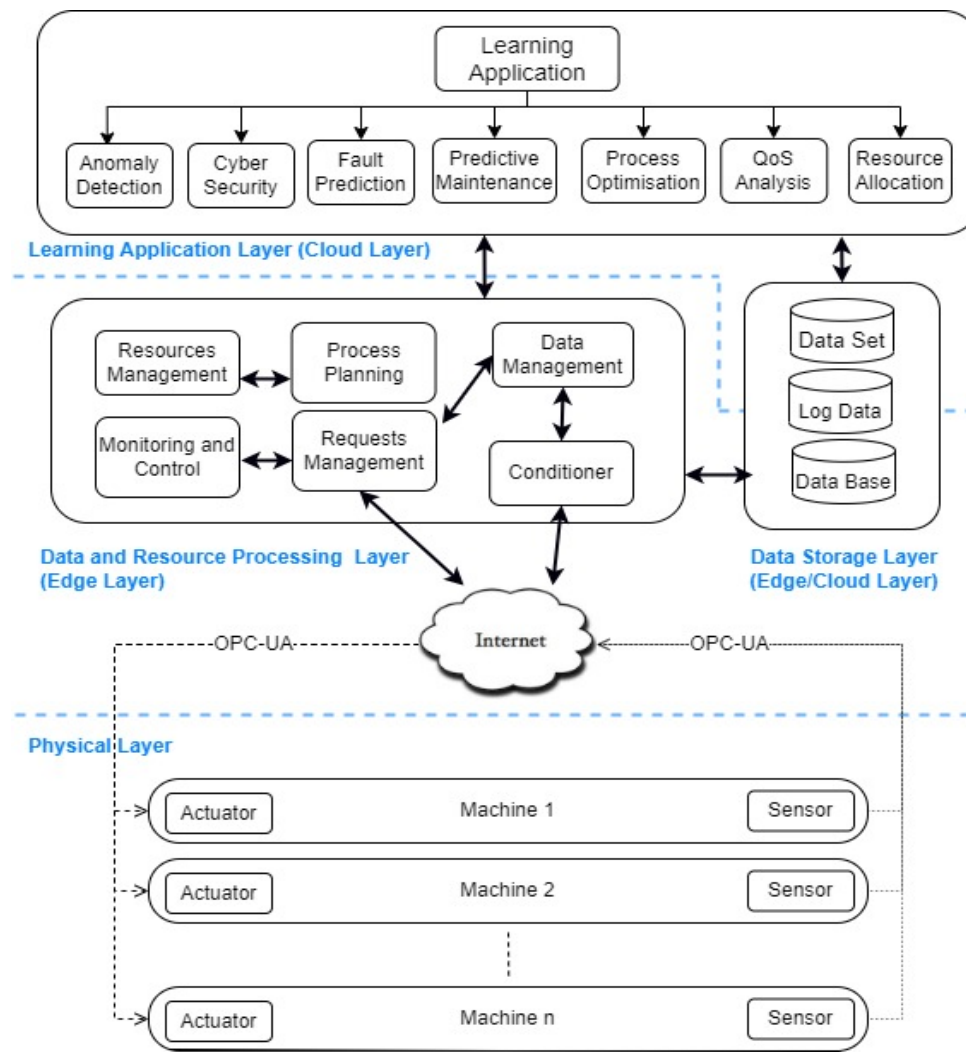


Figure 4.2: The proposed architecture

2.1 Physical layer

It represents the basic local assembly of machines connected to the Internet via the OPC-UA (OPen Connectivity – Unified Architecture) protocol to ensure communication standardization between different units. This layer consists mainly of sensors and actuators:

- Sensors are used to collect signals from machines and then transform them.
- Whereas, the actuators take electrical signals and combine them with a source of energy to create a physical movement.

2.2 Data/Resource Processing Layer

This layer collects data from various sources, and then it interacts directly with the data producer at the physical layer and the data storage in the Edge/Cloud. In addition, It manages requests and responses between the different layers of our architecture. The Data/Resource processing layer contains the following modules:

- **Conditioner:** its role is to record and aggregate values delivered by the sensor to restore information on the state of the system. These values can be used by the data management unit.
- **Data Management:** we preprocess data recovered from conditioner to eliminate collisions and clean the data of any redundancy (collection + aggregation). During data collection, data is acquired from devices at the physical layer, and it is sent to the aggregation component to be summarized into different pieces. Finally the data is sent to the data storage layer.
- **Resource Management:** A resource is defined as anything needed to perform a task. The resource management module ensures the discovery and arrangement of source data by storing the locations of database fragments for updating and querying purposes. This module aims to guarantee the notion of Real-time Queries which is essential for efficient and intelligent monitoring.
- **Planning Process:** its role is to plan, schedule, and allocate resources to optimize efficiency. In CPS, Process planning is an advanced computer-aided planning process system that collaborates with every part of the Learning Application layer (explained below). It enables the automatic generation of a process plan based on machine learning techniques. These latter help to define the optimal process plan by evaluating real-time data on the local utilization of available machines and tools.
- **Monitoring and Control:** This module is responsible for controlling and monitoring the system in real time. It supports processing optimization.
- **Requests management:** it manages the requests sent by physical layer. It is considered as an intelligent router which is able to decide whether the request is urgent or not. If it is urgent, it will be sent directly to the Monitoring and Control module. In this case, the request will be treated and transferred to the corresponding database, otherwise it is the normal case where the data will be, at first, treated by data management and then sent to data storage layer.

2.3 Data Storage Layer

This layer contains the main storage elements such as databases, log files (providing a full history of events over time) and data sets (required by the ML layer). This Data is cleansed, transformed, and prepared by a definite structure and thus provides opportunities for the system to use data as required by the Business. We note here that the data storage layer is distributed between the Edge and the Cloud, and this depends on the nature of the data and the time required to process it. According to our architecture, after receiving data from the Conditioner module, it sends it to the data management module for providing real time services. If we need real time services, data will sent to the Edge layer. Otherwise these data are considered as historical data and sent to the Cloud layer. The datasets that will be used by the ML are in the Cloud layer. This is because it needs an important capacity of storage. In addition, ML steps take time to process (for example the training step).

2.4 Learning Application Layer

A CPS is required to carry out a mission with minimum supervision. In fact, a CPS plans to be flexible systems that can self-aware, self-learning and self-reconfigure. Real-time analytics based on Machine Learning (ML) techniques provide the required functionality. The learning application layer is one of the important layers of our architecture. It solutions to incoming problems based on information that are locally processed and stored, as well as other bases such as the Cloud. Thereby, the system can keep pace with changes in the environment and machinery and make appropriate decisions. To this end, we use ML models, which allow us to perform a variety of tasks in CPS: such as anomaly detection, fault prediction, predictive maintenance, process optimization, intelligent resource allocation, etc. Machine learning models can be trained to allow the correct execution of the processes. They remap how it might handle the error. For example, after a device failure appears it will perform fault recovery and reconfiguration. The ML models can determine the probability factors of a fault. Therefore, it is possible to predict a breakdown and machine maintenance that will avoid production's interruption and as a result increase the company's productivity.

To demonstrate the viability and effectiveness of this layer, we have implemented it in a practical context. This application aligns with the realm of : Fault Prediction Using Supervised and Unsupervised Learning Algorithms [15]. In this real-world scenario, our system leveraged the power of supervised and unsupervised learning algorithms to predict failures in a dynamic environment such as CPS. The main goal of the Fault Prediction module is to predict whether

a device is exposed to failures or not. It allows, also, to identify the failure type. Therefore, its architecture will make use of supervised and unsupervised learning algorithms. First, we use classification techniques for separating data into two classes: faulty and correct data. Next, we apply the clustering method to the faulty data in order to group them into many clusters depending on fault type. We will delve further into the implementation and experimentation of our architecture in the following chapter.

3 Limitations of our Centralized ML-Based Architecture

Our centralized ML-based architecture presents a robust foundation for enhancing the adaptability of CPS. However, it also faces some challenges, such as data security, privacy concerns, scalability, and communication overhead, which need transmitting large volumes of data to a central server.

3.1 Data Security and Privacy Concerns

The effectiveness of our centralized architecture, relying on data aggregation from diverse devices for the learning process, introduces significant challenges in terms of data privacy. These limitations can be summarized as follows:

- **Centralized Data Collection:** The architecture's reliance on centralized data collection poses privacy concerns as sensitive information is aggregated in a single location, making it susceptible to potential breaches. Additionally, the nature of data generated by the components of the CPS is typically private, and these systems do not readily disclose their data, which makes the stage of collecting data from various systems impossible in some cases.
- **Privacy Vulnerabilities during Transmission:** During the transmission of data from various devices to the central server, privacy vulnerabilities may arise. Ensuring secure communication becomes paramount to prevent unauthorized access or data interception.

3.2 Scalability Concerns

As our architecture relies on a centralized system for processing and learning, the increasing volume of data from diverse devices may lead to performance limitations. Scaling the system to handle larger datasets becomes a critical consideration. Additionally, the centralized approach places a heavy load on computational resources, particularly as the number of connected devices

grows. Ensuring that the system can efficiently scale to accommodate increased computational demands is essential for sustained performance.

3.3 Communication Overhead

In a centralized architecture, transmitting data from various devices to a central server may result in a significant volume of data. The sheer quantity of information being sent can impose a strain on network bandwidth and resources. Moreover, transmitting large volumes of data over a network can introduce latency, impacting the speed at which data travels between devices and the central server. This latency may become a critical concern, especially in real-time or time-sensitive applications.

4 Towards a Privacy-Enhanced Architecture Empowered by Collaborative ML

In this section, we propose an advanced privacy-enhanced architecture using collaborative ML to address the limitations identified in our previous system. To overcome the inherent privacy challenges of centralized approaches, we integrate two distinct forms of Collaborative ML: Federated Machine Learning (FML) and Split Machine Learning (SML). This entails deploying intelligent algorithms and models at the edge of a network to facilitate local data processing and decision-making. Through the utilization of collaborative ML, data from diverse edge devices can collaboratively train machine learning models without the need for centralized data collection. This approach ensures enhanced data privacy and reduced latency.

This section provides an overview of the proposed collaborative ML-based architecture, focusing on the components of the architecture and how they function together. We will present the additions made to our previous architecture in terms of collaborative ML, specifically FML and SML.

4.1 The Proposed FML-Based Architecture for Enhancing Data Privacy in CPS

As shown in Fig. 4.3, the FML-based architecture consists of three main entities: local collectors, Edge aggregators and a Cloud layer.

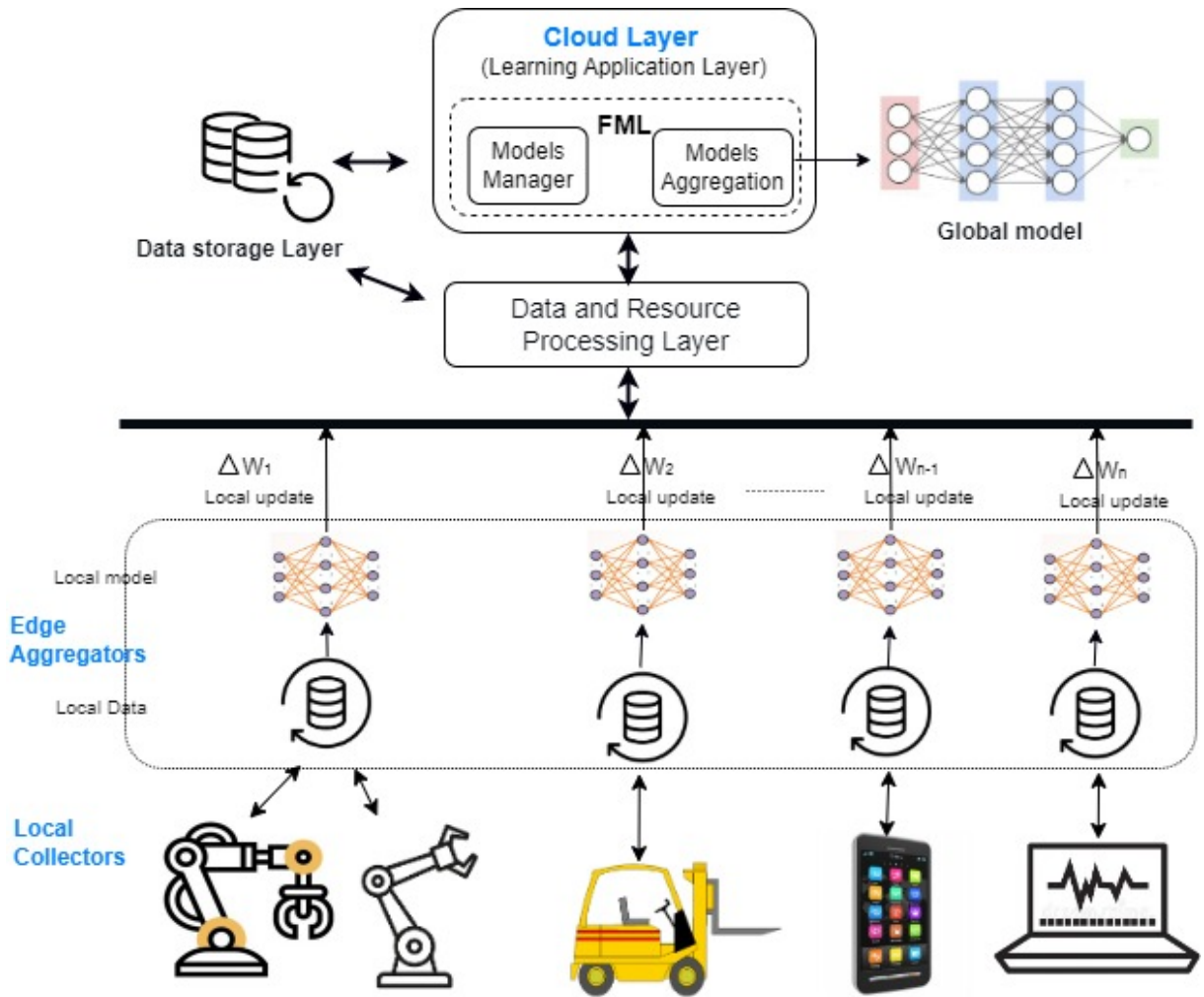


Figure 4.3: The proposed FML-based architecture

4.1.1 Architecture Components

Local Collectors:

Local Collectors are responsible for collecting and managing data from the local devices or nodes within a particular group or federation. They can encompass a wide range of devices, including industrial equipment, individual smartphones, IoT devices, sensors, and other endpoints. The Local Collectors ensure that data remains secure and private, often by aggregating or summarizing the data before its sent to the aggregator. The primary focus of Local Collectors is to prepare and send relevant information to the aggregator, respecting privacy and security constraints.

Edge Aggregators:

Edge Aggregators are responsible for aggregating the locally computed model updates from devices or nodes within their own federation. Each device trains its model locally using its own data, and then these local models are sent to the Edge Aggregator. It combines the models in some way, often by averaging the model parameters, to generate a global model update. This global model update is then shared with other federated groups or used to update the global model maintained by the Cloud layer. Edge aggregators facilitate collaborative learning across devices while ensuring that sensitive data remains on the local devices

Cloud Layer:

It plays a crucial role in the FML framework. Its main goal is to facilitate the aggregation of knowledge that originates from various local aggregators. Due to privacy considerations, our architecture shares only the learned data that is coordinated by the Cloud layer, while the private or sensitive data are stayed at the local aggregators. Due to potential variations in data types and machine learning models across local aggregators, the Models Manager module in our architecture assumes the role of overseeing and harmonizing these diverse models, ensuring effective management and coordination. The updating of the global model is achieved through a synchronization process that involves both the global model and the local aggregators. This synchronization is achieved by the Models Aggregation module, which manages the coordination between these components.

4.1.2 Mode of Operation

In our architecture, the goal is to collaboratively train a global model using data from N clients, each having their own local dataset denoted as D_i where: $1 \leq i \leq n$. This dataset consists of a set of K_i labeled pairs :

$$D_i = (X_i^{k_i}, Y_i^{k_i})_{k=1}^{k_i} \quad (4.7)$$

Where $X_i^{k_i}$ is referred to the training input, and $Y_i^{k_i}$ is the corresponding label.

In our architecture, the process revolves around collaborative training between Edge (Client side) and a Cloud layer (Server side). The goal is to update a global model by utilizing the local models trained on each client's local data. Algorithm 3 captures the behaviour of each client in our architecture.

The client-side process begins with the initialization of a local model (M_{local}) using the initial global model (M_0). This local model is provided by the Cloud layer and it serves as the starting point for the client's training process (line 1). Subsequently, the local model is trained

Algorithm 1 Client Side (Edge layer)**Require:** M_0 : initial global model D : Local data**Ensure:** the gradients of the local loss

- 1: $M_{local} \leftarrow M_0$
- 2: $M_{local} \leftarrow \text{TrainLocalModel}(M_{local}, D)$
- 3: $\text{gradientsLocal} \leftarrow \text{ComputeGradients}(M_{local}, D)$
- 4: $\text{SendToServer}(\text{gradientsLocal})$

using the client's local data (line 2). Through this local training, a client trains its local model M_{local} using its local dataset D to minimize a local loss function : $L_i(M_{local})$. This is done by the formula:

$$M_{local} = \underset{M}{\operatorname{argmin}} L_i(M) \quad (4.8)$$

Following the training phase, the gradients of the local loss with respect to the model's parameters are computed (line 3). These gradients are computed based on the local data and the current model parameters.

$$\nabla L_i = \nabla \text{Loss}(M_{local}, X_i, Y_i) \quad (4.9)$$

Finally, the client sends this gradient to the Cloud layer for aggregation and updating the global model (line 4). We note that clients send only the gradients of their local model parameters to the Cloud layer, rather than sending the entire model or raw data. This approach is essential to maintaining data privacy and reducing communication overhead.

Regarding the server side, its process is described in Algorithm 4. The server plays a pivotal role in managing the aggregation and updates of the global model by utilizing contributions from the devices participating in the training. To begin, the server initializes the global model denoted as M_{global} (line 1). This model serves as the starting point for the collaborative training process. Then, the algorithm proceeds by iterating through a predetermined number of rounds, indicated by the variable $nbrRound$ (line 2). Within each round, the server performs several tasks. It first distributes the current global model M_{global} to the participating clients (line 3). Then, it sets up an accumulator named *aggregated_gradients* to gather the gradients sent by devices (line 4). After that and for each device enlisted in the list of participating devices, the server receives the gradients (*gradients_local*) transmitted by the device. The server subsequently aggregates these local gradients by adding them to the *aggregated_gradients* accumulator (lines 4-8). Once all the gradients have been aggregated, the server calculates the

averaged gradients by dividing the accumulated *aggregated_gradients* by the total number of participating devices (line 9). Afterwards, the server employs the calculated averaged gradients to update the global model (M_{global}) through the *UpdateGlobalModel* function (line 10).

The global model at iteration $t+1$, denoted as $M_{global_{t+1}}$, is obtained by aggregating the local models from the participating clients. In our architecture the aggregation is done using averaging method (where N is the number participating clients):

$$M_{global_{t+1}} = \frac{1}{N} \sum_{i=1}^N M_{global_t}^i \quad (4.10)$$

We note that $M_{global_{t+1}}$ is used as the new global model for the next iteration.

Algorithm 2 Server Side (Cloud layer)

Require: Devices: the participating devices for training

 nbrRound : positiveNumber

Ensure: an updated global model

```

1:  $M_{global} \leftarrow InitializeGlobalModel()$ 
2: for ( $round \leftarrow 0$  to  $nbrRound - 1$ ) do
3:   broadcast  $M_{global}$  to all clients in Devices
4:    $aggregated\_gradients \leftarrow 0$ 
5:   for (each device in Devices) do
6:      $gradients\_local \leftarrow ReceiveGradientsFromClient()$ 
7:      $aggregated\_gradients \leftarrow aggregated\_gradients + gradients\_local$ 
8:   end for
9:    $averaged\_gradients \leftarrow aggregated\_gradients / nbr\_devices$ 
10:   $M_{global} \leftarrow UpdateGlobalModel(M_{global}, averaged\_gradients)$ 
11: end for
12: return ( $M_{global}$ )

```

4.2 The Proposed SML-Based Architecture for Resource-Constrained CPS

While FML offers significant privacy benefits by keeping data on local devices and only sharing model updates, it has limitations when applied to resource-constrained CPS. FML relies on the computational capabilities of client devices to train local models, which can be problematic in environments where devices have limited processing power, memory, and energy resources.

To address these constraints, we propose an architecture based on SML. Unlike FML, SML divides the model training process between the client and server, significantly reducing the computational burden on client devices. In SML, the initial layers of the model are trained on the client side, producing intermediate representations of the data. These representations, which

are much smaller and less complex than raw data, are then sent to the server. The server, with its more substantial computational resources, completes the training of the remaining layers of the model.

This approach offers several advantages for resource-constrained CPS:

- **Reduced Client-Side Computation:** By offloading the majority of the training process to the server, SML minimizes the computational demands on client devices, making it feasible for devices with limited resources.
- **Improved Energy Efficiency:** With less processing required on the client side, SML helps extend the battery life of resource-constrained devices, which is crucial in CPS environments.
- **Maintained Data Privacy:** Similar to FML, SML ensures that raw data remains on the client side, enhancing data privacy. Only abstract, intermediate representations are transmitted to the server, reducing the risk of data leakage.
- **Scalability:** SML is more scalable for large-scale CPS deployments, as the computationally intensive tasks are handled by the server, allowing more devices to participate in the learning process without overloading their limited resources.

4.2.1 Architecture Components

In this subsection, we outline the key components of our SML-based architecture and illustrate how they operate together. We focus on the enhancements introduced compared to our previous architecture presented in Figure 4.2. As illustrated in Figure 4.4, our proposed SML-based architecture for resource-constrained CPS retains the same fundamental components as the FML-based architecture, comprising three main entities: local data collectors, edge aggregators, and the main server. The primary difference lies in the training process, which is optimized to suit resource-constrained environments.

- **Local Data Collectors:** These entities are responsible for collecting data from various sensors and devices within the CPS. They preprocess the data and prepare it for training.
- **Edge Aggregators:** Edge aggregators manage the client-side model (M_c), orchestrate the training process, and actively collect data from local collectors. The client-side model is initialized with random weights, and local training is conducted on the initial layers of the neural network, up to a designated cut layer that separates the client-side and server-side segments.

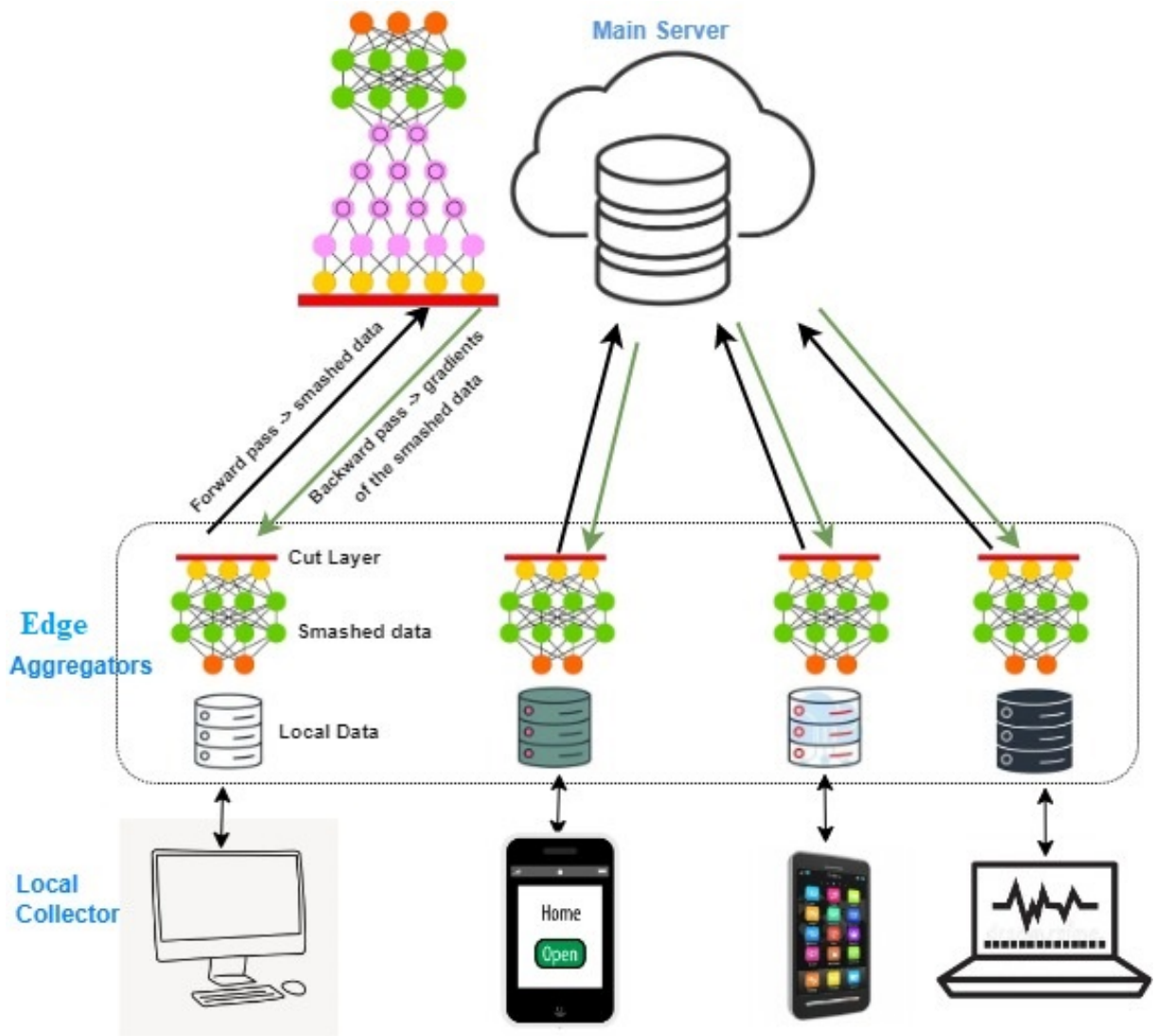


Figure 4.4: The proposed SML-based architecture

- **Main Server:** The main server, or Cloud layer, communicates with clients in a sequential manner to ensure a structured learning process. It completes the training process by handling the later layers of the neural network and aggregating updates from multiple edge aggregators.

In our proposed SML-based architecture for resource-constrained CPS, training is optimized to mitigate the computational limitations of client devices while preserving data privacy. The architecture uses edge aggregators to manage client-side model training, where initial layers of neural networks are locally trained up to a designated cut layer. During training iterations, activations (referred to as 'smashed data') from these layers are securely transmitted to a main server. The server then processes this data through its remaining layers, facilitating the re-

finement of a global model. This iterative process ensures that computations are efficiently distributed, enhancing model accuracy without compromising the privacy of raw data collected by local sensors within the CPS environment.

4.2.2 Collaborative Training in our SML-architecture

In our SML-based architecture, the primary objective is collaborative model training, wherein the server collaborates with each client in sequence for model training. Each client possesses an individual local dataset denoted as D_i .

In our architecture, the training process comprises a Client (C) and a server (S). A global model, denoted as M_{global} , is created, which consists of two distinct parts: M_c (Client-side Model) and M_s (Server-side Model). Algorithm 3 outlines the behavior of each client within our architecture. The M_c is responsible for processing the initial layers of the neural network, conducting local training on the client's unique dataset, and generating "smashed data" after forward propagation up to a designated cut layer. On the other hand, the M_s , residing on a central server, utilizes the smashed data received from the client to complete forward propagation on the remaining layers of the global model. The server side process is described in Algorithm 4. The server conducts forward propagation on the server-side model (M_s) using the received smashed data to obtain to minimize a loss function : $L_i(\text{smash_data})$. This is done by the formula:

$$M_s = \text{argmin}_M L_i(\text{smash_data}) \quad (4.11)$$

After the training phase, the gradients ∇L_i of the loss function L_i with respect to the model's parameters are calculated. These gradients are determined using the *smash_data* and the current model parameters, as shown in formula 4.12.

$$\nabla L_i = \nabla \text{Loss}(M_s, \text{smash_data}) \quad (4.12)$$

This process involves computing the loss function, engaging in backpropagation, and updating its own weights until reaching the cut layer. The gradients corresponding to the smashed data are then communicated back to the client. This collaborative training process between the M_c and the M_s iterates until convergence, contributing to the refinement of the M_{global} .

Algorithm 3 Client-Side Algorithm

Require: M_c : Client-side Model with random weights D_c : Local data on the client side num_epochs : Number of training epochs**Ensure:** Smashed data for the server

```

1: for epoch  $\leftarrow$  1 to  $num\_epochs$  do
2:    $M_c \leftarrow TrainLocalModel(M_c, D_c)$ 
3:    $smash\_data \leftarrow M_c.get\_smash\_data()$ 
4:    $SendToServer(smash\_data)$ 
5:    $gradients\_from\_server \leftarrow ReceiveFromServer()$ 
6:    $M_c.backward(gradients\_from\_server)$ 
7: end for

```

Algorithm 4 Server-Side Algorithm

Require: M_s : Initial server-side model $clients$: List of clients**Ensure:** Updated gradients

```

1: for client in  $clients$  do
2:   for epoch  $\leftarrow$  1 to  $num\_epochs$  do
3:      $smash\_data \leftarrow receive\_from\_client(client)$ 
4:      $predictions \leftarrow M_s.forward(smash\_data)$ 
5:      $loss \leftarrow calcul\_loss(predictions, smash\_data)$ 
6:      $M_s.backward(loss)$ 
7:      $send\_to\_client(client, M_s.get\_gradients())$ 
8:   end for
9: end for

```

5 Conclusion

This chapter has presented a comprehensive framework for adaptive CPS, encompassing a meta-model, a centralized ML architecture, and its enhancement through collaborative ML. The meta-model provides a foundation for understanding and modeling the complex interactions between physical and digital components within CPS. The centralized ML architecture leverages ML techniques to enhance the adaptability and efficiency of CPS. However, the centralized architecture is constrained by data security and privacy concerns, scalability limitations, and communication overhead.

To address these limitations, we introduced collaborative ML approaches, specifically FML and SML. These approaches decentralize model training and data processing, thereby enhancing data privacy and scalability while reducing communication overhead in resource-constrained CPS environments. FML allows local devices to collaboratively train a global model without sharing sensitive data, while SML partitions the training process between client-side and server-side components, optimizing computational resources and preserving data privacy.

In the next section, we will validate our proposed framework through a case study on fault prediction in industrial CPS. This validation aims to demonstrate the effectiveness and practical application of our collaborative ML-enhanced architecture in real-world scenarios, showcasing its potential to improve reliability and operational efficiency in industrial settings.

5. Case Study - Fault Prediction in CPS

Using Centralized/Decentralized ML

In this chapter, our focus turns to the fault prediction module within the learning application layer of our architecture. This module will exploit ML techniques for predicting and anticipating potential problems before they lead to device failure. The implementation of this module serves as a practical exploration of the feasibility and efficacy of our proposed approach.

This chapter begins with the deployment of the fault prediction module within the confines of our centralized architecture. Subsequently, we replicate and refine the implementation within the framework of our FML-based architecture. Prior to delving into the implementations, we will first present a detailed overview of the structure of the fault prediction module, elucidating its components and functionalities. Additionally, an introduction to the dataset employed in this case study will be provided, outlining its characteristics and relevance to fault prediction.

The chapter concludes with a comprehensive comparison of the results obtained from both implementations, offering valuable insights into the performance and effectiveness of each architecture in the context of fault prediction within CPS.

1 Fault Prediction Module Structure

The main goal of the Fault Prediction module is to predict whether a device is exposed to failures or not. It allows, also, to identify the failure type. Therefore, its architecture will make use of supervised and unsupervised learning algorithms. First, we use classification techniques for separating data into two classes: faulty and correct data. Next, we apply the clustering method to the faulty data in order to group them into many clusters depending on fault type. Finally, we apply prediction. As shown in figure 5.1, the functioning of the Fault Prediction module contains the following steps:

- **Classification:** This step permits to separate data them into two classes: correct data and faulty data.
- **Clustering:** Here we treat only the faulty data, and this is for the purpose of grouping them into similar clusters, which allows us to know the nature of the error in each group of wrong data.
- **Prediction:** Here comes the prediction stage with the use of new data.

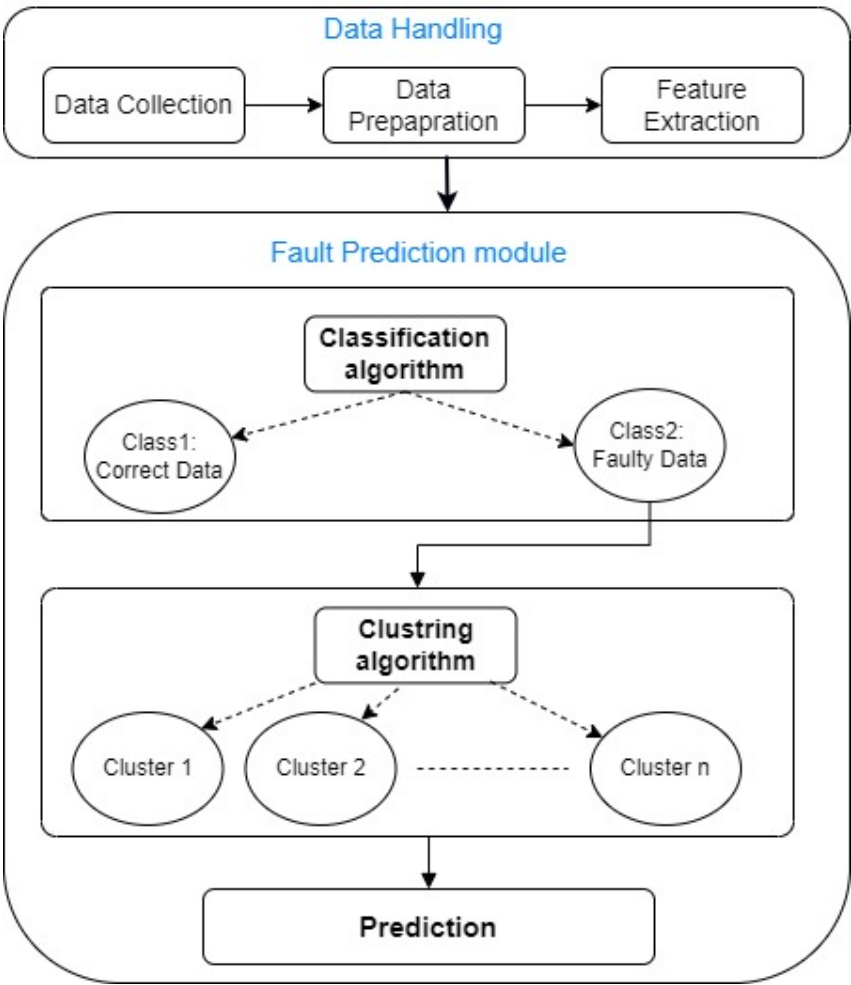


Figure 5.1: Fault Prediction Operating Principle

2 Dataset Description

The data was collected from «Kaggle» known as predictive maintenance which offers us a synthetic data set [70] that reflects the actual predictive maintenance data of a system encountered in the industry. The dataset consists of 10,000 data points stored as rows with 6 features in columns.

- **product ID** : consisting of a letter L, M, or H for low (50% of all products), medium (30%) and high (20%) as product quality variants and a variant-specific serial number.
- **air temperature [K]** : generated using a random walk process later normalized to a standard deviation of 2 K around 300 K
- **process temperature [K]** : generated using a random walk process normalized to a standard deviation of 1 K, added to the air temperature plus 10 K.
- **rotational speed [rpm]** : calculated from a power of 2860 W, overlaid with a normally distributed noise
- **torque [Nm]** : torque values are normally distributed around 40 Nm with a $\sigma = 10$ Nm and no negative values.
- **tool wear [min]** : The quality variants H/M/L add 5/3/2 minutes of tool wear to the used tool in the process.

Additionally, in this dataset, we have the 'machine failure' label that indicates whether the machine has failed in this particular datapoint. If the process fails, the 'machine failure' label is set to 1, which is the case for 339 datapoints.

3 Implementation with Centralized Architecture

In the implementation of the fault prediction module within our centralized architecture, we adhere to the structured approach presented in section 1.4 of chapter 2. This approach ensures methodological rigor and facilitates meaningful insights into the performance of the system. As presented in figure 2.6, these steps are described as follows:

- **Data Collection:** As presented in the previous section, this step deals with the acquisition of relevant datasets for training and testing purposes.

- **Data Preparation:** This step involves preprocessing the collected data to ensure its suitability for the fault prediction module. This involves tasks such as Exploratory Data Analysis, Feature Selection, Handling Unbalanced Data, and data fraction.
- **Choice of Algorithm:** Selection of a suitable machine learning algorithm tailored to the specific characteristics of the fault prediction problem. This step involves a careful consideration of algorithmic strengths, weaknesses, and compatibility with the dataset.
- **Model Training and Evaluation:** The chosen algorithm is applied to the prepared dataset for training, and the model's performance is evaluated using distinct metrics. This step gauges the model's ability to learn from the data and make accurate predictions.
- **Parameter Tuning:** Optimization of model parameters to enhance predictive accuracy. Through systematic tuning, we strive to fine-tune the model for optimal performance, ensuring it generalizes well to unseen data.
- **Prediction:** The final model is employed to make predictions on new data.

The software development environment, comprising tools such as Google Colab, Python, Keras, NumPy, and others, along with the hardware environment, are detailed in the appendix of this manuscript

3.1 Data preparation

Data preparation is the most important phase in the ML process. It involves cleaning, structuring and organizing data. Following the ML pipeline, This step involves tasks such as Exploratory Data Analysis, Feature Selection, Handling Unbalanced Data, and data fraction.

3.1.1 Exploratory Data Analysis

This step is very critical for analyzing and understanding the insights of the key characteristics of the data. It is mainly used to discover patterns and anomalies. Furthermore, it consists of extracting correlations or relationships between the different data by identifying the most and the least related attributes. To do so, data visualization methods have to be applied such as correlation matrix, and pair plots. The Correlation matrix measures both the strength and direction of the linear relationship between two variables. Correlation values are standardized. It is given by the following formula:

$$r_{XY} = \frac{Cov(X,Y)}{\sigma_X * \sigma_Y} \quad (5.1)$$

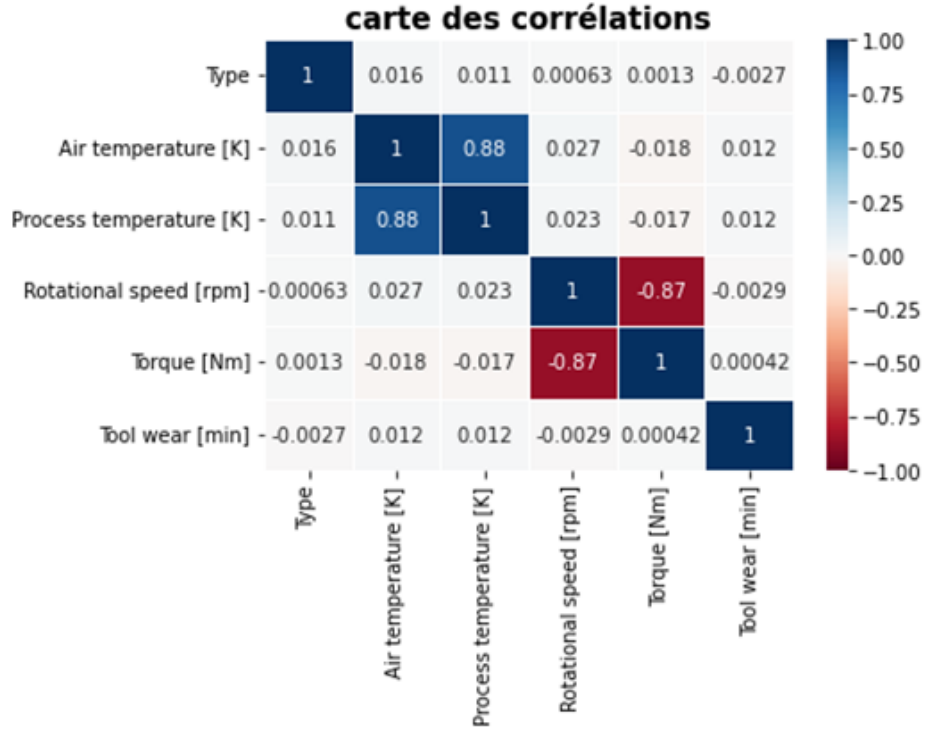


Figure 5.2: Correlation matrix between the different features

Where:

- $cov(X, Y)$ is the covariance between variables X and Y.
- σ_X and σ_Y are the standard deviations of X and Y, respectively.

The formula for the covariance $cov(X, Y)$ between two variables X and Y is given by:

$$cov(X, Y) = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{N} \quad (5.2)$$

Where:

- X_i and Y_i are individual data points in the datasets X and Y.
- $\bar{X}$ and $\bar{Y}$ are the means of X and Y, respectively.
- N is the number of data points.

Figure 5.2 represents the correlation matrix of the different attributes in the selected data. We can see, clearly, that the two attributes air temperature and process temperature are highly

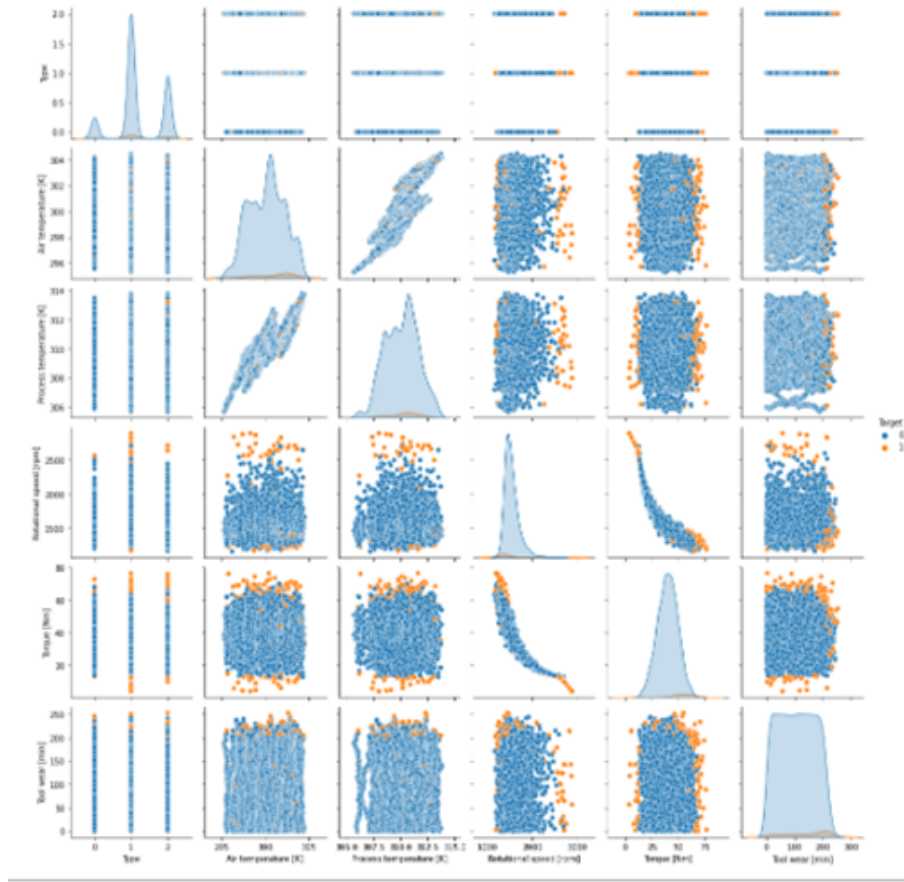


Figure 5.3: Bivariate relationship between each pair of features

correlated. In this step, we can also study the bivariate relationship between each pair of features in order to analyze the different distribution of features. Figure 5.3 depicts the relationship between the features of our dataset. It can be seen that the target values are highly unbalanced, i.e, the target feature has more observations in one specific class, which is the correct data class. Moreover, we notice that the following attributes: air temperature, process temperature pair, torque, and rotational speed are significantly different in their distribution compared to other attributes.

3.1.2 Feature Selection

The selection process allows identifying the most efficient features for prediction by selecting the attributes with the highest score and removing those with the lowest one. In our case, we have used the *SelectKBest* technique to pick out the top k features with the highest ranked scores. Table 5.1 presents the score of each feature of our dataset. It is shown that Torque, Tool wear, Air temperature, Rotational speed, and process temperature are the features with the

Features	Scores
Torque [Nm]	342.79
Tool wear [min]	88.93
Air temperature [K]	61.95
Rotational speed [rpm]	16.05
Process temperature [K]	09.95

Table 5.1: Scores of each features

highest scores.

3.1.3 Handling Unbalanced Data

In classification, training data may exhibit classes with imbalanced proportions, where the target class has a skewed distribution of observations. This imbalance can lead to misleading accuracy metrics, as the model might struggle to effectively predict instances belonging to the minority class. In our dataset, the class representing correct data (Target is 0) has a significantly higher number of observations, while the other class has a notably lower number of instances.

To address this issue, we employed the Synthetic Minority Oversampling Technique (SMOTE) method, one of the widely used oversampling techniques for handling class imbalance. SMOTE aims to rectify the distribution of classes by randomly augmenting examples of the minority class through synthetic data generation.

3.1.4 Data fraction

The prediction model must be tested on data that it has not yet encountered to see whether it can successfully generalize new data from the data has already seen. To do so, we must split data into two sub-sets: training set (with a majority of 75% of samples) and test set (with a minority of samples 25%). We emphasize that *train_test_split* function, of *scikit-learn* library, has been used to split randomly our data according to arbitrary proportions.

3.2 Choosing ML Algorithms for Fault Prediction

In the process of developing the Fault Prediction module, the choice of ML plays a crucial role in ensuring the effectiveness and accuracy of the model. This section outlines the rationale behind the selection of two specific algorithms, one for classification and the other for clustering.

Classifier	Accuracy	Runtime
RF	97.66	1.96
Decision Tree	95.76	0.06
KNN	95.44	0.21
SVM	84.24	12.48
GaussianNB	82.22	0.02

Table 5.2: The obtained accuracy of each classifier

3.2.1 Classification Algorithm: Identifying Correct and Faulty Data

The first algorithm employed in the Fault Prediction module is a classification model. The main objective of this model is to categorize data into two distinct classes: correct and faulty. This binary classification serves as the initial step in the fault prediction pipeline, allowing the system to differentiate between normal operation and anomalous behavior.

Many ML algorithms have been successfully applied for Classification domain such as: Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Random Forest (RF), Gaussian Naive Bayes, and Decision Tree. In order to choose the best classifier, we conducted a series of test to compare the aforementioned supervised machine learning predictive models using our dataset. We note that we have applied the default models without any hyper-parameter setting and compared their accuracy.

Table 5.2 depicts the accuracy of each model. We can see clearly that the RF model outperforms the other models in terms of accuracy. Also, RF is capable of handling large datasets with high dimensionality. It enhances the accuracy of the model and prevents the over fitting issue. For the above reasons, we chose to select RF Classifier in order to build a model that can efficiently classify out data into two classes (faulty and correct data).

During the classification phase with RF, we have used confusion matrix to visualize the performance of the model by displaying the measurements of true positives (TP), true negatives (TN), False Positives (FP), and False Negatives (FN). The model correctly predicted all observations that fall on the matrix diagonal, while observations that fall outside the diagonal correspond to errors in the model. In Figure 5.5, we can see that our model was able to predict 2142 data belongs to faulty class and 2098 belongs to correct class. However, 69 cases belong to faulty class were estimated otherwise. In the other side, 33 cases that belongs to correct class were predicted wrongly.

Finally, classification report is calculated to evaluate the performance of RF model. It allows us to measure the quality of the predictions of the algorithm by showing the scores of the

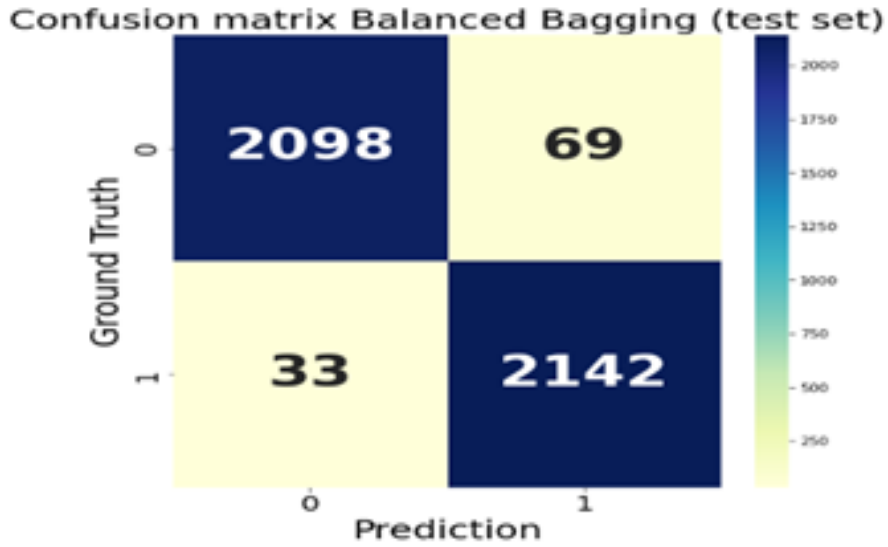


Figure 5.4: Confusion Matrix

accuracy, precision, recall and F1-score. These metrics are calculated as follows:

- **Accuracy:** is the proportion of correctly classified instances out of the total instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.3)$$

- **Precision:** is the ratio of correctly predicted positive observations to the total predicted positives. It is a measure of the accuracy of the positive predictions.

$$Precision = \frac{TP}{TP + FP} \quad (5.4)$$

- **Recall:** is the ratio of correctly predicted positive observations to all observations in the actual class. It measures the ability of the model to capture all the positive instances.

$$Recall = \frac{TP}{TP + FN} \quad (5.5)$$

- **F1-score:** is the harmonic mean of precision and recall. It provides a balance between precision and recall, especially when the classes are imbalanced.

$$F1 - score = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \quad (5.6)$$

Table 5.3 presents the comprehensive evaluation results for our RF model. The RF model

Metric	Class 0 (Faulty data)	Class 1 (Correct data)
Accuracy	0.9766	0.9766
Precision	0.9841	0.9670
Recall	0.9707	0.9839
F1-score	0.9774	0.9754

Table 5.3: Performance Metrics for Classification Using RF

demonstrated impressive performance with an overall accuracy of approximately 97.66%. This indicates that the model effectively classified instances into faulty and correct data. The high precision values for both faulty (Class 0) and correct (Class 1) data (approximately 98.41% and 96.70%, respectively) reflect the model's accuracy in making positive predictions. The recall values, approximately 97.07% for faulty data and 98.39% for correct data, signify the model's ability to capture a high proportion of actual instances. The F1-scores, balancing precision and recall, are approximately 97.74% for faulty data and 97.54% for correct data, indicating a harmonious trade-off. Overall, these metrics collectively highlight the model's robustness and effectiveness in classifying data into the designated categories.

3.2.2 Clustering Algorithm: Grouping Predicted Faulty Data

After identifying potentially faulty instances, the clustering model comes into play to further analyze and group these data points into clusters. Clustering provides valuable insights by revealing patterns or similarities among the identified faulty data, aiding in the interpretation and diagnosis of the detected faults. This step enhances the fault prediction process by offering a more granular understanding of the diverse types of faults present in the system.

The most used clustering algorithms are: k-means, hierarchical clustering, and fuzzy C-means clustering. Several comparative studies [42][125] demonstrate that K-means exhibited a better performance in terms of accuracy than the other methods investigated. In our case, we have used K-means to group faulty data into clusters. The fact that this technique adapts to new examples, simple to implement and returns clusters that can be easily interpreted and even visualized motivate this choice.

The initial step in applying K-means involves determining the optimal number of clusters, denoted as 'K'. To achieve this, we employ the Elbow method, a popular technique for identifying the point at which the addition of more clusters does not significantly improve the model's performance. Figure 5.5 provides a crucial insight into the identification of the optimal 'K' value. By analyzing the curve, we observe an 'elbow' point, where the rate of improvement diminishes. In our case, the Elbow method suggests the presence of 4 clusters as the optimal

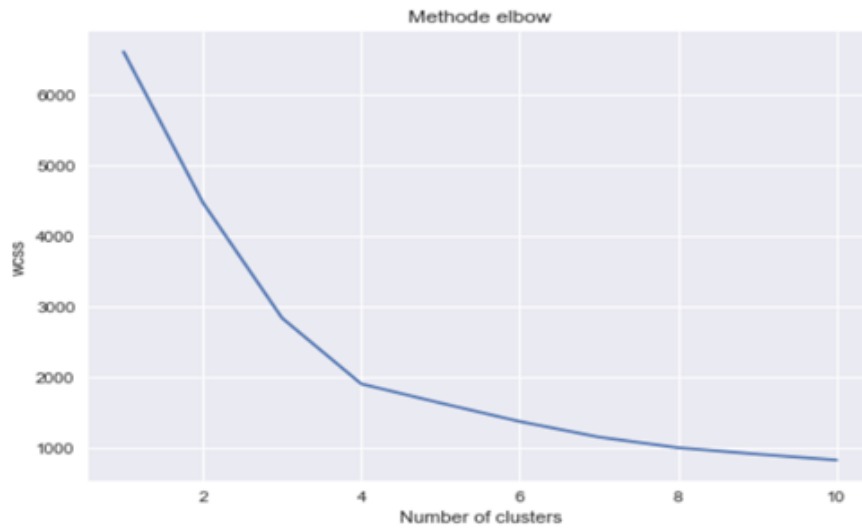


Figure 5.5: Elbow method curve

configuration for the K-means algorithm.

In the next step, we apply the K-means clustering algorithm with 4 clusters as parameters. This allows the algorithm to group similar instances within the faulty data, facilitating the extraction of meaningful insights and potentially revealing distinct patterns or subsets that might not be immediately apparent through other analytical approaches. Figure 5.6 illustrates the clustering results, showcasing how the faulty data is organized into distinct clusters within the three-dimensional space. Notably, the representation highlights the presence of four discernible clusters, each corresponding to a specific type of failure—designated as types 0, 1, 2, and 3.

3.3 Deploying the Trained Model for Real-world Predictions

In the final phase of our fault prediction module, we transition to the prediction stage where the model is tested on unseen observations. Upon successful prediction, the fault prediction module has the capacity to generate an output dataset encompassing all instances that are deemed susceptible to failure, along with their corresponding failure types. This output serves as a valuable resource for further analysis, decision-making, and proactive maintenance efforts within the industrial process.

While our centralized learning-based architecture has demonstrated impressive results in terms of accuracy and precision, it is not without its challenges. Concerns regarding data security, privacy, scalability, stability, and communication overhead have surfaced. The centralized model necessitates transmitting large volumes of data to a central server, raising privacy issues and causing potential communication bottlenecks. In transitioning to a decentralized architec-

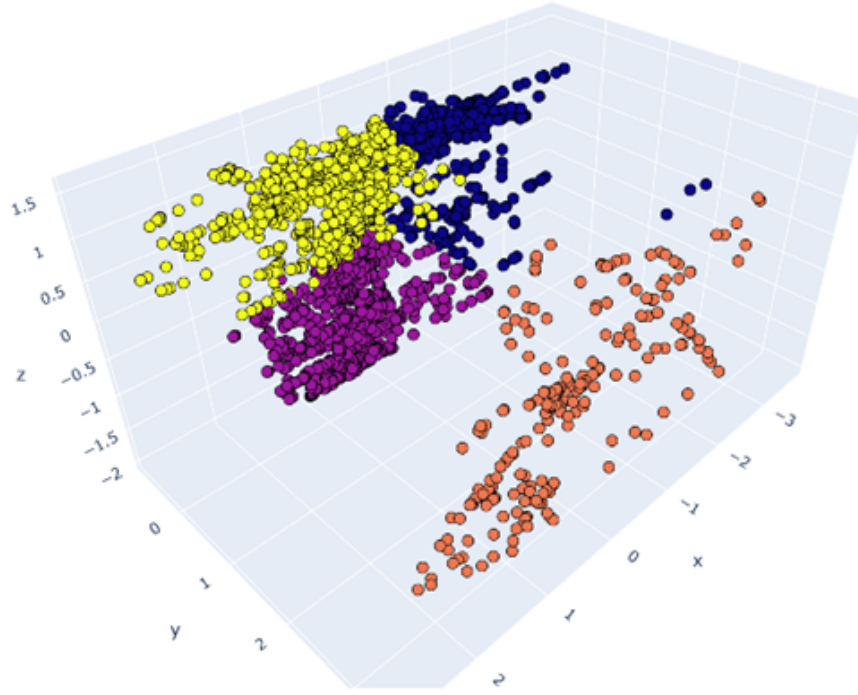


Figure 5.6: Clustering result in 3D

ture with FML, we aim to address these limitations while diligently striving to maintain the high-performance achieved by our centralized model.

4 Implementation with Decentralized Architecture

In this section, we present the implementation of our proposed fault prediction module within both the FML and SML-based architectures. This implementation follows a structured set of steps to ensure a seamless and effective integration of the collaborative ML techniques into our adaptive CPS framework.

We specifically focus on two critical stages of the implementation process: Data Preparation and Model Training. Data Preparation involves gathering, preprocessing, and partitioning the data in a manner suitable for decentralized learning environments. Model Training entails the actual training of the machine learning models using collaborative techniques.

The implementation was carried out using the Google Federated TensorFlow (TFF) platform, which provides robust tools for collaborative learning. For the FML-based architecture, we used the Federated Averaging algorithm (FedAvg) algorithm to enable efficient model aggregation across distributed devices. In the SML-based architecture, we applied the Vanilla SML algorithm to effectively split the model training process between client and server.

Details and motivations for our technical choices, including the selection of TFF, FedAvg, and Vanilla SML, are provided in Annex 2. These choices were made to ensure scalability, efficiency, and privacy preservation in our collaborative ML framework.

4.1 Data Preparation

The dataset that was used in the centralized approach is also being used in this step. This strategic decision allows for a direct comparison between the results obtained from the centralized and decentralized models, providing insights into the relative performance of the two approaches.

In our approach the most important stages of data preparation is density estimation which refers to the process of estimating the underlying probability distribution of data within each client's local dataset without aggregating the actual data itself. This estimation is performed locally on individual clients, and the results are then used to guide the training process. In this step, we conducted initial experimentation to analyze and study the data from our clients. With the aim of evaluating our proposal, we partitioned the dataset across three distinct clients. This partitioning was carried out over multiple iterative phases, the goal being to achieve a reasonable correlation among the datasets of the various clients. To facilitate this analysis, we used data visualization methods, including correlation matrix plot. This allowed us to visually observe the relationships between different variables and highlight emerging patterns and trends. Fig. 5.7 presents the data visualization of the three clients.

Fig. 5.7 shows a clear correlation among the data of the various clients, particularly in relation to the variables Rotational speed, Torque, and Tool wear. A significant correlation in density estimation suggests that the data distributions across clients are aligned. This can simplify model training, aggregation, and result in faster convergence.

4.2 Training process in FML-based architecture

As previously discussed in chapter 4, the training process in FML relies on collaboration between the server and the clients. This collaborative process iterates over multiple rounds with the objective of achieving a minimal loss value and a high degree of accuracy. As demonstrated in Algorithm 3, local training relies on calculating the loss value, which is a measure of how well the model's predictions align with the actual target values in the training data. A smaller loss value means that the model's predictions are closer to the actual data points.

Fig. 5.8 shows the loss average during local optimization. We observe that the number of training rounds can significantly affect the loss value. In the initial rounds of training, we

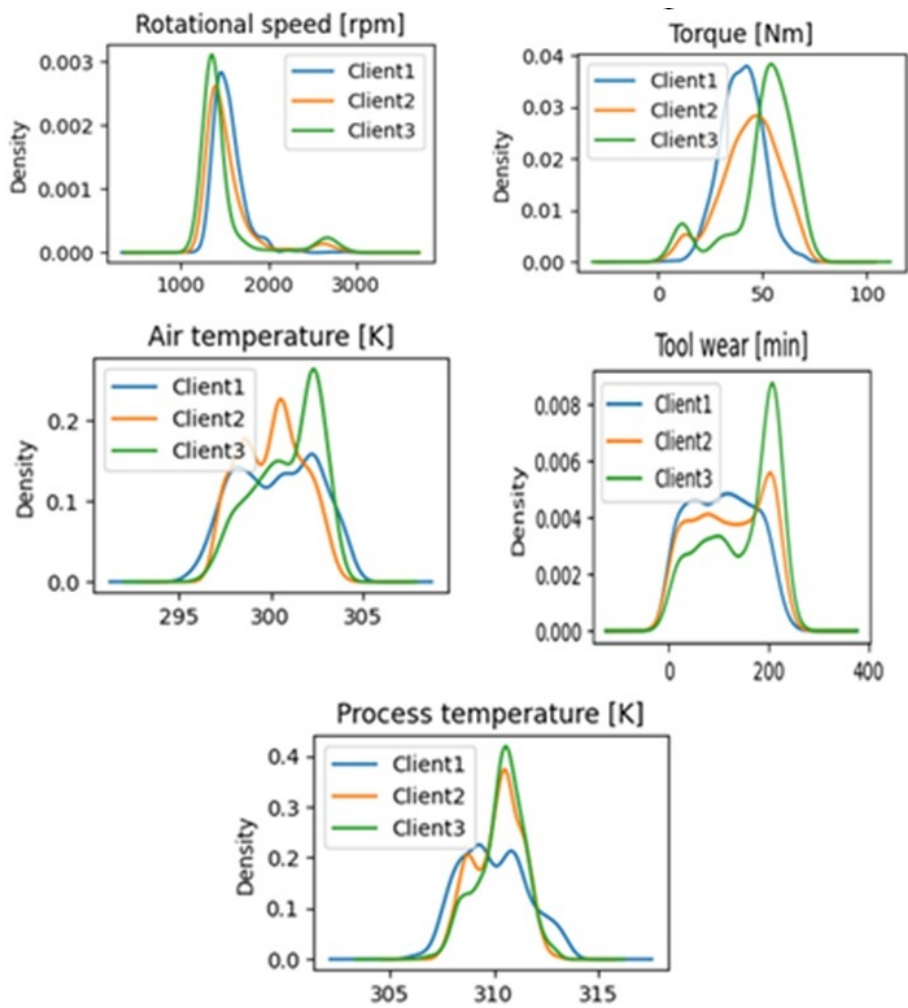


Figure 5.7: Data visualization

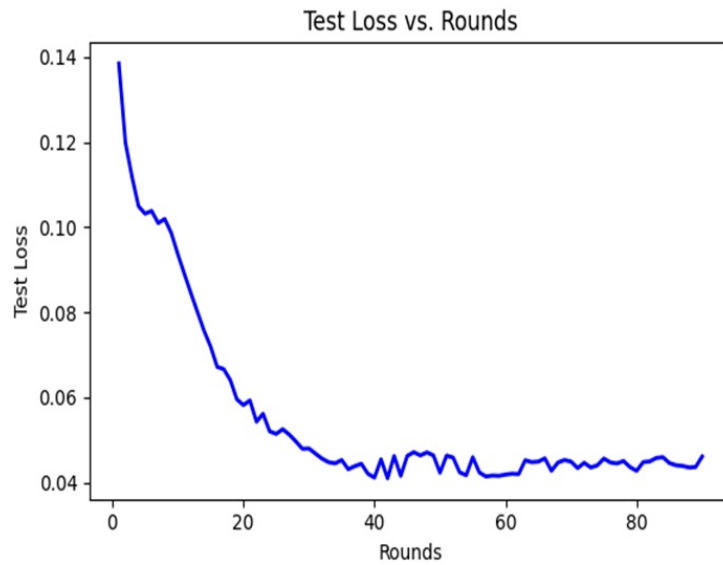


Figure 5.8: Loss values vs number of rounds

observe an important reduction in the loss value. This is because the model is learning from the client data and making significant improvements. After a sufficient number of rounds (round = 40), the loss value is stabilizing at a certain minimum value. This minimum represents the best achievable performance given the data and model architecture. Further rounds might not significantly reduce the loss. We tried to increase the number of rounds, but we noticed that the loss value is constant and sometimes increasing, and this indicates that the model has reached a point of convergence. Convergence means that the model has learned most of the patterns and relationships present in the training data, and further training does not significantly improve performance.

Another important aspect that warrants attention when training local models is the occurrence of Overfitting, which mean that the model learns the training data too well. In this case, the model might become too specialized in capturing noise in the training data, leading to poor generalization on unseen data.

Fig. 5.9 provides a clear depiction of how accuracy is related to the number of training rounds. We observe that accuracy increases until it reaches a fixed or stable value, it typically indicates that the model has converged and it has reached its optimal performance on the given data. This stable accuracy value is positive because it indicates that the model has not overfit the training data by memorizing noise or irrelevant details.

As a result of this experimentation, it is essential to strike a balance between training long enough for the model to learn but not so long that it starts to overfit or reaches a point of diminishing returns in terms of improvement. In fact, monitoring both training and validation

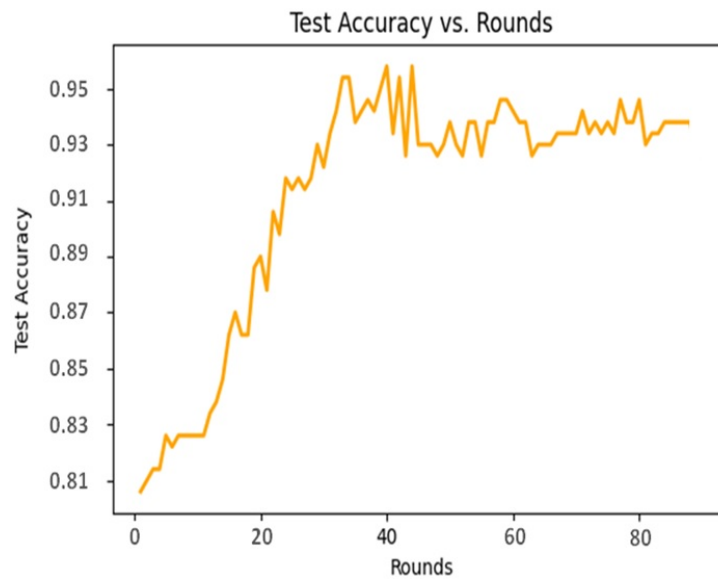


Figure 5.9: Accuracy vs number of rounds

accuracy is crucial to make informed decisions about when to stop training and to ensure the model achieves the best overall performance. We note that in this experimentation we gave great attention to preparing data while using FML-based techniques to train local models in order to obtain high accuracy, so that these results allow us to obtain good expectations with the new data (unseen data). Among the techniques used during training:

- **Client-Level Regularization:** We apply regularization techniques like Federated L1 and Federated L2 on the client side to encourage simpler local models.
- **Early Stopping:** Monitor the performance of the global model on a validation dataset during training rounds to detect when the model starts to overfit.
- **Learning Rate:** Learning rate controls the step size during gradient descent. Tuning it helps find the right balance between convergence speed and model accuracy.
- **Number of Training Rounds:** The number of training rounds affects convergence and model accuracy.

In addition to the accuracy of 0.95 obtained (as illustrated in Figure 5.9), our decentralized architecture demonstrates remarkable performance metrics. The precision, reflecting the ratio of true positive predictions to the total predicted positives, attains a notable value of 0.94. Similarly, the recall, representing the proportion of actual positives correctly predicted by the model, achieves a commendable 0.95.

4.3 Training process in SML-based architecture

As elucidated in chapter 4, the training process in SML hinges on a synergistic collaboration between the server and the clients. This collaborative endeavor unfolds across multiple epochs, aiming to minimize the loss function and maximize the model's accuracy.

The central element of the training process is the computation of the loss value on the server side. This loss value serves as a crucial metric, quantifying the disparity between the model's predictions and the actual target values in the training dataset. A smaller loss value indicates a closer alignment between the model's predictions and the smashed data points, signifying an improved predictive capability.

This approach ensures that the training process not only refines the model's parameters but also steers it towards a state where its predictions better capture the underlying patterns within the training data.

Figure 5.10 depicts the relationship between the number of epochs and both the loss and accuracy for each client. It is evident that the number of training epochs plays a crucial role in shaping the loss values. During the initial training epochs, a substantial reduction in the loss is observed, indicating significant improvements as the model learns from client data. As training progresses (epochs = 70), the loss stabilizes at a minimum value, representing the optimal performance achievable with the given data and model architecture. Trying to add more epochs shows that the loss levels stabilize, indicating that the model has reached a point where further training doesn't result in substantial improvement, signifying convergence. Convergence signifies that the model has effectively captured patterns and relationships present in the training data.

Regarding accuracy, the graph illustrates dynamic changes during the training process. As the number of training epochs increases, accuracy steadily rises until it attains a stable value, reaching this stability at around 70 epochs. This stability suggests that the model has learned comprehensively, performing optimally on the provided dataset. The attainment of a consistent accuracy level indicates that the model has successfully learned and adapted to the underlying patterns in the training data. Notably, at each epoch, the server leverages the smashed data received from the clients to enhance the global model, contributing to the continuous improvement in accuracy. After an initial increase during the early epochs, the accuracy values for all three clients stabilized at an average of 0.9756.

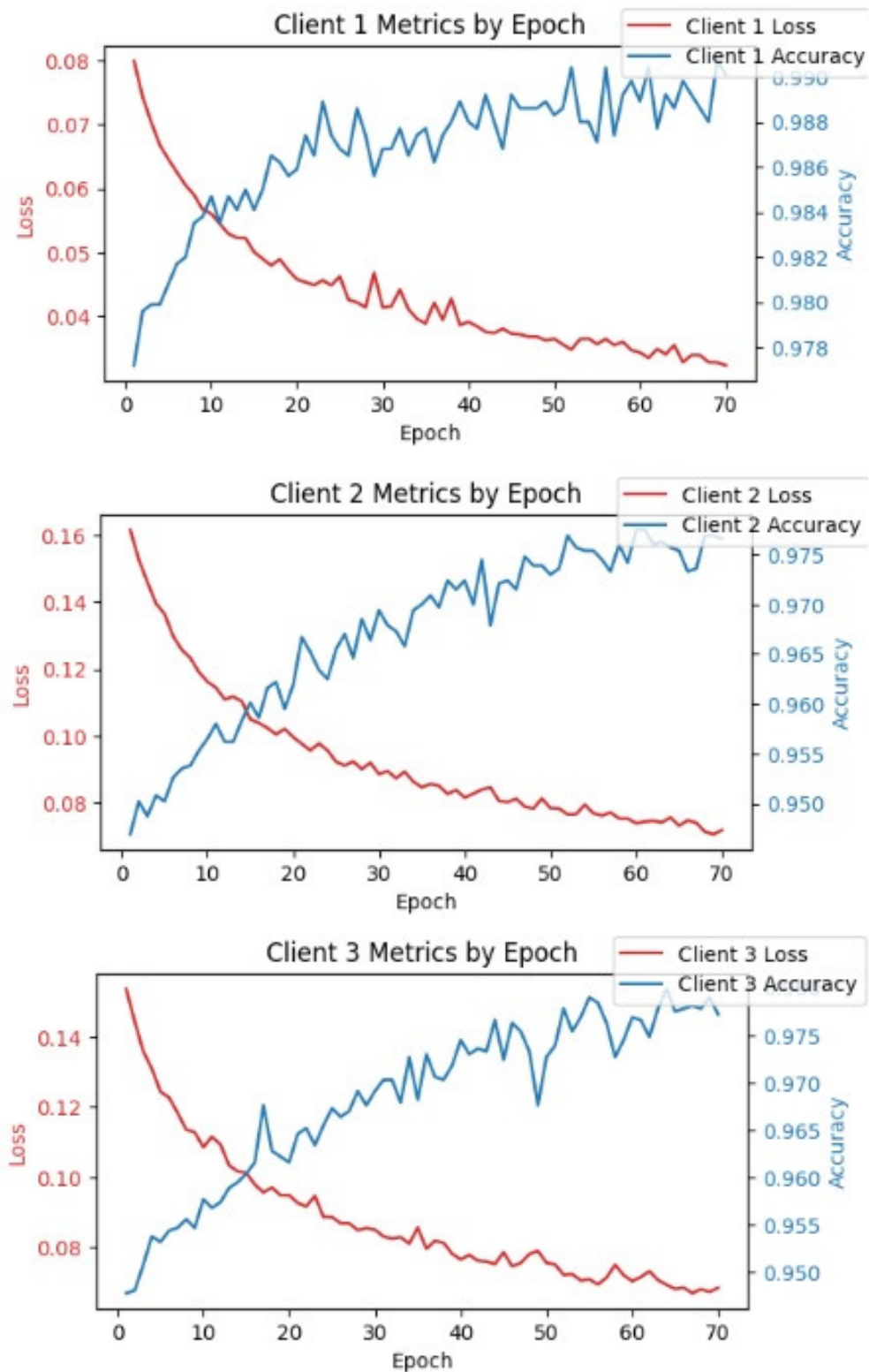


Figure 5.10: Loss and accuracy convergence for each client

4.4 Comparative Analysis

Table 5.4 presents the results obtained by the three architectural approaches proposed in this thesis. It’s important to note that we utilized the same dataset for the sake of comparison. Like presented in Table 5.4, it is striking that both the centralized and SML techniques show similar degrees of accuracy, with both achieving accuracies up to 97%. This parity in accuracy underscores the viability of SML as a competitive alternative to centralized learning approaches.

Table 5.4: Comparison of accuracy metrics across different ML approaches

ML approach	Accuracy
SML	0.9756
FML	0.9544
Centralized	0.9785

To further assess the performance of decentralized approaches in resource-constrained environments, we delve into a comparative analysis between FML and SML approaches. While Table 5.4 provides insights into the accuracy metrics across different ML approaches, additional metrics are necessary to evaluate their efficiency in resource-limited contexts. Specifically, we focus on the "Learning time for each client" metric, which sheds light on the computational resource utilization of each approach.

Maintaining consistent experimental configurations for FML and SML ensures a fair comparison. This includes using identical client configurations and dataset distributions across clients. Such uniformity enables an accurate evaluation and comparison of the performance of FML and SML under resource-constrained environments.

As presented in Figure 5.11, the results obtained from the experiments reveal notable differences in the learning time for each client between FML and SML. In resource-constrained environments, where efficient resource utilization is paramount, SML demonstrates a clear advantage. The learning time for each client in SML is significantly shorter compared to FML, indicating that clients utilizing SML consume fewer computational resources.

Specifically, the experimental results are summarized in Table 5.5, which shows the comparison of learning time between FML and SML for each client.

On average, across all clients, the SML approach achieves a reduction in learning time of approximately 41%. These results highlight the efficiency of SML in reducing learning time, which is critical for real-time applications in CPS. By significantly lowering the computational load on clients, SML not only enhances learning speed but also conserves resources, making it a highly suitable approach for resource-constrained environments.

Client	Learning Time in FML	Learning Time in SML	Reduction (%)
Client 1	12.2 seconds	9.5 seconds	22.13%
Client 2	11.3 seconds	6.0 seconds	46.90%
Client 3	11.3 seconds	5.0 seconds	55.75%

Table 5.5: Comparison of learning time between FML and SML for each client

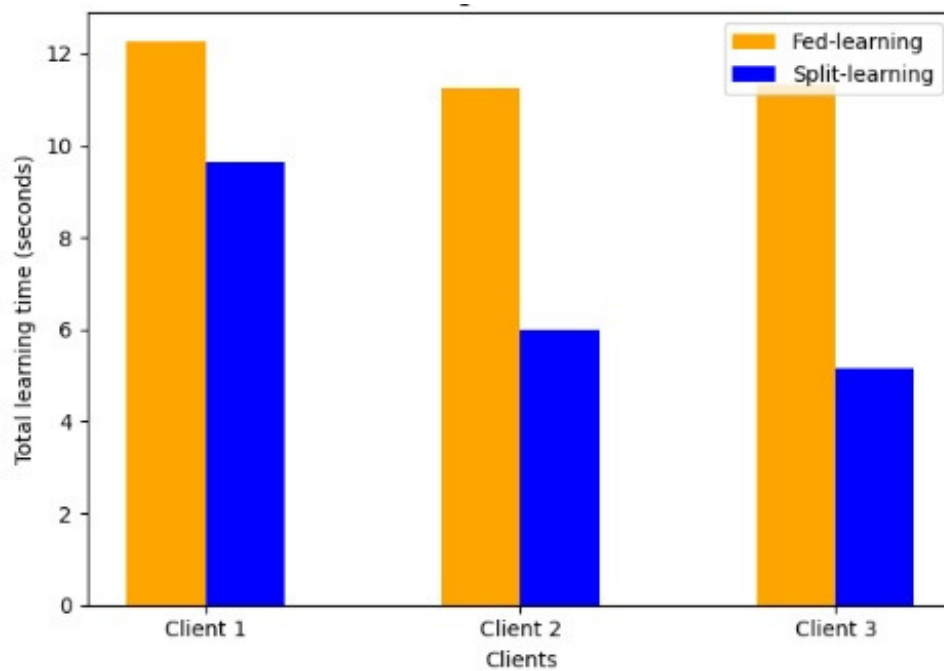


Figure 5.11: Total learning time for each client in FML and SML

Consequently, SML emerges as a more efficient and suitable approach for resource-constrained CPS deployments. Its ability to optimize computational resource utilization makes it a compelling alternative to FML in such environments. By effectively addressing the challenges associated with limited computational resources, SML showcases its potential to enhance the performance and adaptability of CPS systems operating under resource constraints.

However, it is important to acknowledge some potential limitations of the SML approach. Firstly, like in FML, SML relies on a central server for model aggregation, which could become a single point of failure. Implementing fault-tolerant mechanisms or exploring decentralized alternatives could mitigate this risk. Secondly, despite reducing communication overhead compared to centralized approaches, SML still requires transmitting model updates. Optimizing this further or managing it in networks with limited bandwidth remains an area for improvement. Furthermore, differences in local data distributions and training processes could lead to model inconsistencies, highlighting the importance of ensuring convergence and model stability across different clients. Moreover, the overall performance of SML can still be affected by network latency, especially in environments with poor connectivity. Strategies to minimize the impact of network delays should be explored.

5 Conclusion

In this chapter, our primary objective was to validate the effectiveness of our approach through the implementation of the fault prediction module in two distinct scenarios: centralized and decentralized architecture. The chapter began with a presentation of the fault prediction module structure, providing detailed insights into its components and functionalities. This was followed by an in-depth exploration of the implementation using centralized architecture, encompassing data preparation, ML algorithm selection, and deployment of the trained model. Subsequently, the chapter delved into the implementation using decentralized architecture, outlining the data preparation and training processes. Finally, the chapter concluded with a comprehensive comparison of the results obtained from both implementations, offering valuable insights into the performance and effectiveness of each architecture in the context of fault prediction within CPS.

The findings of this chapter demonstrate the feasibility and efficacy of the proposed approach for fault prediction in CPS. Both centralized and decentralized architectures exhibited promising results in predicting and anticipating potential problems before they lead to device failure. However, the decentralized architecture (FML and SML) offered several advantages over the centralized approach, including reduced communication overhead and enhanced data privacy. By distributing the data across multiple nodes, the decentralized approach eliminates

the need to store sensitive data in a centralized location, thereby reducing the risk of data breaches and unauthorized access. Additionally, the SML approach further enhances the decentralized architecture by efficiently managing resource constraints in CPS, partitioning data and computation tasks to ensure optimal use of limited resources while maintaining data privacy and security.

Conclusion

1 Research summary

In this thesis, we have explored the dynamic and complex realm of CPS and the critical need for adaptability in such systems. The integration of ML techniques has been proposed as a powerful solution to monitor and adapt CPS behavior in response to changing environmental conditions. This endeavor addresses the challenges posed by the highly dynamic nature of CPS environments, where pre-planned design-time adaptations may fall short.

Our thesis presents a series of logically interlinked contributions that lay the foundation for the development of more resilient, secure, and intelligent CPS. Firstly, we introduce a comprehensive meta-model for adaptive CPS, providing a conceptual framework for understanding and modeling the intricate interactions between physical and digital components within these systems. This meta-model serves as a valuable tool for CPS designers and developers, enabling them to effectively capture and represent the complexities of adaptive CPS behavior.

Secondly, we develop a centralized ML-based architecture, incorporating advanced techniques such as ML and Cloud/Edge Computing to enhance CPS functionalities. This architecture provides a robust platform for CPS applications, enabling them to effectively process vast amounts of data, extract meaningful insights, and make intelligent decisions in real-time. The utilization of ML algorithms empowers CPS with the ability to continuously learn and adapt, ensuring optimal performance and reliability in the face of changing conditions.

Following this, we propose a Privacy-Enhanced Architecture empowered by FML. This architecture addresses scalability challenges and enhances security and data privacy by distributing ML computations across multiple nodes. By federating ML training and inference, we achieve a balance between global performance and local data privacy, thus mitigating risks associated with centralized data storage. Building on these contributions, our final contribution introduces a novel SML approach to tackle resource constraints in CPS. This architecture optimizes resource usage while preserving data privacy and security. By efficiently processing and collaborating in environments with limited computational resources, SML provides a viable solution for maintaining functionality and performance under resource constraints.

To validate these contributions, a case study on fault prediction in industrial CPS environments has been conducted. The fault prediction module, implemented using both centralized, FML, and SML approaches, offers insights into the performance and suitability of each architecture.

2 Future work

As future work, we envision several directions for research and development:

- A crucial avenue is to combine FML and SML within the same architecture. Integrating these approaches could leverage the strengths of both techniques, offering enhanced adaptability, scalability, and efficiency for CPS applications.
- A comprehensive comparison of our architecture with other federated learning techniques, including FedProx and FedNova, is also planned. This analysis will provide valuable insights into the relative strengths and weaknesses of each technique, particularly in the context of enhancing CPS adaptability.
- We intend to conduct experiments with a larger set of clients, explore varying data distributions, and assess the scalability of our architecture. These efforts will contribute to a more thorough understanding of its robustness and suitability for real-world, large-scale CPS deployments.
- We will focus on the transition from controlled experimental settings to real-world CPS deployments. By assessing the performance and adaptability of our architecture in practical scenarios, we will validate its efficacy and its potential to address the evolving challenges of dynamic environments.

Annex 1 : Implementation Details of the Centralized Architecture

This annex provides an in-depth exploration of the implementation details behind the centralized architecture designed for adaptive CPS. The following sections delve into the specific libraries utilized, the software environment, and the hardware setup that collectively contributed to the successful development of our architecture.

1 Libraries Used for Implementation

The successful implementation of the centralized architecture for adaptive CPS relied on a robust selection of libraries that streamlined various aspects of the development process. Each library played a distinct role in enhancing the functionality, efficiency, and visualization capabilities of the architecture. Here, we provide an in-depth overview of the key libraries employed:

1.1 Scikit-learn

Scikit-learn [97], a machine learning library in Python, served as the cornerstone for implementing ML algorithms within the centralized architecture. It provided a versatile set of tools for data preprocessing, classification, regression, clustering, and more. Its simplicity and extensibility made it an ideal choice for seamlessly integrating machine learning functionalities into the CPS environment.

1.2 Pandas

Pandas [114], a powerful data manipulation and analysis library, played a pivotal role in handling structured data within the architecture. Its DataFrame structure facilitated efficient data organization, cleaning, and preprocessing. Pandas streamlined the process of preparing data for machine learning tasks, contributing to the overall adaptability of the CPS.

1.3 NumPy

NumPy [113], a fundamental package for scientific computing in Python, provided support for large, multi-dimensional arrays and matrices. In the context of the centralized architecture, NumPy's numerical operations and array manipulations were instrumental for efficient data processing, enabling seamless integration with ML algorithms.

1.4 Matplotlib

Matplotlib [112], a comprehensive 2D plotting library, was employed for creating static, animated, and interactive visualizations. It played a crucial role in visually representing data patterns, model predictions, and other insights. Matplotlib enhanced the interpretability of the architecture's outputs, contributing to better decision-making in dynamic CPS scenarios.

1.5 Seaborn

Seaborn [115], a statistical data visualization library based on Matplotlib, complemented the visualization capabilities of the architecture. Its high-level interface simplified the creation of informative and attractive statistical graphics. Seaborn, in conjunction with Matplotlib, facilitated the generation of clear and insightful visual representations of complex data structures within the CPS environment.

2 Work Environment

2.1 Software Environment

The software environment plays a crucial role in the successful implementation of the centralized architecture for adaptive CPS. The following components were utilized to ensure a robust and efficient development process:

- **Anaconda :** Anaconda [106] is a powerful open-source distribution, was employed to manage and deploy the Python environment. Anaconda facilitates seamless integration of various libraries and tools essential for machine learning and data analysis, providing a convenient platform for development.
- **Jupyter Notebook :** Jupyter Notebook [111], an interactive web-based application, served as the primary interface for developing, documenting, and executing code. Its versatility and integration capabilities with Anaconda made it an ideal choice for creating and sharing live code, equations, visualizations, and narrative text in a collaborative environment.

2.2 Hardware Environment

For the implementation of the centralized architecture, a DELL laptop with the following specifications was used:

- Type: DELL Laptop
- Processor: Intel® Core™ i5-4200U CPU@ 2.50GHz 2.50GHz
- RAM: 4.00 GB
- Operating System: Windows 10, 64-bit
- Graphics Card: Intel ® HD Graphics 5500, Intel ® HD Graphics 4600
- Network Card: Realtek PCIe FE Family Controller

Annex 2 : Implementation Details of the Decentralized Architecture

Annex 2 outlines the intricate details of implementing the decentralized architecture. It also highlights the exploration of various platforms for Federated Machine Learning (FML) solutions, leading to a comparative study to identify the most suitable platform for integration with our FML-based architecture.

1 FML-Based Platforms

These platforms serve as the foundation for our exploration and comparative study, guiding the selection of the most appropriate FML solution for integration with our decentralized architecture.

- **TensorFlow Federated [116]** : TensorFlow Federated (TFF) is an open-source framework developed by Google that facilitates the implementation of Federated Learning (FL) solutions. It provides tools for training machine learning models across decentralized devices, making it a prominent choice for FML applications.
- **PySyft [82]** : PySyft is a flexible and extensible library for federated and privacy-preserving machine learning, built on PyTorch and TensorFlow. PySyft allows for the development of privacy-focused applications, making it a valuable platform for FML solutions.
- **IBM Federated Learning [107]** : IBM Federated Learning is a platform provided by IBM that focuses on enabling collaborative machine learning across distributed devices. It offers a set of tools and services for building and deploying FML solutions in various domains.
- **Intel Open Federated Learning [110]** : Intel Open Federated Learning is an initiative by Intel to advance federated learning technologies. It leverages Intel's hardware capabilities to optimize FL solutions, making it suitable for scenarios where performance and scalability are crucial.
- **Flower [36]** : Flower is an open-source Python library for building federated learning systems. It acts as a framework-agnostic extension to existing machine learning frameworks, facilitating the development of federated learning applications.

2 Comparative Studies of Collaborative-Based Platforms

It is crucial to note that each collaborative learning platform comes with its own set of criteria. Evaluating each platform is essential to determine which one best suits specific needs. Several comparison criteria exist for Federated Learning Frameworks, including:

- **Security:** Collaborative Learning frameworks must ensure complete data security, guaranteeing user data confidentiality at all times.
- **Communication Efficiency:** Collaborative Learning frameworks should effectively manage communication between devices to minimize delays and interruptions.
- **Scalability:** Collaborative Learning frameworks must handle thousands of clients for large-scale usage.
- **Complexity:** The relative complexity of each framework can influence ease of use and maintenance. Frameworks with simpler APIs may facilitate quicker deployment and adoption.
- **Performance:** Collaborative Learning frameworks should maintain high performance even under significant bandwidth pressure.
- **Free Platform:** A free platform refers to an online platform or service that is accessible without charges for users. This implies no payment is required for using the platform or accessing its basic features.
- **Cloud Usage:** Cloud services are computing services provided over the Internet by cloud service providers. These services allow users to access computing resources, applications, and data on-demand without the need for their physical infrastructure.

These criteria serve as the basis for our comparative study, guiding the selection of the most suitable federated learning platform for integration with our architecture. Table 5.6 presents a comparison of Collaborative ML platforms. The symbol ✓ indicates the presence of the corresponding feature in each platform.

In our case, Google TFF stands out as a strong candidate due to its comprehensive support for security, communication efficiency, scalability, and performance. TFF provides a robust framework for implementing federated learning, aligning well with the requirements of our centralized architecture. The extensive capabilities of TFF, including its security features and

Platform	Security	Communication	Efficiency	Scalability	Complexity	Performance	Free	Cloud Usage
TensorFlow Federated	✓	✓		✓	✓	✓	✓	✓
PySyft	✓	✓		✓		✓	✓	
IBM FL	✓	✓		✓		✓		✓
Intel Open FL	✓				✓	✓		✓
Flower	✓				✓		✓	

Table 5.6: Comparison of FML Platforms

efficient communication protocols, make it a suitable choice for ensuring the success of our FML solution. Additionally, TFF offers Cloud usage and is free, further enhancing its appeal for our implementation.

3 Implementation of the FML-based architecture with TFF

In this section, we explain the experimental setup we used to implement our FML-based architecture using the TFF platform. TFF is used for parallel client training, and the training process involves data batch division and distribution to individual clients. The federated model is constructed based on the same architecture as the global model and is trained using the Federated Averaging (FedAvg) algorithm [71].

As we mentioned in chapter 4, the training process centers on a collaborative training task involving both Edge Collectors and a Cloud layer. The goal is to update a global model by utilizing the local models trained on each client's local data. The first step of the training process is the initialization of the global model. This step sets the model architecture and parameter values before engaging in collaborative training. It involves setting up the architecture of the learning model (specifying the number and types of layers, activation functions, and other architectural components). In our case, we define the architecture of the global model using TensorFlow's Keras API within TFF's context. We use the `tff.learning.from_keras_model` function to convert the Keras model into a TFF-compatible federated learning model.

For clients selection, we use the TFF's built-in federated computations and functions. The `tff.federated` module provides functionalities for creating federated computations, including the selection of clients for each round of federated training. In our architecture, the `select_clients` function is defined as a federated computation using the `@tff.federated_computation_decorator`. The function takes `client_weights` as an argument, which could be a federated integer type representing the weights of each client.

Regarding the step of local training (client update) we define the function `local_training` as follow: `local_training(global_model, data_batch)`. The function takes as parameters the `global_model` as the model architecture and a `data_batch` representing the local data on the client. This function allows as initializing a copy from the global model using `tff.learning.ModelWeights.from_model`. Local training is performed using TFF's gradient tape mechanism. The loss is computed based on the model's predictions and actual labels, and gradients are calculated. After executing the federated computation, the updated local model will contain the local model with updated parameters after the local training process. The resulting local model updates are sent to the Cloud layer.

For model aggregation we have used the Federated Averaging (FedAvg) algorithm. It permits aggregation local model updates from clients to update the global model. In our case, we define the function *models_aggregation* as a federated computation using the `@tff.federated_computation` decorator as follow: *models_aggregation(global_model, local_models)*. The function takes as an input the global model and a list of local models representing the updated models from selected clients. In this function, the predefined *with_weights* method is used to attach the client weights to each local model. Also, the predefined *tff.federated_mean* method is applied to compute the weighted average of the local models. This function calculates the weighted sum of model weights and divides it by the total weight to obtain the aggregated weights.

After the global model has been updated through the aggregation process, it needs to be broadcasted to all participating clients. This synchronization guarantees that each client has access to the most up-to-date global model for the subsequent round of training.

In this experimentation, the global DL network is formulated as a Sequential model. Its architecture consists of two densely connected layers with ReLU (Rectified Linear Unit) activation functions, followed by a final layer with a sigmoid activation function. The global DL network is trained using the FedAvg algorithm, where model updates from local clients are aggregated on the Cloud layer. Each training round consists of a set number of epochs (`num_epochs = 70`), with a specified batch size (`batch_size = 32`) for processing data. Additionally, the entire training process spans a predetermined number of rounds (`num_rounds = 80`).

4 Vanilla SML Implementation with TensorFlow Keras

Central to the SML paradigm are the client models, which encapsulate domain-specific knowledge while training on local datasets. Utilizing the TensorFlow Keras API [33], we instantiate client models with architectures tailored to accommodate the dimensions and characteristics of local data. Each client model comprises multiple densely connected layers, incorporating activation functions such as ReLU to introduce non-linearity and facilitate model convergence. In our case, client models are created using the `create_client_model()` function, which defines the architecture and compiles the models.

Client models are trained iteratively across multiple epochs using the `fit()` function from the TensorFlow Keras API, with each epoch encompassing forward and backward propagation facilitated by the `train_on_batch()` method. This method updates model parameters using stochastic gradient descent. Training parameters such as batch size and learning rate are meticulously tuned using functions such as `compile()` and `fit()` to optimize convergence while

mitigating computational overhead. In our case, the `compile()` function configures the model for training, specifying the Adam optimizer for efficient gradient descent and the binary cross-entropy loss function. This loss function quantifies the difference between model predictions and ground truth labels during binary classification tasks.

During training, client models exclusively access and learn from their respective local datasets, preserving data privacy and confidentiality. This is facilitated by functions such as `fit()`, which train the model on local data without sharing it externally. Following the client model construction, a central server model is also created, adhering to the principles of Vanilla SML. We utilize the `Sequential()` constructor from the TensorFlow Keras API to define the server model's architecture. This architecture is designed to be compatible with the client models, using the function `Dense()` to add densely connected layers. The server model acts as the central nexus for knowledge aggregation within the SML paradigm. During the training process, client models train locally and extract relevant information (e.g., gradients) after a designated cut layer. These are then sent to the server for aggregation. In our case, the server uses the functions `get_weights()` and `set_weights()` to retrieve these gradients and incorporate them into its own model, facilitating the collaborative learning process. This iterative exchange of gradients between clients and the server continues until a convergence criterion is met.

Following construction, the server model is compiled using the `compile()` function to prepare for training and knowledge aggregation, entailing specifying optimization algorithms, loss functions, and optional evaluation metrics. In our implementation, the Adam optimizer is employed for efficient gradient descent optimization, while binary cross-entropy serves as the loss function for binary classification tasks. Evaluation metrics, including accuracy, provide insights into model performance and convergence during training and aggregation phases, facilitated by functions such as `evaluate()`.

Publications During the Thesis

During the course of this doctoral research, the following publications have been produced, reflecting the advancements and contributions made in the fields of Cyber-Physical Systems and Machine Learning :

- **Conference Papers:**

- Azeri Nabila, Hioual Ouassila, and Hioual Ouided. "Towards an Approach for Modeling and Architecting of Self-Adaptive Cyber-Physical Systems." In *2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, 1-7, IEEE, 2022. [[12](#)]
- Azeri Nabila, Zouikri Zeinb, Rezgui Meriem, Hioual Ouided, and Hioual Ouassila. "Fault Prediction Using Supervised and Unsupervised Learning Algorithms in Cyber Physical Systems." In *2022 2nd International Conference on New Technologies of Information and Communication (NTIC)*, 1-6, IEEE, 2022. [[15](#)]
- Azeri Nabila, Hioual Ouided, and Hioual Ouassila. "Enhancing Self-Adaptive Cyber-Physical Systems using Federated Machine Learning." In *TACC 2023: 3rd Tunisian-Algerian Joint Conference on Applied Computing*, 108-119, ceur-ws.org, 2023. [[14](#)]

- **Journal Paper:**

- Azeri Nabila, Hioual Ouided, and Hioual Ouassila. "Efficient Vanilla Split Learning for Privacy-Preserving Collaboration in Resource-Constrained Cyber-Physical Systems." *Informatica*, 48(11), 167-180, 2024.[[13](#)]

Bibliography

- [1] Nadeem Abbas, Jesper Andersson, and Danny Weyns. Asple: A methodology to develop self-adaptive software systems with systematic reuse. *Journal of Systems and Software*, 167:110626, 2020.
- [2] Kazeem B Adedeji and Yskandar Hamam. Cyber-physical systems for water supply network management: Basics, challenges, and roadmap. *Sustainability*, 12(22):9555, 2020.
- [3] Rania Salih Ahmed, Elmustafa Sayed Ali Ahmed, and Rashid A Saeed. Machine learning in cyber-physical systems in industry 4.0. In *Artificial intelligence paradigms for smart cyber-physical systems*, pages 20–41. IGI global, 2021.
- [4] Syed Hassan Ahmed, Gwanghyeon Kim, and Dongkyun Kim. Cyber physical system: Architecture, applications and research challenges. *2013 IFIP Wireless Days (WD)*, pages 1–5, 2013.
- [5] Mohammad Al Faruque, Francesco Regazzoni, and Miroslav Pajic. Design methodologies for securing cyber-physical systems. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pages 30–36. IEEE, 2015.
- [6] Fahd AlDosari. Security and privacy challenges in cyber-physical systems. *Journal of Information Security*, 8(4):285–295, 2017.
- [7] Rasim Alguliyev, Yadigar Imamverdiyev, and Lyudmila Sukhostat. Cyber-physical systems and their security issues. *Computers in Industry*, 100:212–223, 2018.
- [8] Nazakat Ali, Manzoor Hussain, and Jang-Eui Hong. Fault-tolerance by resilient state transition for collaborative cyber-physical systems. *Mathematics*, 9(22):2851, 2021.
- [9] Odey Alshboul, Rabia Emhamed Al Mamlook, Ali Shehadeh, and Tahir Munir. Empirical exploration of predictive maintenance in concrete manufacturing: Harnessing machine learning for enhanced equipment reliability in construction project management. *Computers & Industrial Engineering*, page 110046, 2024.
- [10] Oroos Arshi and Surajit Mondal. Advancements in sensors and actuators technologies for smart cities: a comprehensive review. *Smart Construction and Sustainable Cities*, 1(1):18, 2023.

- [11] Mariette Awad, Rahul Khanna, Mariette Awad, and Rahul Khanna. Machine learning in action: Examples. *Efficient learning machines: Theories, concepts, and applications for engineers and system designers*, pages 209–240, 2015.
- [12] Nabila Azeri, Ouassila Hioual, and Ouided Hioual. Towards an approach for modeling and architecting of self-adaptive cyber-physical systems. In *2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–7. IEEE, 2022.
- [13] Nabila Azeri, Ouassila Hioual, and Ouided Hioual. Efficient vanilla split learning for privacy-preserving collaboration in resource-constrained cyber-physical systems. *Informatica*, 48(11):167–180, 2024.
- [14] Nabila Azeri, Ouided Hioual, and Ouassila Hioual. Enhancing self-adaptive cyber-physical systems using federated machine learning. In *TACC 2023 : 3rd Tunisian-Algerian Joint Conference on Applied Computing*, pages 108–119. ceur-ws.org, 2023.
- [15] Nabila Azeri, Zeinb Zouikri, Meriem Rezgui, Ouided Hioual, and Ouassila Hioual. Fault prediction using supervised and unsupervised learning algorithms in cyber physical systems. In *2022 2nd International Conference on New Technologies of Information and Communication (NTIC)*, pages 1–6. IEEE, 2022.
- [16] Sami Babouche, Samir Ouchani, Mourad Zghal, et al. An adaptive attack prediction framework in cyber-physical systems. In *2022 Ninth International Conference on Software Defined Systems (SDS)*, pages 1–7. IEEE, 2022.
- [17] Martin Bauer, Nicola Bui, Jourik De Loof, Carsten Magerkurth, Andreas Nettsträter, Julinda Stefa, and Joachim W Walewski. Iot reference model. *Enabling Things to Talk: Designing IoT solutions with the IoT architectural reference model*, pages 113–162, 2013.
- [18] Robert Martin Brett Murphy, Shi-Wan Lin. The industrial internet reference architecture, 2022. <https://www.iiconsortium.org/wp-content/uploads/sites/2/2022/11/IIRA-v1.10.pdf>, Accessed: January 30, 2024.
- [19] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi Müller, Mauro Pezzè, and Mary Shaw. Engineering self-adaptive systems through feedback loops. *Software engineering for self-adaptive systems*, pages 48–70, 2009.

- [20] Jesus Castillo Cabello, Hadis Karimipour, Amir Namavar Jahromi, Ali Dehghantanha, and Reza M Parizi. Big-data and cyber-physical systems in healthcare: Challenges and opportunities. *Handbook of Big Data Privacy*, pages 255–283, 2020.
- [21] Thyago P Carvalho, Fabrizzio AAMN Soares, Roberto Vita, Roberto da P Francisco, João P Basto, and Symone GS Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, 2019.
- [22] Betty HC Cheng, Kerstin I Eder, Martin Gogolla, Lars Grunske, Marin Litoiu, Hausi A Müller, Patrizio Pelliccione, Anna Perini, Nauman A Qureshi, Bernhard Rumpe, et al. Using models at runtime to address assurance for self-adaptive systems. *Models@ run. time: foundations, applications, and roadmaps*, pages 101–136, 2014.
- [23] Sunil Kumar Chinnamgari. *R Machine Learning Projects: Implement supervised, unsupervised, and reinforcement learning techniques using R 3.5*. Packt Publishing Ltd, 2019.
- [24] Autonomic Computing et al. An architectural blueprint for autonomic computing. *IBM White Paper*, 31(2006):1–6, 2006.
- [25] Baudouin Dafflon, Nejib Moalla, and Yacine Ouzrout. The challenges, approaches, and used techniques of cps for manufacturing in industry 4.0: a literature review. *The International Journal of Advanced Manufacturing Technology*, 113:2395–2412, 2021.
- [26] Pablo Felipe Soares De Melo and Eduardo Paciencia Godoy. Controller interface for industry 4.0 based on rami 4.0 and opc ua. In *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT)*, pages 229–234. Ieee, 2019.
- [27] Kakan Dey, Ryan Fries, and Shofiq Ahmed. Future of transportation cyber-physical systems—smart cities/regions. *Transportation cyber-physical systems*, pages 267–307, 2018.
- [28] Javier Diaz-Rozo, Concha Bielza, and Pedro Larrañaga. Machine learning-based cps for clustering high throughput machining cycle conditions. *Procedia Manufacturing*, 10:997–1008, 2017.
- [29] Seyed Mehran Dibaji, Alefiya Hussain, and Hideaki Ishii. A tutorial on security and privacy challenges in cps. *Security and Resilience of Control Systems: Theory and Applications*, pages 121–146, 2022.

- [30] Delia Ioana Dogaru and Ioan Dumitrache. Cyber-physical systems in healthcare networks. In *2015 E-Health and Bioengineering Conference (EHB)*, pages 1–4. IEEE, 2015.
- [31] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach. *Advances in Neural Information Processing Systems*, 33:3557–3568, 2020.
- [32] National Science Foundation. Cyber-physical systems (cps), 2022. <https://www.nsf.gov/pubs/2021/nsf21551/nsf21551.htm>, Accessed: November 27, 2023.
- [33] Multi framework deep learning API. Keras: The high-level api for tensorflow, 2023. <https://www.tensorflow.org/guide/keras>.
- [34] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, 2018.
- [35] Ilias Gerostathopoulos, Dominik Skoda, Frantisek Plasil, Tomas Bures, and Alessia Knauss. Tuning self-adaptation in cyber-physical systems through architectural homeostasis. *Journal of Systems and Software*, 148:37–55, 2019.
- [36] Flowers Labs GmbH. Flower a friendly federated learning framework, 2023. <https://flower.dev/>.
- [37] Alexander Graß, Christian Beecks, and Jose Angel Carvajal Soto. Unsupervised anomaly detection in production lines. In *Machine Learning for Cyber Physical Systems: Selected papers from the International Conference MLACPS 2018*, pages 18–25. Springer, 2019.
- [38] Edward R Griffor, Christopher Greer, David A Wollman, and Martin J Burns. Framework for cyber-physical systems: Volume 2, working group reports. 2017.
- [39] Achim Guldner, Maximilian Hoffmann, Christian Lohr, Rüdiger Machhamer, Lukas Malburg, Marlies Morgen, Stephanie C Rodermund, Florian Schäfer, Lars Schaupter, Jens Schneider, et al. A framework for ai-based self-adaptive cyber-physical process systems. *it-Information Technology*, (0), 2023.
- [40] Volkan Gunes, Steffen Peter, Tony Givargis, and Frank Vahid. A survey on concepts, applications, and challenges in cyber-physical systems. *KSII Transactions on Internet and Information Systems (TIIS)*, 8(12):4242–4268, 2014.

- [41] Zhiwei Guo, Keping Yu, Zhihan Lv, Kim-Kwang Raymond Choo, Peng Shi, and Joel JPC Rodrigues. Deep federated learning enhanced secure poi microservices for cyber-physical systems. *IEEE Wireless Communications*, 29(2):22–29, 2022.
- [42] Manoj Kr Gupta and Pravin Chandra. A comparative study of clustering algorithms. In *2019 6th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 801–805. IEEE, 2019.
- [43] Mohammad Reza Habibi, Hamid Reza Baghaee, Tomislav Dragičević, and Frede Blaabjerg. Detection of false data injection cyber-attacks in dc microgrids based on recurrent neural networks. *IEEE Journal of Emerging and Selected Topics in Power Electronics*, 9(5):5294–5310, 2020.
- [44] Sara M Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaella Mirandola, and Diego Perez-Palacin. Uncertainty in self-adaptive systems: A research community perspective. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 15(4):1–36, 2021.
- [45] Johannes Iber, Tobias Rauter, Michael Krisper, and Christian Kreiner. The potential of self-adaptive software systems in industrial control systems. In *Systems, Software and Services Process Improvement: 24th European Conference, EuroSPI 2017, Ostrava, Czech Republic, September 6–8, 2017, Proceedings 24*, pages 150–161. Springer, 2017.
- [46] Bruno JD Jacobs, Bas Donkers, and Dennis Fok. Model-based purchase predictions for large assortments. *Marketing Science*, 35(3):389–404, 2016.
- [47] Amir Namavar Jahromi, Hadis Karimipour, Ali Dehghantanha, and Kim-Kwang Raymond Choo. Toward detection and attribution of cyber-attacks in iot-enabled cyber-physical systems. *IEEE Internet of Things Journal*, 8(17):13712–13722, 2021.
- [48] Juliza Jamaludin and Jemmy Mohd Rohani. Cyber-physical system (cps): State of the art. In *2018 International Conference on Computing, Electronic and Electrical Engineering (ICE Cube)*, pages 1–5. IEEE, 2018.
- [49] Václav Jirkovský, Marek Obitko, and Vladimír Mařík. Understanding data heterogeneity in the context of cyber-physical systems integration. *IEEE Transactions on Industrial Informatics*, 13(2):660–667, 2016.

- [50] Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.
- [51] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*, pages 5132–5143. PMLR, 2020.
- [52] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
- [53] Marwa Keshk, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Privacy-preserving big data analytics for cyber-physical systems. *Wireless Networks*, pages 1–9, 2022.
- [54] Muhammad Umer Khan, Shuai Li, Qixin Wang, and Zili Shao. Cps oriented control design for networked surveillance robots with multiple physical constraints. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(5):778–791, 2016.
- [55] Tanveer Khan, Khoa Nguyen, and Antonis Michalas. Split ways: Privacy-preserving training of encrypted data using split learning. *arXiv preprint arXiv:2301.08778*, 2023.
- [56] Sangjun Kim and Kyung-Joon Park. A survey on machine-learning based security design for cyber-physical systems. *Applied Sciences*, 11(12):5458, 2021.
- [57] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.
- [58] Jakub Konečný, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.
- [59] Sotiris B Kotsiantis, Ioannis Zaharakis, P Pintelas, et al. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24, 2007.

- [60] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17:184–206, 2015.
- [61] Sungmoon Kwon, Hyunguk Yoo, and Taeshik Shon. Ieee 1815.1-based power system security with bidirectional rnn-based network anomalous attack detection for cyber-physical system. *IEEE Access*, 8:77572–77586, 2020.
- [62] Veronika Lesch, Marwin Züfle, André Bauer, Lukas Iffländer, Christian Krupitzer, and Samuel Kounev. A literature review of iot and cps—what they are, and what they are not. *Journal of Systems and Software*, 200:111631, 2023.
- [63] Jian-Qiang Li, F Richard Yu, Genqiang Deng, Chengwen Luo, Zhong Ming, and Qiao Yan. Industrial internet: A survey on the enabling technologies, applications, and challenges. *IEEE Communications Surveys & Tutorials*, 19(3):1504–1526, 2017.
- [64] Qinbin Li, Bingsheng He, and Dawn Song. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10713–10722, 2021.
- [65] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.
- [66] Shi-Wan Lin, Brett Murphy, Erich Clauer, Ulrich Loewen, Ralf Neubert, Gerd Bachmann, Madhusudan Pai, and Martin Hankel. Architecture alignment and interoperability: An industrial internet consortium and plattform industrie 4.0 joint whitepaper. *White Paper, Industrial Internet Consortium*, 2017.
- [67] Ji Liu, Jizhou Huang, Yang Zhou, Xuhong Li, Shilei Ji, Haoyi Xiong, and Dejing Dou. From distributed machine learning to federated learning: A survey. *Knowledge and Information Systems*, 64(4):885–917, 2022.
- [68] Sin Kit Lo, Qinghua Lu, Chen Wang, Hye-Young Paik, and Liming Zhu. A systematic literature review on federated machine learning: From a software engineering perspective. *ACM Computing Surveys (CSUR)*, 54(5):1–39, 2021.
- [69] Tiziana Margaria, Hafiz Ahmad Awais Chaudhary, Ivan Guevara, Stephen Ryan, and Alexander Schieweck. The interoperability challenge: building a model-driven digital

- thread platform for cps. In *International Symposium on Leveraging Applications of Formal Methods*, pages 393–413. Springer, 2021.
- [70] Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 third international conference on artificial intelligence for industries (ai4i)*, pages 69–74. IEEE, 2020.
- [71] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.
- [72] Fatima Zohra Merabet and Djamel Benmerzoug. Qos prediction for service selection and recommendation with a deep latent features autoencoder. *Computer Science and Information Systems*, 19(2):709–733, 2022.
- [73] Tom M Mitchell. *Machine learning*, 1997.
- [74] Khaled Salah Mohamed and Khaled Salah Mohamed. Iot physical layer: Sensors, actuators, controllers and programming. *The Era of Internet of Things: Towards a Smart World*, pages 21–47, 2019.
- [75] Hossein Mohammadi Rouzbahani, Hadis Karimipour, Abolfazl Rahimnejad, Ali Dehghantanha, and Gautam Srivastava. Anomaly detection in cyber-physical systems using machine learning. *Handbook of big data privacy*, pages 219–235, 2020.
- [76] Dietmar PF Möller and Hamid Vakilzadian. Cyber-physical systems in smart transportation. In *2016 IEEE international conference on electro information technology (EIT)*, pages 0776–0781. IEEE, 2016.
- [77] Paula Monteiro, Marcia Carvalho, Francisco Morais, Monica Melo, Ricardo J Machado, and Fernando Pereira. Adoption of architecture reference models for industrial information management systems. In *2018 international conference on intelligent systems (IS)*, pages 763–770. IEEE, 2018.
- [78] Henry Muccini, Mohammad Sharaf, and Danny Weyns. Self-adaptation for cyber-physical systems: a systematic literature review. In *Proceedings of the 11th international symposium on software engineering for adaptive and self-managing systems*, pages 75–81, 2016.

- [79] Omprakash Nayak, Jaykumar Lachure, and Rajesh Doriya. Fog enabled cyber-physical attack detection using ensemble machine learning. In *2022 1st International Conference on Sustainable Technology for Power and Energy Systems (STPES)*, pages 1–6. IEEE, 2022.
- [80] Haifeng Niu and S Jagannathan. Optimal defense and control of dynamic systems modeled as cyber-physical systems. *The Journal of Defense Modeling and Simulation*, 12(4):423–438, 2015.
- [81] Felix O Olowononi, Danda B Rawat, and Chunmei Liu. Resilient machine learning for networked cyber physical systems: A survey for machine learning security to securing machine learning for cps. *IEEE Communications Surveys & Tutorials*, 23(1):524–552, 2020.
- [82] OpenMined. Tpsyft, 2023. <https://blog.openmined.org/tag/pysyft/>.
- [83] Outsystems. What is machine learning and how does it work?, 2023. <https://www.outsystems.com/glossary/what-is-machine-learning/>, Accessed: December 28, 2023.
- [84] Federica Pascucci. Fault tolerance in cyber-physical systems. In *Encyclopedia of Cryptography, Security and Privacy*, pages 1–3. Springer, 2023.
- [85] Diego GS Pivoto, Luiz FF de Almeida, Rodrigo da Rosa Righi, Joel JPC Rodrigues, Alexandre Baratella Lugli, and Antonio M Alberti. Cyber-physical systems architectures for industrial internet of things applications in industry 4.0: A literature review. *Journal of manufacturing systems*, 58:176–192, 2021.
- [86] Pian Qi, Diletta Chiaro, Antonella Guzzo, Michele Ianni, Giancarlo Fortino, and Francesco Piccialli. Model aggregation techniques in federated learning: A comprehensive survey. *Future Generation Computer Systems*, 2023.
- [87] Leonardo Ramos, Gabriel Lisbôa Guimarães Divino, Guilherme Cano Lopes, Breno Bernard Nicolau de França, Leonardo Montecchi, and Esther Luna Colombini. The rocs framework to support the development of autonomous robots. *Journal of Software Engineering Research and Development*, 7:10–1, 2019.
- [88] Matthias Rohr, Simon Giesecke, Marcel Hiel, Willem-Jan van den Heuvel, Hans Weigand, and Wilhelm Hasselbring. A classification scheme for self-adaptation research. 2006.

- [89] Mutaz Ryalat, Hisham ElMoaqet, and Marwa AlFaouri. Design of a smart factory based on cyber-physical systems and internet of things towards industry 4.0. *Applied Sciences*, 13(4):2156, 2023.
- [90] Luca Sabatucci, Valeria Seidita, and Massimo Cossentino. The four types of self-adaptive systems: a metamodel. In *Intelligent Interactive Multimedia Systems and Services 2017 10*, pages 440–450. Springer, 2018.
- [91] Shagan Sah. Machine learning: a review of learning types. 2020.
- [92] Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM transactions on autonomous and adaptive systems (TAAS)*, 4(2):1–42, 2009.
- [93] Tariq Samad. Human-in-the-loop control and cyber-physical-human systems: applications and categorization. *Cyber-Physical-Human Systems: Fundamentals and Applications*, pages 1–23, 2023.
- [94] Arthur L Samuel. Machine learning. *The Technology Review*, 62(1):42–45, 1959.
- [95] Arman Sargolzaei, Carl D Crane, Alireza Abbaspour, and Shirin Noei. A machine learning approach for fault detection in vehicular cyber-physical systems. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 636–640. IEEE, 2016.
- [96] Karsten Schweichhart. Reference architectural model industrie 4.0 (rami 4.0). *An Introduction*, 40, 2016.
- [97] scikit-learn development Team. scikit-learn : Machine learning in python, 2023. <https://scikit-learn.org/stable/>.
- [98] Deepti Sehrawat and Nasib Singh Gill. Smart sensors: Analysis of different types of iot sensors. In *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 523–528. IEEE, 2019.
- [99] Yesha Shastri. A step-by-step guide to federated learning in computer vision, 2023. <https://www.v7labs.com/blog/federated-learning-guide>, Accessed: January 27, 2024.

- [100] sigmaos. Internet of things (iot) terms explained: Cyber-physical system (cps), 2023. <https://sigmaos.com/tips/startups/internet-of-things-iot-terms-explained-cyber-physical-system-cps>, Accessed: November 28, 2023.
- [101] Coursera Staff. 3 types of machine learning you should know, 2023. <https://www.coursera.org/articles/types-of-machine-learning>, Accessed: December 28, 2023.
- [102] Javatpoint Staff. Classification algorithm in machine learning, 2023. <https://www.javatpoint.com/classification-algorithm-in-machine-learning>, Accessed: January 26, 2024.
- [103] Javatpoint Staff. Classification algorithm in machine learning, 2023. <https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning>, Accessed: January 26, 2024.
- [104] Javatpoint Staff. Unsupervised machine learning, 2023. <https://www.javatpoint.com/unsupervised-machine-learningg>, Accessed: January 26, 2024.
- [105] Jože Tavčar and Imre Horvath. A review of the principles of designing smart cyber-physical systems for run-time adaptation: Learned lessons and open issues. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1):145–158, 2018.
- [106] Anaconda Team. Anaconda : Data science ai platformn, 2023. <https://www.anaconda.com/>.
- [107] IBM Team. Ibm federated learning, 2023. <https://ibmfl.res.ibm.com/>.
- [108] IBM team. What is a machine learning pipeline?, 2024. <https://www.ibm.com/topics/machine-learning-pipeline>, Accessed: January 27, 2024.
- [109] IBM team. What is unsupervised learning?, 2024. <https://www.ibm.com/topics/unsupervised-learning>, Accessed: January 26, 2024.
- [110] Intel Team. Try federated learning with openfl, 2023. <https://www.intel.com/content/www/us/en/developer/articles/technical/try-federated-learning-with-openfl.html>.

- [111] Jupyter Team. Project jupyter documentation, 2023. <https://docs.jupyter.org/en/latest/>.
- [112] Matplotlib Team. Matplotlib: Visualization with python, 2023. <https://matplotlib.org/>.
- [113] NumPy Team. Numpy : The fundamental package for scientific computing with python, 2023. <https://numpy.org/>.
- [114] Pandas Team. Pandas : Python data analysis library, 2023. <https://pandas.pydata.org/>.
- [115] Seaborn Team. Seaborn: statistical data visualization, 2023. <https://seaborn.pydata.org/>.
- [116] TFF. Tensorflow federated: Machine learning on decentralized data, 2023. <https://www.tensorflow.org/federated>.
- [117] Yu-Chee Tseng, Meng-Shiuan Pan, and Yuen-Yung Tsai. Wireless sensor networks for emergency navigation. *Computer*, 39(7):55–62, 2006.
- [118] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018.
- [119] Omar Abdel Wahab, Azzam Mourad, Hadi Otrouk, and Tarik Taleb. Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems. *IEEE Communications Surveys & Tutorials*, 23(2):1342–1397, 2021.
- [120] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems*, 33:7611–7623, 2020.
- [121] Danny Weyns. *An introduction to self-adaptive systems: A contemporary software engineering perspective*. John Wiley & Sons, 2020.
- [122] Danny Weyns and Jesper Andersson. From self-adaptation to self-evolution leveraging the operational design domain. *arXiv preprint arXiv:2303.15260*, 2023.

- [123] Chathurika S Wickramasinghe, Kasun Amarasinghe, Daniel L Marino, Craig Rieger, and Milos Manic. Explainable unsupervised machine learning for cyber-physical systems. *IEEE Access*, 9:131824–131843, 2021.
- [124] Mingtao Wu, Zhengyi Song, and Young B Moon. Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods. *Journal of intelligent manufacturing*, 30:1111–1123, 2019.
- [125] Xiuge Wu, Tinghuai Ma, Jie Cao, Yuan Tian, and Alia Alabdulkarim. A comparative study of clustering ensemble algorithms. *Computers & Electrical Engineering*, 68:603–615, 2018.
- [126] Gang Xiong, Fenghua Zhu, Xiwei Liu, Xisong Dong, Wuling Huang, Songhang Chen, and Kai Zhao. Cyber-physical-social system in intelligent transportation. *IEEE/CAA Journal of Automatica Sinica*, 2(3):320–333, 2015.
- [127] Minrui Xu, Jialiang Peng, Brij B Gupta, Jiawen Kang, Zehui Xiong, Zhenni Li, and Ahmed A Abd El-Latif. Multiagent federated reinforcement learning for secure incentive mechanism in intelligent cyber-physical systems. *IEEE Internet of Things Journal*, 9(22):22095–22108, 2021.
- [128] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.
- [129] Chong Zhang, Xiao Liu, Xi Zheng, Rui Li, and Huai Liu. Fenghuolun: A federated learning based edge computing platform for cyber-physical systems. In *2020 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops)*, pages 1–4. IEEE, 2020.
- [130] Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. Understanding uncertainty in cyber-physical systems: a conceptual model. In *Modelling Foundations and Applications: 12th European Conference, ECMFA 2016, Held as Part of STAF 2016, Vienna, Austria, July 6-7, 2016, Proceedings 12*, pages 247–264. Springer, 2016.
- [131] Yin Zhang, Meikang Qiu, Chun-Wei Tsai, Mohammad Mehedi Hassan, and Atif Alamri. Health-cps: Healthcare cyber-physical system assisted by cloud and big data. *IEEE Systems Journal*, 11(1):88–95, 2015.