Intelligent Microservices-based Approach to Support Data Analytics for IoT Applications

Safa Ben Atitallah

UNIVERSITY OF MANOUBA
NATIONAL SCHOOL OF COMPUTER SCIENCE

PHD THESIS
For the degree of
Doctor in Computer Science

Intelligent Microservices-based Approach to Support Data Analytics for IoT Applications

---

By
Safa Ben Atitallah

Publicly defended on 10 July 2023, in the front of the jury composed of:

President: Imed Riadh FARAH, Professor, University of Manouba, Tunisia.
Examiner: Yassine JAMOUSI, Professor, Sultan Qaboos University, Oman.
Examiner: Faisal SAEED, Associate Professor, Birmingham City University, United Kingdom.
Examiner: Hanen IDOUDI, Associate Professor, University of Manouba, Tunisia.
Thesis Director: Henda BEN GHEZALA, Professor, University of Manouba, Tunisia.
Thesis Co-Director: Wadii BOULILA, Associate Professor, Prince Sultan University, Saudi Arabia.
Thesis Supervisor: Maha DRISS, Assistant Professor, Prince Sultan University, Saudi Arabia.

Abstract

The rapid development of the Internet of Things (IoT) has created a huge and complex network of interlinked equipment generating massive amounts of data. To efficiently analyze this data for a variety of applications, we propose an intelligent approach that utilizes multi-source and heterogeneous data from IoT devices.
Our approach incorporates centralized and distributed learning techniques and is implemented through a set of secure, flexible, and scalable microservices. We divide the data analytics functionality into containerized microservices and train analytics models using AI methods before deploying them on edge computing nodes. We also utilize transfer learning and ensemble learning techniques to enhance model generalization and prediction accuracy.
The adoption of a microservices-based architecture offers several advantages, including scalability, flexibility, reliability, modularity, and integration. This simplifies the development, implementation, and management of intricate machine learning and deep learning systems compared to conventional monolithic architectures.
Our approach has significant implications for the IoT industry, providing a powerful tool for unlocking valuable insights from IoT data. We validate our approach through a set of IoT case studies and comparative studies, demonstrating its effectiveness and practicality in real-world scenarios. Overall, our suggested approach offers a robust and effective solution for analyzing IoT data and deriving important insights that can be used in a wide range of use cases.

Keywords: Microservices architecture; Internet of Things; Data analytics; Deep learning; Federated learning; Transfer learning; Ensemble learning; IoT applications; Smart services; Usability; Flexibility.

First of all, I would like to thank God very much.
I dedicate my thesis work
to my dear parents Mehrez and Ahlem,
to my dear husband Omar,
to my daughters Nouha and Roua,
to my son Ibrahim,
to my sister Marwa,
to my brothers Jabeur and Ahmed,
to all my family,
to all my friends,
and to all people I love.

Acknowledgements

Above all else, I thank Allah for blessing me with good health and unwavering patience. I was able to navigate the numerous challenges and obstacles that I encountered during my Ph.D. journey and throughout my life because of His divine guidance and help.

I want to express my deepest sincerest gratitude to my doctoral research director, Pr. Henda HAJJAMI BEN GHZELA, Professor at the University of Manouba, for her exceptional guidance and unwavering support during my thesis.

I am deeply indebted to my thesis co-director, Dr. Wadii BOULILA, whose selfless guidance and mentorship have been instrumental in my academic journey. His valuable support, constructive feedback, and encouragement have inspired me to become a better researcher.

Furthermore, I would also like to extend my appreciation to my advisor, Dr. Maha DRISS, whose unwavering support and encouragement have been instrumental in my success. I am incredibly thankful for the priceless chance to pursue my Ph.D. under her tutelage and guidance.

I wish to thank Dr. Imed Riadh FARAH, Professor at the University of Manouba, for the honor he has bestowed upon me by presiding over my thesis defense jury.

I am also deeply thankful to Dr. Yassine JAMMOUSI and Dr. Faisal SAEED for taking the time to review this thesis. I thank them for the remarks and critiques they have provided me, which have helped improve this manuscript.

I also thank Dr. Hanen IDOUDI, a Professor at the University of Manouba, who kindly agreed to be part of this jury.

Finally, I wish to offer my heartfelt appreciation to my family for their ongoing encouragement and understanding throughout my research journey. Their unconditional love, encouragement, and understanding have been a constant source of motivation and strength for me all through the good and bad moments of this challenging path.

Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 27

ANN Artificial Neural Network. 30

ARIMA Autoregressive Integrated Moving Average. 41

CNN Convolutional Neural Network. 30

DBN Deep Belief Network. 30

DL Deep Learning. 4

DNN Deep Neural Networks. 30

DoS Denial-of-Service. 97

DRL Deep Reinforcement Learning. 30

DST Dempster-Shafer Theory. 102

DT Decision Tree. 43

FedAvg Federated Averaging. 68

FL Federated Learning. 3

FN False Negative. 23

FP False Positive. 23

GAN Generative Adversarial Network. 30

IoT Internet of Things. 1

KNN K-Nearest Neighbours. 43

LDA Linear discriminant analysis. 43

LEACH Low Energy Aware Cluster Hierarchy. 97

LISPS Lightweight IoT Smart Public Safety. 44

LR Linear Regression. 43

MCC Matthews Correlation Coefficient. 24

ML Machine Learning. 10

MLMA Machine Learning in Microservices Architecture. 43

MLP MultiLayer Perceptron. 44

PE Portable Execution. 94

QoS Quality of service. 54

RBM Restricted Boltzmann Machines. 30

RNN Recurrent Neural Network. 30

SAE Stacked Autoencoder. 30

SOA Service Oriented Architecture. 6

SVM Support Vector Machine. 43

TL Transfer Learning. 8

TN True Negative. 23

TP True Positive. 23

WKG Water Knowledge Graph. 109

WQI Water Quality Index. 109

WSNs Wireless Sensor Networks. 97

XDL Explainable Deep Learning. 117

Chapter 1

General Introduction

Contents

1.1 Context and Motivation

Nowadays, the Internet of Things (IoT) has emerged to be a disruptive Internet technology. It helps to propel the Internet to the next level by connecting enormous heterogeneous devices, sensors, and actuators [1]. These devices can communicate and engage with the actual world, gather environmental data, and support smart decision-making processes [2]. The incorporation of these intelligent equipment in different sectors of the environment has enabled the creation of intelligent and innovative services, providing valuable insights and actionable knowledge. The IoT has facilitated the establishment of applications for various sectors, including smart cities, home automation, industrial management, and many more [3]. An IoT application could be identified as a set of automated operations and data combined

with other objects, including hardware and software. All these entities interact and exchange information with each other and the environment to achieve shared objectives and goals.

Alongside the rise of IoT and cloud computing, the amount of data generated has increased exponentially, leading to the need for efficient predictive analytics for various IoT applications. Consequently, IoT big data analytics has emerged as a key study field, as traditional methods of data processing are unable to deal with huge quantities of data [4]. Predictive analytics is a crucial process that transforms raw data into actionable insights and has been widely applied in diverse domains, including healthcare, smart home, energy management, and social media analysis, among others [5]. Therefore, it is imperative to develop novel data processing techniques and predictive analytic models that can handle the scale and complexity of IoT data for better decision-making and improved performance of IoT applications.

The growing volume of data produced through IoT devices and the need for efficient, scalable, and reusable analytics solutions have brought about several challenges [6]. These challenges include the heterogeneity of data with various types and forms, the growing volume of data, monolithic service development approaches, and the efficiency of predictive analytics. To address these challenges, there is a need to design a novel approach based on the microservices architecture that leverages the capabilities of data analytics techniques. This approach will enable the development of intelligent microservices that provide efficient predictive analytics, improving the efficiency of IoT applications in different domains. By adopting a microservices-based approach, it will be possible to overcome the limitations of monolithic architectures and enable the development of scalable, reusable, and efficient IoT-based analytics solutions.

1.2 Problem Statement

IoT devices are generating huge quantities of data from everywhere, each and every time, and in different formats. The collected data is moved to the cloud to be monitored, processed, and investigated in order to assist smart IoT applications. Employing data analytics on these huge data sets plays an essential role in various IoT applications, such as energy management, business, security, etc. Indeed, big data analytics has emerged as a major IT trend that involves academia, research

institutions, and industries [7]. However, the IoT analytics community is confronted by new issues and challenges.

Managing data from multiple sources can be difficult because each device in the IoT ecosystem may produce data in different formats and with varying levels of accuracy. In addition, the data may be incomplete or contain errors that need to be corrected before it can be analyzed effectively. This can lead to delays and inaccuracies in data processing, which can negatively impact the performance of IoT applications. Dealing with different data types and formats is also challenging, as IoT devices generate data in a wide range of formats, such as text, images, and sensor data. Different data types require different processing methods, and handling them all is a complex and time-consuming procedure. Moreover, the heterogeneity of data types in IoT can also result in interoperability issues, which further exacerbate the problem. In addition, managing the fast-growing data volumes can be a significant challenge, as the volume of data produced by IoT equipment is increasing exponentially. Traditional techniques of data storing and processing may not be able to keep up with this growth, leading to scalability issues that can impact the effectiveness in general and the re-usability of IoT applications.

In furtherance of the difficulties associated with processing large amounts of data, transferring this data stored on the edge equipment to the centralized server can also be costly and risky [8]. This is because IoT devices are often located in dispersed locations, far from the central server. Transferring large volumes of data from these devices to the centralized server can be time-consuming and expensive.

Furthermore, transferring data from IoT devices to the central server may also expose data to various attacks and threats. These threats include eavesdropping, data tampering, and denial-of-service assaults. Attackers can intercept data during transmission, steal sensitive information, or modify the data for malicious purposes. This can compromise the privacy and security of data and potentially cause harm to individuals and organizations.

In summary, developing a scalable design of data analytics approach for IoT applications that takes into account all these considerations is crucial to successfully handling and investigating the huge volumes of data produced by IoT devices. The efficiency of IoT data analytics may be optimized and improved by using technologies such as edge computing and Federated Learning (FL).

1.3 Objectives

This work aims to propose a microservices-based architecture for the creation of IoT applications. The suggested design is made up of software components that have been designed to support centralized and distributed learning and satisfy the needs of IoT applications. We advocate an intelligent microservices-based approach to guarantee data privacy and deliver proper data analytics to the end users. We propose decomposing the IoT application into a linkable set of microservices running across different computing layers (cloud/edge). We focus on delivering an appropriate service computation distribution method that takes into account the layer's constraints and decreases data transmission cost. The suggested approach can aid in developing sustainable computing architectures by allowing an efficient distribution of data processing at the device's edge.

1.4 Contributions

The following are the primary contributions of this research study:

- **A survey of leveraging Deep Learning (DL) and IoT big data analytics to aid in the development of smart cities [P12]**
  In this survey, we present an overview of the available research on the usage of IoT and DL to construct smart cities. To begin, we define the IoT and identify the features of IoT big data. Then, we discuss the various computing platforms employed for IoT big data analytics, such as cloud, fog, and edge computing. Following that, we report widely used models for DL and current studies using IoT and DL to create smart services and applications for smart cities. Lastly, we discuss the present challenges and concerns encountered in the development of smart city services.

- **Comprehensive review of microservices for data analytics in IoT applications [P3]**
  The published review represents a pioneering effort to discuss the current state of solutions that employ microservices-based architecture to support data analytics in IoT applications. Our comprehensive analysis sheds light on the potential benefits and obstacles of using microservices in the context of IoT,

and we highlight the most promising directions for further studies in this field. Furthermore, we emphasize the importance of integrating cutting-edge microservices technologies into IoT systems design to enhance effectiveness, scalability, and adaptability.

- **The development and implementation of DL models to enable analytical processing of many kinds of data generated through IoT applications. [P1, P2, P3, P4, P10, P11]**
  In this part of the contributions, we proposed different DL models in various IoT domains, including healthcare, security, and water management. This part allowed us to prepare the test−bed and design solutions for specific problems.

- **The suggestion and the implementation of a microservices-based approach that supports data analytics for IoT applications [P5]**
  We proposed an intelligent microservices-based approach to ensure the appropriate privacy and provide adequate data analytics to the end users. The suggested design is made up of several software components that have been designed to support centralized and distributed learning and satisfy the needs of IoT applications. We suggested decomposing the IoT application into a linkable set of microservices that run across different computing layers (cloud/edge). We focused on delivering an appropriate service computation distribution method that takes into account the layer's constraints and decreases data transmission cost. The performance analysis proved that our proposal satisfies the objective of the development of sustainable computing architectures as well as the high efficiency of the results.

- **The demonstration of the viability and the validity of the suggested approach through a set of experimentation [P7, P8, P9]**
  In this part, a series of experiments are conducted, and the findings are analyzed and contrasted with existing cutting-edge methodologies. This allowed for a thorough evaluation of the performance and productivity of the proposed solutions, as well as the identification of areas for further improvement.

1.5 Thesis Outline

The rest of this thesis is structured in the following order:

Chapter 2 thoroughly examines the latest advancements in IoT, data analytics, and microservices paradigm. This chapter explains the learning techniques for developing valuable IoT application services. It discusses the essential benefits of microservices for IoT data analytics, challenges, and future directions.

Chapter 3 is dedicated to the related works. In this chapter, various works in the field are reviewed, their limitations are outlined, and the gap in the existing research is identified.

Chapter 4 presents the suggested approach in detail. This chapter elaborates on the different phases of the approach and how each phase contributes to the overall effectiveness of the proposed approach.

In **Chapter 5**, several analytics models have been proposed and designed, ranging from simple DL models to SOA designs. The proposed models have been evaluated and compared regarding their efficiency and effectiveness. Moreover, each model's strengths and weaknesses are identified, and recommendations for future improvement are given.

Finally, **Chapter 6** summarizes the different work phases and provides some concluding remarks, along with promising work that could be carried out in the future within this field of study to enhance the efficiency and effectiveness of IoT data analytics using the proposed microservices-based approach.

Chapter 2

Preliminary Concepts and Theoretical Background

Contents

2.1 Introduction

The convergence of IoT technologies and advanced data analytics has ushered in a new era of intelligent and connected systems. In this background chapter, we explore key components that form the foundation of efficient IoT data analytics including IoT, microservices paradigm, data analytics, Transfer Learning (TL), FL, and ensemble methods. We begin by delving into the IoT and its transformative role in collecting vast amounts of data from interconnected devices. Next, we explore the microservices paradigm, a powerful framework for building scalable and flexible IoT data analytics solutions. Data analytics techniques, including ML and DL, play a crucial role in extracting insights from vast amounts of IoT data. TL, FL, and ensemble methods contribute to the advancement of IoT data analytics by enhancing performance, addressing privacy concerns, and improving predictive accuracy and robustness.

By understanding IoT Microservices architecture, data analytics, TL, FL, and ensemble methods, we gain insights into the fundamental components that drive effective IoT data analytics.

2.2 Internet of Things

Before detailing the different data analytics in IoT applications, an overview of IoT is necessary. Therefore, we will provide the IoT definition, the IoT architecture, the properties of IoT big data, and the different applications of IoT architecture.

2.2.1 Definition

IoT is a concept that involves connecting numerous objects to create a smart environment [9]. This linkage is accomplished using standardized communication protocols that allow devices to exchange and share data through different frameworks [10]. Consequently, IoT significantly enhances the interaction and effectiveness of vital infrastructure in fields such as transportation, security, education, agriculture, and healthcare. The IoT design consists of four different stages, namely hardware, connectivity and communication middleware, big data storage and analytics, and IoT applications [11]. Each of these levels is briefly described in the following paragraphs, and the structure is shown in Figure 2.1.

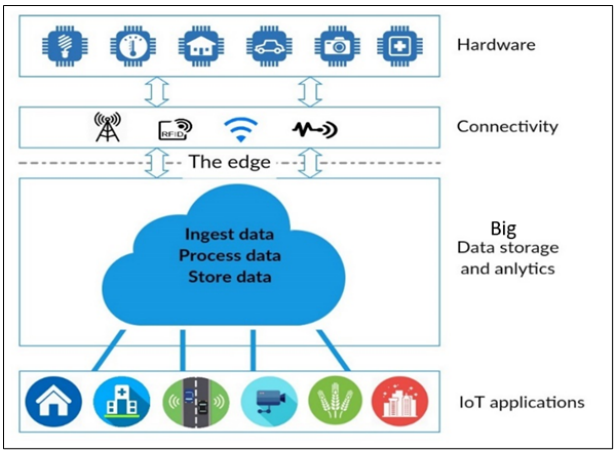


FIGURE 2.1: The design of IoT systems.

Hardware

At the hardware level of IoT design, smart devices, which consist of sensors and actuators, serve a crucial part in capturing and processing signals. Sensors are responsible for collecting data generated by the surrounding environment, whereas

actuators convert electrical signals into tangible movements or operations. By collecting real-time data, sensors enable the interconnectivity of physical devices and digital networks. Depending on the purpose of the IoT application, various types of sensors are utilized, such as wearable sensors for monitoring human activity and others for measuring factors like temperature, humidity, air quality, pressure, speed, movement, length, and even heart rate [12].

Connectivity and communication middleware

In the IoT ecosystem, the collected data from the hardware level is usually stored in the cloud for further investigation and analysis. However, to achieve this, it is essential to have reliable and efficient data transmission from detectors to storage and analytics tools. This is where connectivity and communication middleware come into play. These middleware act as intermediaries between the hardware level and the storage and analytics tools. Their primary function is to make sure that data is transferred smoothly and securely from sensors to the cloud. Different types of middleware technologies such as WIFI, RFID, and Ethernet are used depending on the application requirements [13].

Big data storage and analytics

To utilize the generated data from IoT devices, it is critical to save and examine it in order to extract meaningful insights that can inform decision-making [14]. Data analytics is the procedure of translating raw data into usable insights, and it includes the following kinds of analysis: descriptive, predictive, and prescriptive [14]. Descriptive analytics summarizes and describes past events and data, providing a historical perspective. Predictive analytics use models of statistics and Machine Learning (ML) techniques to identify patterns and trends in historical data to make informed predictions about future events. Prescriptive analytics takes the analysis a step further and provides recommendations on actions to achieve a desired outcome.

IoT applications

To leverage the potential of IoT, the top-level IoT architecture is dedicated to IoT applications. IoT applications have been developed across various sectors, such as factory automation, predictive maintenance, waste management, and public safety, to enable real-time decision-making and actions in a smart environment. Through the use of IoT applications, the environment can be transformed into a dynamic and interactive ecosystem, enabling efficient and optimized workflows.

2.2.2 IoT Architecture

IoT architecture is the foundation that allows interconnected devices to communicate with each other and create an IoT system network. This architecture is typically structured in layers, enabling administrators to analyze, monitor, and maintain the system's integrity [15].

The IoT architectural stack is a commonly used framework that includes five layers, as depicted in Figure 2.2: the business layer, application layer, processing layer, and perception layer. Each layer is significant in the IoT system's overall functioning, from collecting and processing data to providing valuable insights and taking appropriate actions. The following points provide a basic overview of the responsibilities of these different layers:

- **Business layer:** This business layer serves as the foundation of the IoT architecture, as it oversees and manages the entire system's functionalities, applications, and business models. This layer acts as a bridge between the technical aspects of IoT and the business objectives of the organization. It provides a holistic view of the system and helps stakeholders to make informed decisions based on data-driven insights. Moreover, this layer ensures that the IoT system aligns with the business's overall strategy, goals, and vision, providing a sustainable growth and development framework.

- **Application layer:** This application layer of IoT architecture plays an important role in managing the processed data. It serves as a mediator between the processing layer and end-users or clients, allowing them to interact with the system and make decisions based on the generated insights. This layer transforms data into useful information that is tailored to specific user requirements. Furthermore, it enables the development of customized applications and services that cater to various domains and industries, such as healthcare, smart homes, and transportation.

- **Processing layer:** This processing layer performs data processing and analysis to extract meaningful insights and knowledge from the collected data. This layer employs various data analytics techniques, such as descriptive, predictive, and prescriptive analytics, to transform raw data into actionable insights. Additionally, this layer employs advanced ML techniques to uncover hidden trends and relationships in the data. The results obtained from this layer can

be used to optimize various aspects of IoT systems, including energy efficiency, performance, and security.

- **Transport layer:** This transport layer establishes communication links among IoT devices and transfers collected data to upper layers. This layer uses various communication protocols and technologies such as Wi-Fi, Zigbee, Bluetooth, and cellular networks to ensure seamless and secure data transmission. Additionally, the transport layer manages the quality of service to prioritize critical data and ensure timely delivery.
- **Perception layer:** The perception layer has the task of gathering information from the surroundings with the aid of a range of sensors and actuators. The sensors acquire data and transmit it to the transport layer for additional analysis and handling, while the actuators receive instructions from the processing layer and execute physical actions in the environment. This layer is vital for the entire efficiency and efficacy of the IoT ecosystem since it is the first step where data is gathered, making it critical to guarantee the sensors' and actuators' precision and dependability.

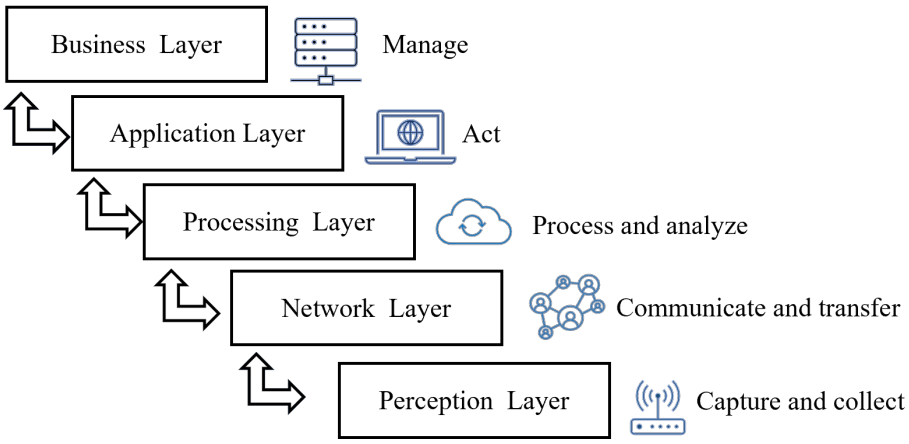


FIGURE 2.2: IoT architecture

2.2.3 IoT Big Data Characteristics

IoT has gained popularity in various fields, resulting in the creation of numerous applications that create huge data volumes in various forms, such as images, videos,

and audio. To address the challenges of IoT big data, Mohammadi et al. [13] have identified six key features, known as the six V's. The initial characteristic is Volume, which pertains to the enormous quantity of data produced, necessitating new methods for processing and extracting valuable insights. The second characteristic is Velocity, which denotes the rapid generation of data by numerous IoT equipment that automatically gather, record, and transmits data throughout various platforms leveraging internet standards. The third feature is Variety, which includes unstructured, quasi-structured, semi-structured, and structured data. The fourth characteristic is Veracity, which relates to the correctness and reliability of the obtained data, affecting the quality of analytics results. Variability, the fifth characteristic, depicts the varied patterns of data collection and movement based on both time and information sources. Finally, the sixth feature is Value, which evaluates the significance of the gathered IoT data upon examination, providing useful insights for improving efficiency in production and gaining a competitive edge.

According to Chebbi et al., [16], the introduction of innovative software and hardware architectures has accelerated the expansion of big data created by IoT. These systems have been created to manage the particular features of IoT big data in a timely manner. Multiple research investigations have examined the currently accessible big data mediums, classifying them as horizontal or vertical scaling platforms [17]. Horizontal expansion frameworks are designed to offer users incremental improvements in program efficiency and performance. Apache Hadoop, Mahout, Spark, and H2O represent a few of the foremost prominent horizontal scaling systems. Vertical scaling platforms, in contrast, are made up of numerous parts that may add additional strength or capacities, such as boosting processing capacity by attaching additional processors and quicker memory. Graphics processing units, high-performance computing clusters, and field programmable gate arrays are examples of prominent vertical scaling platforms [3].

2.2.4 Applications of IoT

The Internet's swift growth, IoT, and other technological advancements are the major factors contributing to the continuous growth of smart services across various fields [18]. The creation of smart cities is a prominent field of IoT application, in which multiple devices and applications are built to function autonomously or adapt to operational or environmental changes without human interaction. These smart devices can improve their decision-making capabilities and actions by analyzing

previously gathered data. To facilitate this process, The acquired data is kept on cloud or fog servers, and there it is analyzed using various data analytics techniques.

Various industries have recently been able to adopt smart technologies and practices. Some of the sectors that have benefited from these technologies include:

- **Smart homes:**
  Smart homes are equipped with internet-connected devices such as refrigerators, lights, and televisions that exchange data with each other and with users [19]. The major goal of smart home deployment is to improve the control of devices and optimum energy usage.

- **Smart healthcare:**
  Smart healthcare utilizes technology to provide continuous monitoring of patient's welfare and wellness, allowing for immediate information and enhancing care [20]. Wearable and non-wearable devices are used to control a user's daily activities and record various health metrics, such as blood pressure, heart rate, body temperature, oxygen saturation, respiratory rate, and glucose level. These gadgets are linked to the internet and provide information to doctors, allowing for remote monitoring and personalized care.

- **Smart transportation:**
  Smart transportation aims to improve transportation systems by utilizing advanced equipment and optimized services [21]. This includes providing convenient smart parking solutions, reducing population density through efficient public transportation, and managing mobility inside the city. The ultimate goal is to enhance traffic flow and provide a seamless and sustainable transportation experience for all users.

- **Smart surveillance:**
  Smart surveillance is the use of observation technologies, like advanced cameras, strategically positioned around a city to improve security and avoid incidents. These smart security systems gather and analyze data in real-time to spot potential catastrophes before they happen, which can help avoid harmful situations from occurring [22].

- **Smart agriculture:**
  Smart agriculture entails using IoT devices in farming activities in order to speed up and enhance standard processes [23]. This approach helps to ensure

healthy crop production and can enhance various processes such as collecting, packing, and shipment via the use of services like real-time monitoring, predictive analytics, and automated irrigation systems.

- **Smart environment:**
  The use of advanced smart sensors distributed across various regions of the city to enhance infrastructure and monitor the surroundings is referred to as the smart environment [24]. By collecting and analyzing data on factors like air quality and water management, smart environment technologies aim to create a healthier and more sustainable environment [25].

The emergence and evolution of IoT data have given rise to various data analytics techniques. Big data analytics, specifically, involves analyzing vast and diverse data sets to find unnoticed relationships, behaviors, and customer needs, among other things [26].

2.3 Microservices Paradigm: Overview

Nowadays, IoT and big data analytics are popular areas of research. However, there is still a significant gap between the IoT's ability to gather environmental information as well as the analytical capabilities required to analyze such data. This creates various challenges, making it challenging to exploit data analytics in IoT systems. Among the most challenging challenges involve handling heterogeneous IoT data, quickly analyzing acquired data, and creating and maintaining flexible and adaptable functions that may be utilized in different applications and settings. To handle these challenges, an adaptable and scalable architecture, rather than a monolithic one, is the best approach. A number of recent research projects have chosen to employ the microservices-based architecture for IoT application development in this context.

The microservices architecture has transformed software application development, becoming popular as a means of creating adaptable and distributed applications [27]. Each service is constructed separately in a microservices-based architecture, with cheap communication techniques and standards adapted to meet specific company requirements [28]. This method is especially advantageous for restricted resource setting like IoT, where a modular structure is essential for enabling multiple

analytical functions. This architecture employs a collection of distinct services that are in charge of certain analytic features, allowing them to be established, expanded, and operated separately from other services. This leads to enhanced efficiency and resistance to faults, which is especially important in an IoT setting. Unlike monolithic solutions, this architecture considers each service's analytic process, resulting in more efficient utilization of resources [29].

2.3.1 Microservices' Definition

Microservices architecture is a kind of web-based service that enables the development of distributed software platforms by leveraging a number of fine-grained, loosely linked, and reusable elements of software which collaborate to fulfill the system's goals [27]. Every single service in a microservices-based architecture operates in its own unique operation and shares data with others. Unlike the monolithic development approach, which uses just one piece of coding to create and construct standard applications, a microservices architecture represents a program as an ensemble of distinct but interrelated components which are capable of being created, evaluated, and deployed separately, despite being distinct elements. APIs, containers, a service mesh, service-oriented architecture concepts, and the cloud are examples of these things, in addition to individual services. By decomposing an application into these entities, microservices architecture provides greater flexibility, scalability, and resilience than traditional approaches.

Microservices' scalable and flexibly linked architecture serves as an ideal fit for IoT software, allowing it to be easily available across several settings. By using this approach, IoT applications may be composed of smaller parts, leading to less dependency and greater flexibility for change. This approach is particularly useful in IoT systems because it helps to address the diversification that occurs through the usage of numerous hardware and standards. Using approaches like web-based semantic frameworks and tools, microservices may be leveraged to assure consistent information delivery. This approach provides a way to standardize data and enable various devices as well as protocols to interact with one another, thereby reducing the complexity of IoT systems. Additionally, microservices architecture provides a scalable solution to IoT software development, allowing individual services to be scaled up or down as required to meet the demands of changing environments.

2.3.2 Microservices Advantages for IoT Data Analytics

The microservices architecture is gaining popularity in IoT systems due to several advantages. Here are some of the main benefits of this development style [30]:

- **Availability:** Because of the independence of the services, the microservices design provides the availability of software applications with less downtime, which substantially helps to enhance and boost overall performance. Actually, whenever a failure occurs, it doesn't impact the entire system; rather, it only impacts the affected entities, which could be substituted by utilizing devoted payment processes.
- **Dynamism:** Microservices design is a DevOps-based technique that employs parts that are modular as opposed to a monolithic structure to assure an enhanced flexible and efficient approach for creating, carrying out, and sustaining software systems.
- **Modularity:** Microservices give programs with robust encapsulation, well-defined interactions, and unambiguous relationships. A microservice can be constructed in any way which offers an established connection for additional services. Its implementation details are internal to the service, and they can change without affecting or synchronizing the rest of the system. Dependencies between microservices are explicit because they are defined by coordination and choreography guidelines.
- **Scalability:** The microservices architecture allows software development teams to use different technologies and programming languages for each service, without having to worry about compatibility issues. This means that teams can choose the best technology stack for each module and upgrade individual services without disrupting the entire system. Additionally, components can be introduced without requiring downtime or re-deployment. Furthermore, services can be deployed across multiple servers, reducing the impact of resource-intensive components on the overall system performance [31].
- **Functionalities' Optimization:** The microservices architecture enables easier customization of each service to enhance and optimize business functionality since the focus is on a single service rather than the entire application. This approach allows development teams to concentrate on business logic capabilities rather than technologies, as each service can be developed and upgraded

independently. Additionally, existing services can be modified and reused for different purposes, eliminating the need to reimplement new services from scratch.

- **Resilience:** With a microservices architecture, it becomes easier to identify and fix performance issues as the independent service deployment provides better fault isolation. In case of a failure, only the specific service is affected, which makes it easier for developers to roll back an update or make changes to the service without affecting the overall application. This reduces the possibility of downtime, and complex applications remain unaffected by a single failure, which in turn simplifies the process of identifying and fixing the main cause of performance issues [32].

The data analytics results can be improved by utilizing various techniques, such as TL, FL, and ensemble learning, among others. These techniques will be discussed in the upcoming sections.

2.4 Data Analytics

Recently, IoT big data analytics domain has gained significant attention. This is mainly due to the limitations demonstrated by traditional data processing methods, especially when dealing with vast amounts of diverse data.

The process of data analytics involves five key stages [33], which are illustrated in Figure 2.3. The initial stage is requirements comprehension, which entails recognizing the issue, the companies' demands, and the reason for performing data analytics. This stage is crucial, even though it may be time-consuming and challenging, as it sets the groundwork for the subsequent stages. Once the needs are established, the next stage is data collection. This involves gathering data, which must then be cleaned, filtered, and sorted to ensure the accuracy and effectiveness of the analysis. The next stage involves using appropriate analytics techniques to manipulate and analyze data. Finally, the outcomes and outputs provided by the analytics process are interpreted in the last stage.

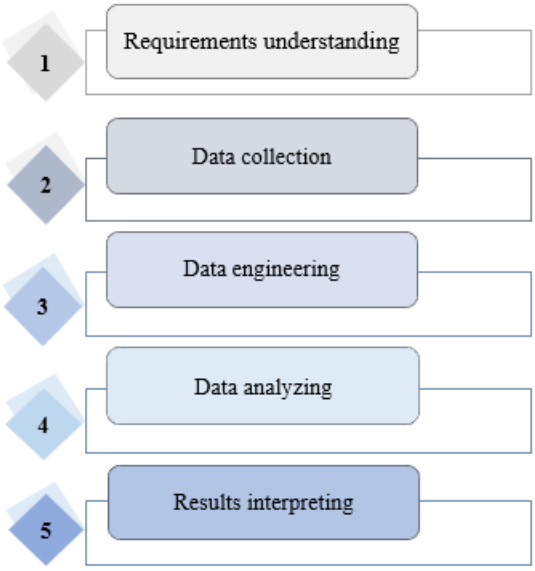


FIGURE 2.3: The different steps of the data analytics process.

In the parts that follow, we will present the categories of data analytics, outline the primary analytical tasks performed by researchers, examine the primary computational infrastructures employed in IoT big data analytics, and offer an outline of the major analytics approaches.

2.4.1 Paradigms

As indicated by Saheb et al. [34], data analytics are typically divided in four major groups: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning [34]. This classification is depicted in Figure 2.4. Each of these categories is briefly described below:

- **Supervised learning:** This refers to the process of acquiring knowledge by using labeled data to predict outputs for new, unlabeled data. Classification and regression are among the most common applications of supervised learning.

- **Unsupervised learning:** This involves examining the trends and correlations that emerge organically among data. Unsupervised learning is employed to discover unseen and intriguing patterns within data.
- **Semi-supervised learning:** This occurs when the experimental data has a little bit of data with labels mixed together with a big amount of data that is unlabeled.
- **Reinforcement learning:** This is a hybrid of supervised and unsupervised learning. It enables an "agent" to gain knowledge from its surroundings, allowing it to act and behave intelligently [35]. To inspire and create a flawless behavior policy, the agent communicates in real-time with its surroundings through "actions" and obtains "rewards" from the surroundings. The most extensively used reinforcement learning approach is Q-learning.

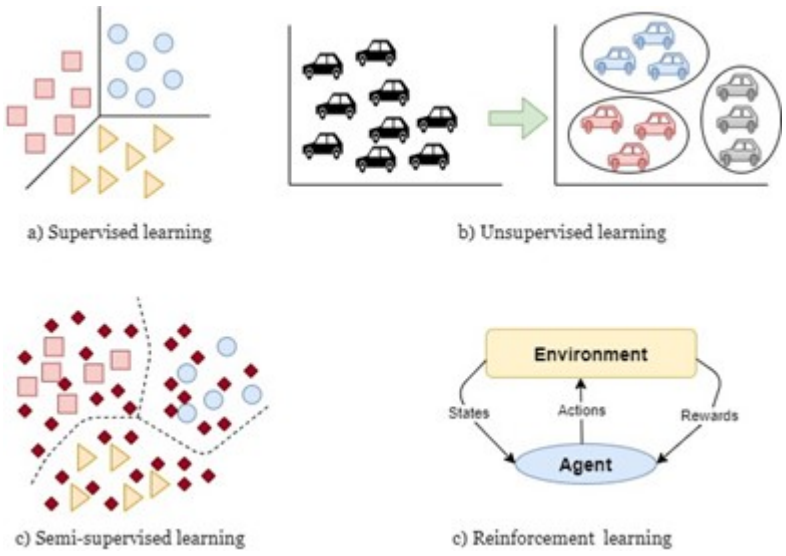


FIGURE 2.4: Different types of learning techniques.

2.4.2 Types of Data Analytics

As per the findings of Marjani et al. [26], there are four main categories of data analytics: descriptive, diagnostic, predictive, and prescriptive analytics, as depicted in Figure 2.5. Below are concise explanations of each of the four types of data analytics:

- **Descriptive analytics**: This refers to a broad type of analytics that seeks to improve taking decisions through analyzing and gaining knowledge from previous accomplishments in order to improve present performance.
- **Diagnostic analytics:**This sort of analytics aids in identifying and comprehending the main reasons for occurrences, as well as determining which elements impacted the final outcome.
- **Predictive analytics:** This type of analytics aims to gain knowledge and useful information regarding the future using unprocessed data, employing the both historic and present data to forecast behaviors and occurrences.
- **Prescriptive analytics:** This type of data analysis is the more powerful approach that assesses the best response from several options and suggests decision options to maximize potential benefits or minimize risks in the future.

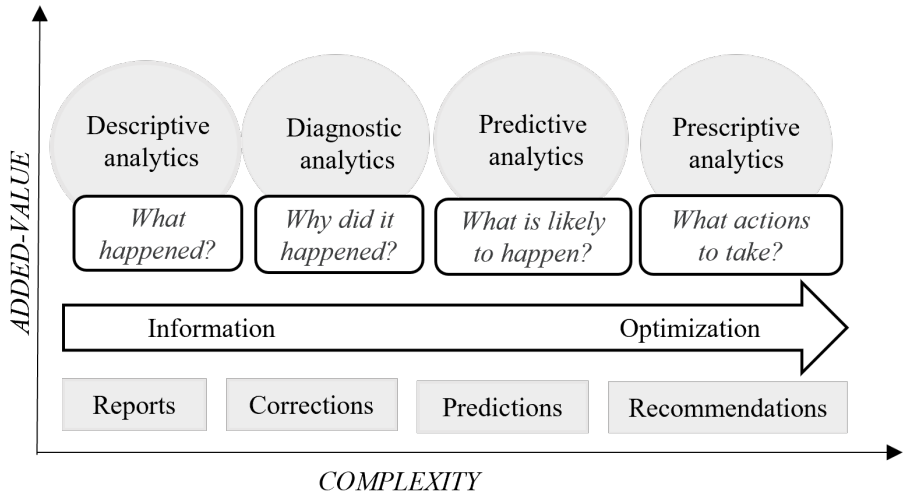


FIGURE 2.5: Data analytics categorization.

2.4.3 Analytical Tasks

Various analytical techniques have been suggested to accomplish different analytical tasks [26], which include:

- **Classification:** This technique involves using classification algorithms to distinguish the class of a new observation among a set of categories based on the learning gained from the input data. It is widely used in ML and can be applied in various fields such as image recognition, sentiment analysis, and spam filtering.
- **Clustering:** This technique involves using clustering methods to form and recognize groupings of comparable goods based on specific characteristic qualities. It is useful for discovering patterns in data and is commonly used in market segmentation, social network analysis, and image segmentation.
- **Recognition:** This technique involves teaching machines how to identify and categorize events, objects, places, or people from images or video data, mainly. It is commonly used in computer vision and can be utilized in a variety of areas, including recognition of faces and handwriting recognition.
- **Detection:** This technique involves analyzing photos and videos to detect items or situations. It is widely employed in surveillance systems, medical imaging, and quality control in manufacturing.
- **Prediction:** This technique is a more advanced sort of analytics that is utilized to uncover trends concerning occurrences in the future and indicate the necessary actions to be performed. It is commonly used in predictive maintenance, demand forecasting, and fraud detection.
- **Analysis:** This technique involves cleaning and processing data to extract relevant information that supports decision-making. It is a crucial step in data analytics and can be used in various fields such as business intelligence, customer profiling, and social media analysis.
- **Recommendation:** This technique uses data mining techniques to offer clients products, solutions, and recommendations on the most appropriate path of action. It is commonly used in recommendation systems, personalized marketing, and e-commerce.

2.4.4 Evaluation Metrics

Several metrics are employed to evaluate the performance of data analytics models, including accuracy, precision, recall, and F1-score. These statistical measures are represented mathematically in Equations 4.7 – 2.5, where:

- True Positive (TP): The prediction is positive, and it is accurate.
- True Negative (TN): The prediction is negative, and it is accurate.
- False Positive (FP): The prediction is positive, but it is inaccurate.
- False Negative (FN): The prediction is negative, but it is inaccurate.

Accuracy: is a measure applied for assessing a strategy's overall performance across every category. It can be described by the proportion of accurate forecasts compared to the entire forecasts generated through the algorithm. In simpler terms, accuracy assesses a model's ability to properly anticipate both positive and negative outputs.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Precision: is a measure applied for assessing the model accuracy in classifying a sample as positive or negative. It can be described by the proportion of the number of TP forecasts to the sum of TP and FP forecasts. Therefore, precision examines how well the model identifies positive samples without falsely classifying negative samples as positive.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

Recall: is a measure applied for assessing the capability of the model to recognize positive instances appropriately. It can be described by the proportion of the number of TP forecasts to the sum of TP and FN forecasts. Therefore, recall assesses how well the model detects all positive instances without missing any.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

F1-score: is a statistic that incorporates precision and recall measurements to provide an individual value-added rating for performance verification. The F1-score gives a more realistic assessment of the model's performance through the combination of accuracy and recall.

$$F1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.4}$$

Specificity: is a measure applied for evaluating the model's capability to appropriately detect negative samples. It measures how well the model identifies negative samples without falsely classifying positive samples as negative.

$$Specificity = \frac{TN}{TP + FN} \tag{2.5}$$

Matthews Correlation Coefficient (MCC): is considered to be a balanced metric, as it takes into account both the number of right forecasts versus the number of incorrect forecasts. It is particularly useful in cases where the data is imbalanced, as it provides a more accurate assessment of the model's effectiveness compared to other measures that do not take into account the imbalance in the data.

$$MCC = \frac{(TP * TN) - (FP * FN)}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{2.6}$$

Loss: is a function used to estimate the variation between a model's forecast and the real result for a given input. It is also known as the cost function or goal function. This function is used to assist the model during training by modifying its parameters so that the disparity between expected and actual outputs is as little as possible.

2.4.5 Computational Infrastructures for IoT Big Data Analytics

Data analytics for big IoT data sets has resulted in substantial improvements in accuracy and performance over the last several years [3]. These advancements, nevertheless, come with significant computing and memory needs, which may be met by utilizing modern computing platforms. In addition to cloud computing, new ideas such as fog and edge computing have evolved to enable data analytics to be performed as near to the data source as feasible. Both fog and edge computing expands the cloud network, although they operate in different locations. While fog computing is performed on local network servers, edge computing is performed directly on intelligent IoT devices. The next sections will go through the ideas of cloud, edge, and fog computing in further detail.

2.4.5.1 Cloud Computing

According to Stergiou et al., [36], cloud computing is an approach that allows users to access data from any location and at any moment. It is composed of multiple server rooms and data centers which are available via the Internet to users worldwide. Cloud computing has several distinct features, including:

- **Internet-based storage:** Cloud computing is an infrastructure for connecting servers to various dispersed devices using TCP/IP networks. This allows for storage deployment and data movement to be carried out with ease.
- **Internet-based services:** The cloud provides a broad range of services, including connectivity and computational intelligence to assist people in various locations.
- **Internet-based applications:** Cloud applications are programs that are hosted on the cloud, and they carry out their functions via Internet connections rather than functioning locally on machines after deployment.

The application of data analytics techniques to IoT data is enabled through the use of cloud computing [37]. With this method, IoT data is transferred to the cloud infrastructure, where it is stored and analyzed using various analytics methods in real time. Despite its benefits in terms of environmental sustainability compatibility, and adaptability, cloud computing for IoT data analytics comes with certain limitations [38]. Among these constraints are the following:

- **High computation cost:** The transmission of large volumes of data during the analytics phases increases network bandwidth usage, resulting in high costs for cloud resource utilization.
- **Latency:** The response time for accessing cloud services is dependent on network strength. The cloud may sometimes fail to respond quickly, particularly for time-critical applications like financial trading applications.
- **Reliability:** Cloud applications and other services are delivered via the internet and wireless connectivity. If there are any problems with the network connection, it can have a direct impact on the dependability of the services provided.
- **Privacy:** Storing data collected from IoT devices in the cloud can raise privacy concerns due to the sensitive and private nature of the information. There is a risk that the data may be lost during transmission or accessed by unauthorized individuals.

2.4.5.2 Fog Computing

To avoid having all data analytics processes conducted solely on the cloud, fog computing has been proposed as an alternative [39]. Fog computing is an extension of cloud computing that puts computational power closer to the network's edge, specifically designed to sustain various IoT applications [40]. It involves moving computational nodes and data analytics nearer to end-user devices, making use of a large number of diverse ubiquitous and distributed units that interact and possibly collaborate with one other and the network to conduct data storage and processing operations without the participation of other entities. As stated by Vaquero et al., fog computing is the placement of processing nodes and data analytics close to ending devices [41].

Storage, deployment, and networking are all services provided by cloud and fog computing concepts. The usage of fog computing, on the other hand, is limited to certain geographical locations, whereas the cloud is not. Furthermore, fog computing is designed primarily for active IoT applications that require immediate reactions. Fog computing, in essence, acts as a bridge between cloud services and edge devices such as IoT devices, allowing computation, communication, and storage operations to take place close to the edge. This method addresses challenges including rapid response times, restricted network capacity, and security and privacy concerns. The connection between edge devices and the cloud is depicted in Figure 2.6, showing how fog nodes play a role in linking them together.

2.4.5.3 Edge Computing

Edge computing has been introduced as a response to the limitations of cloud computing, as explained in the study by Yu et al. [42]. Edge computing involves processing data as close as possible to where it was generated, thereby extending the capabilities of cloud computing. This computing model offers increased computational power, enabling big volumes of data to be analyzed on the network edge before being transmitted to central cloud servers. The rapid expansion in data creation has made data transmission to the cloud difficult. By analyzing data on the edge instead of transmitting it into the cloud for investigation, bandwidth efficiency is improved, response time is reduced, network pressure is decreased, and energy consumption is minimized.

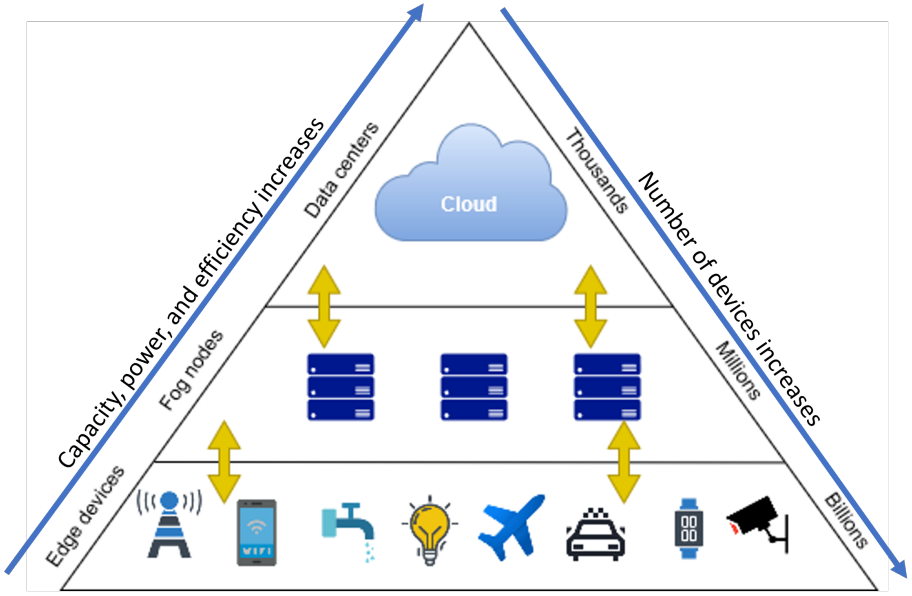


FIGURE 2.6: Fog nodes act as an intermediary between edge devices and the cloud.

Edge computing is particularly well-suited for IoT applications, according to Shi et al., [43]. An edge gateway may be employed to link every intelligent device and locally process data using on-demand services. This approach boosts reliability, safety, and privacy.

2.4.6 Data Analytics Techniques

The growth of ML and DL has brought Artificial Intelligence (AI) into various fields of our lives, making it an essential aspect. The capacity of computers to execute activities that normally require human intellect, such as picture identification, speech-to-text transformation, and translating languages, is referred to as AI. AI's capability has grown tremendously throughout the years.

ML is a subclass of AI that allows computer systems to acquire knowledge and enhance their behavior via training them on a specified dataset [44], [45]. Once the model is trained, it can make decisions without explicit programming.

DL, a more sophisticated ML approach, has sparked a lot of academic interest over the past decade [46]. According to Li et al., [47], DL is a potential strategy for retrieving reliable insights from raw data supplied by IoT devices in challenging situations.

Traditional ML systems rely on hand-engineered feature extractors to handle raw data, which often requires domain expertise to develop an efficient model. Additionally, feature selection must be adjusted and restarted for each new situation. In contrast, DL techniques are based on representation learning, which means they can directly detect and acquire meaningful characteristics or features from raw data, frequently outperforming standard ML methods, particularly when handling a huge quantity of data [48]. Figure 2.7 depicts the relationship among AI, ML, and DL.

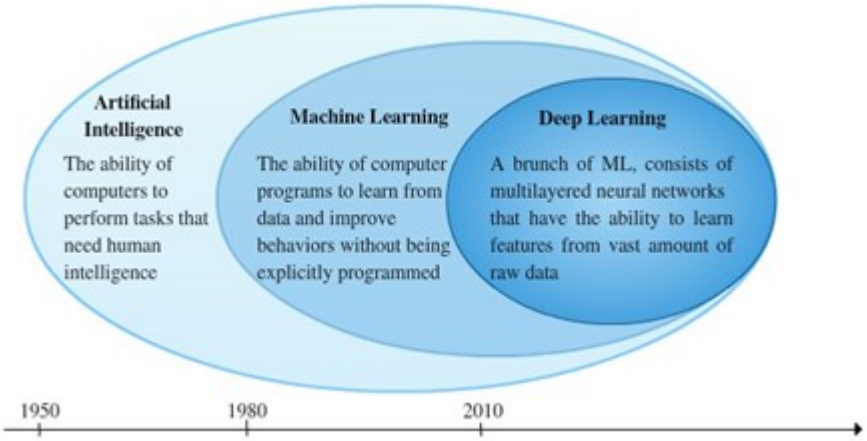


FIGURE 2.7: A diagram that uses overlapping circles to show the distinctions among AI, ML, and DL.

2.4.7 Deep Learning

DL is a category within the computing paradigm inspired by the human brain, composed of several neural networks [49]. A neural network is designed similarly to a neuron, with three layers: input, hidden, and output. These neurons process incoming data and execute computations to generate a weighted sum before producing an output through an activation function. Each neuron has weights as well as a bias.

The weights that are used in the initial layer are set up according to the given input size, and the bias is adjusted throughout training.

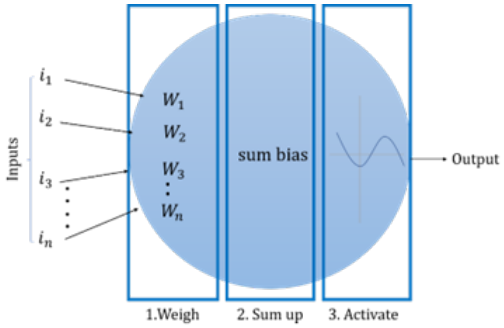


FIGURE 2.8: The neuron structure.

Weight initializers are utilized to set the initial weights of neural network layers and prevent the activation outputs from suffering from gradient vanishing and exploding issues during the training phase [50]. The problem of vanishing gradients is caused by the output loss of feedback signals to previous layers during back-propagation, leading to low or missing signals, rendering the network untrainable [51]. Hence, setting the weight values carefully is crucial to obtaining better outcomes and superior performance.

According to Sun et al., [52], there are three types of weight initialization methods. The first type involves constant approaches that employ identical weights for all connections in the neural network. Examples of these methods include the zero and one initializers. However, this approach can cause the learning algorithm to be unable to adjust or update the network weights, leading to a "locked" model where all layers perform the same computations [53]. The second category includes distribution methods, which randomly assign values to input matrix numbers using the Gaussian or uniform distribution. However, selecting the appropriate distribution parameters, such as the mean and standard deviation, is crucial, as incorrect values can negatively impact the model's performance and cause the issue of vanishing gradients. Random initialization that utilizes heuristics and prior information is the third category. Heuristics, in addition to nonlinear activation functions, are utilized in this technique to give weights to layers. Although these heuristics can mitigate the problem of vanishing or exploding gradients, they are not foolproof and may not completely eliminate the issue.

In the field of DL, there are several commonly used and relevant models. Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Belief Network (DBN), Stacked Autoencoder (SAE), Deep Reinforcement Learning (DRL), and Generative Adversarial Network (GAN) are examples of these models. DL algorithms are part of the larger category of Artificial Neural Network (ANN), which are distinguished by the presence of several hidden layers [54]. Because of the inclusion of these layered structures in DL approaches, the name Deep Neural Networks (DNN) is widely employed. CNNs are especially successful in image or video processing because they use filter operations to extract significant characteristics [55]. RNNs, on the other hand, are ideal for time-series predictions since they rely on loop connections between layers [56]. DBN is a DL model with stacked Restricted Boltzmann Machines (RBM) in its architecture [57], whereas SAEs are made up of several autoencoders [58]. DRL combines DNNs and reinforcement learning, allowing agents to acquire knowledge from their surroundings and set long-term reward plans [59]. GANs, on the other hand, are composite DL structures that combine a generative model and a discriminative model to achieve the intended outcomes [60].

In particular, CNNs are capable of autonomously learning hierarchical representations from the input. This hierarchical feature learning capability enables CNNs to automatically recognize detailed patterns and representations from raw data, making them particularly well-suited for picture classification, object recognition, and image segmentation applications. Furthermore, CNNs excel in detecting local patterns and structures in data. Additionally, CNNs may be efficiently pre-trained on large datasets such as ImageNet and then fine-tuned for specific tasks utilizing TL. This method overcomes the limits of training deep networks from scratch, especially when dealing with little labeled data. By leveraging the knowledge and representations learned from large-scale datasets, CNNs can significantly enhance their performance on targeted IoT tasks and facilitate efficient utilization of available data resources.

In the subsequent section, we will provide a comprehensive and detailed overview of the CNN architecture, elucidating its various components and their respective functions. By delving into the intricacies of CNNs, we aim to offer a comprehensive understanding of their structural elements and the roles they play in extracting meaningful features from input data.

2.4.7.1 Convolutional Neural Network

CNN is a DNN with numerous layers, each of which performs different filter orations [61]. It is commonly used for processing images and videos due to its effectiveness. CNN uses supervised learning, which means that it is capable of recognizing and categorizing patterns in data based on labeled examples provided during the training phase. This feature classifies CNN as a discriminative DL architecture according to reference [61].

The structure of this network is comprised of several layers, starting with an input layer and followed by a sequence of hidden layers before concluding with an output layer. The network is constructed with a sequence of convolutional and pooling layers, that are then succeeded by a fully connected layer. Within each convolutional layer, multiple filters are incorporated, generating distinct analyses of the input data to yield feature maps.

The CNN network comprises three fundamental layers: the convolutional layer, the pooling layer, and the fully connected layer. The following is a basic description of these layers:

- **T**he CNN model begins with a convolutional layer that performs a filter to the input picture, which has been transformed into a matrix. The resulting output is called a feature map, which contains the acquired features. The filter used in this layer is essentially a small matrix with predetermined dimensions and randomly selected values. It traverses the input matrix from left to right and then proceeds downwards, moving in steps as determined via the stride parameter as long as it completes the entire matrix. The convolutional layer process can be represented by Equation 2.7.

$$y_j = f(\sum_i K_{ij} * x_i + b_j) \tag{2.7}$$

  where x presents an input picture, y_j denotes the output of the j_th convolutional layer, f serves as the activation operation, K_ij is the convolutional kernel times the i_th input $x_(i)$, and b_j is the bias. Figure 2.9 depicts convolutional procedures.

- **Pooling layer:** is utilized to simplify the CNN model by extracting the most important features from the generated feature maps through either an average

FIGURE 2.9: Explanation of the operations carried out within the convolutional layer

or max-pooling operation. To perform this operation, a kernel or window of a predetermined size is applied to the feature map. The kernel gathers either the average or maximum values of the matrix elements based on the operation being performed and slides across the entire feature map with a set stride. Figure 2.10 illustrates the max pooling operation in the pooling layer, where four slide positions are shown with different colors. The resulting pooled values illustrate how the computations of the model become less complex.

FIGURE 2.10: Explanation of the operations carried out within the pooling layer.

- **Fully connected layer:** This layer is in charge of the CNN network's last phase by reconnecting the processed parts to reconstruct the entire image. This layer converts the two-dimensional array to just one list and uses activation algorithms, like Sigmoid and Tanh, to generate values for probabilities that reflect the likelihood of the processed picture belonging to a particular category. The output layer sends the picture to the output class with the biggest probability ratio.

Typically, the convolutional and pooling layers are often merged at the beginning of the network design, whereas the fully connected layers are piled together in the conclusion, unlike the former two.

How Does CNN Work?

CNNs have proven to be effective in various domains, particularly in image classification and recognition. The capacity of CNNs to discover and acquire hidden features using massive datasets is one of its most significant advantages [62]. CNNs follow a series of steps during the training phase, which are outlined below.

During the training stage, the input layer allocates weights to the data being entered, which are subsequently transmitted to the following layer. Depending on the initializer type used, the weights may be set to either constant or randomized values [63]. The subsequent layers receive weights, apply different filter functions, and generate an output that becomes the input for the subsequent layer. Finally, the ultimate output is established in the end layer.

During the stage of training, a loss function is utilized to evaluate how well a prediction performs. The function assesses the rate of error by analyzing and contrasting both the predicted and the actual outcomes. There are different types of loss functions created for various intentions. In particular, the binary cross-entropy function is implemented for binary classification issues that deal with only two categories, the categorical cross-entropy function is utilized for multi-class classification issues, and the mean-squared error function is employed for regression issues.

To verify the weights of a neuron, various optimization algorithms can be employed, including Stochastic Gradient Descent and Adam. These algorithms investigate the gradient of the loss function and endeavor to modify and adjust the network's weights or learning rates to decrease the losses. This sequence of operations is reiterated during the training period until the weights are adjusted for each neuron in every layer, and the error rate value declines. The model can be used after the training is completed.

A CNN model's performance can be improved by optimizing the right parameters. These parameters are the internal model configuration variables computed via data, such as the weights linking neural network layers.

2.5 Transfer Learning

TL approach has obtained considerable adoption in various sectors and applications, and it aims to improve performance on new tasks by reusing previously trained models [53], [64], [65]. This approach can lead to significant acceleration in training and better performance on related issues [66]. Image classification is among the most extensive applications of DL and TL. By training a model on a wide and diverse set of data, it can become a generic model that can be used to learn new tasks. The learned feature maps can be utilized afterward for training a new model using different data without having to start from scratch. The fine-tuning technique is a widely used approach to tailor pre-trained models to new data. It involves unfreezing a subset of the initial layers of the pre-existing model and training the newly added layers concurrently with the final layers of the base model, for the new problem. Significant performance improvements can be achieved by fine-tuning the pre-trained features to adapt them to new data [67]. Fine-tuning is a powerful tool in the ML toolbox and has been shown to yield improved accuracy and generalization when adapting pre-trained models to new tasks or datasets.

In the field of TL, several CNN architectures can be found. MobileNetV2, DenseNet201, InceptionV3, ResNet50, and Xception are some models that will be discussed in the parts that follow:

2.5.1 MobileNetV2

The MobileNetV2 model, an improved version of the initial MobileNet architecture, uses a reversed residual design with bottleneck layers that include convolutional units [68]. Each convolutional block is supported by skip connections between the starting and ending points. By utilizing a thin skip connection technique, MobileNetV2 can retrieve earlier activations that were not modified for each convolutional block. This design improves upon the performance of previous versions while maintaining computational efficiency.

2.5.2 DenseNet201

In the realm of DL, a highly effective CNN model with a feed-forward structure, called DenseNet201, was introduced by Huang et al. [69]. In this model, each layer takes feature maps out of the previous layer and generates its own, which are subsequently passed over to the following layer. This approach drastically decreases the total amount of parameters and resolves the issue of vanishing gradients while also improving performance by repeatedly using feature maps. Nonetheless, due to its numerous layers, this model necessitates a longer training time than other DL models.

2.5.3 InceptionV3

The InceptionV3 architecture, introduced in Szegedy et al.'s work [70], is a DNN with 48 layers that improves on the original Inception model's processing capabilities by increasing its depth. InceptionV3 uses several convolutional blocks simultaneously, each one has varying filter sizes (1x1, 3x3, and 5x5), which enables it to capture both local and generalized characteristics due to its hybrid filter architecture. Additionally, the parallel computation employed by the model results in better performance than other existing CNN architectures in several applications.

2.5.4 ResNet50

The ResNet50 model was proposed to make deep network training easier by utilizing a residual learning framework [71]. This representation encompasses an identity mapping feature that helps to address problems related to vanishing and exploding gradients. The identity mapping serves as an instant link among the convolutional layers' input and output. The ResNet50 model is a popular architecture for TL because it allows for easier training than other CNN models and helps to prevent accuracy degradation across various applications, especially those related to image classification [72].

2.5.5 Xception

Google developed a CNN model called Xception, which is composed of 71 layers. It is a DL model that uses depthwise separable convolutional blocks, which are designed to extract features from input images with high efficiency and accuracy [73]. Xception uses a set of convolutional layers, inception modules, depth-wise and separable convolutional blocks, and residual links. Xception has been shown to perform well in various studies related to image classification, including remote sensing and construction applications [74], [75].

2.6 Federated Learning

FL methods adopt a distributed learning strategy where models are trained on local devices sans exposing their private data [76]. The training process is carried out using separate devices, such as smartphones or hospital computers, where the data remains confidential. FL has become an essential factor in the development of IoT applications, such as sentiment analysis, location tracking, and cyberattack detection while maintaining data privacy. These methods enable the creation of models that can efficiently adapt to changes while ensuring user privacy by keeping the training data locally and sharing only the results of analytics [77].

In this context, FL is a promising technique that has achieved great success in bringing intelligence to the edge layers of IoT devices. Involved clients individually train a global model that is initialized by a central authority using their own data. FL offers several advantages [78], including:

- FL enables edge devices to acquire knowledge via ML/DL models and preserve their training datasets without having to host them on a centralized server.
- Data is accessible on edge devices, allowing for real-time changes to AI models. This feature saves time and makes data available without requiring a connection to the centralized server.
- FL ensures data security by storing it locally on the device.
- Because of its modest complexity and decentralized structure, it is appropriate for deployment on resource-constrained devices.

2.7 Ensemble Methods

Ensembling is a strategy used to improve the performance of data analytics models. It involves merging multiple learning algorithms to leverage their combined performance and enhance the accuracy of the current model [79], [80]. The concept of ensemble learning revolves around combining various models in a way where the output model surpasses the individual models. The process of integrating decisions using multiple models has proven that it is an excellent approach to improving the overall efficiency of a model.

Ensembling techniques are a popular approach to improving model performance, with several methods proposed. Hard voting and soft voting are two common techniques where the predictions of multiple models are combined. In hard voting, the model with the highest number of votes is selected, while in soft voting, the probabilities of each model's predictions are taken into account. Another technique is stacking, where a meta-model is trained on the predictions of multiple base models. Bayesian inference is used in evidence-based theory to combine the outputs of different models. These techniques have been demonstrated to be highly effective in enhancing model accuracy and are widely used in a variety of applications.

2.8 Conclusion

This chapter provides an overview of IoT and its impact in various domains in addition to the microservices-based architecture. The fundamental data analytics concepts are discussed, along with techniques that can enhance learning, such as TL, FL, and ensemble methods. It is important to keep up with the advancements in these areas to address the challenges that arise in its development and deployment.

In the next chapter, we will delve into the suggested methods of data analytics in the IoT environment. We will explore the different techniques and algorithms that are commonly used in this domain, including but not limited to ML, DL, statistical analysis, and data visualization.

Chapter 3

Related Work: IoT Data Analytics Approaches Based on Microservices Architecture

Contents

3.1 Introduction

With the increasing number of IoT devices, huge quantities of data are being generated. However, this data is useless if not properly analyzed. There are many analytics solutions available that allow users to gain valuable insights from the massive IoT data. This chapter discusses the importance of data analytics in improving IoT applications and explores the latest research efforts focused on using microservices architecture for IoT data analytics. The proposed approaches are categorized into five groups based on their underlying techniques. Lastly, a comparison is made between these works.

3.2 The Role of Data Analytics in IoT

AI, a subfield of computer science, is concerned with the development of algorithms for processing data. Its classification has evolved into two main groups based on the extent of intelligence: weak AI and strong AI [81]. Weak AI, or narrow AI, involves categorizing data using a pre-existing statistical model that has been trained to perform specific tasks. In contrast, strong AI, also called general AI or AGI, has the ability to create a system that can work intelligently and independently by utilizing ML on any available standardized data.

ML is a powerful AI approach that gives a computer system the ability to learn and evolve from a data set while also being able to tackle problems without human intervention. This cutting-edge branch of AGI is known for its ability to utilize vast quantities of initial data, referred to as a training set, to generate statistical algorithms that can effectively analyze and act on new data. In recent years, several advanced techniques have emerged from ML, such as DL, TL, and FL, each with its unique set of benefits and applications. These innovative methods are transforming the field of AI and expanding its potential for solving complex problems in diverse domains.

The adoption of the microservices architecture for creating IoT applications has proven to be a game-changer, offering numerous benefits and advantages. By breaking down the development procedure into a set of steps, this approach enables adaptable application deployment and the development of extensible, reusable, and resilient infrastructures [82], [83]. In line with the microservices architecture, various

research endeavors have focused on integrating AI techniques into the exploration of accumulated IoT data to deliver intelligent solutions and services across diverse fields. With these cutting-edge techniques, IoT systems can analyze large quantities of data in real time, extract valuable insights, and produce informed decisions that enhance various processes' efficiency, reliability, and security. Overall, the integration of microservices and AI represents a promising pathway toward unlocking the full potential of IoT and achieving more significant breakthroughs in various domains.

In the upcoming subsection, we will delve into various proposed approaches that leverage microservices architecture and utilize a range of data analytics methods to propose intelligent IoT services.

3.3 IoT Data Analytics Approaches with Microservices Architecture

After conducting a review of relevant literature,

FL is a distributed method approach where models are trained collaboratively across multiple decentralized devices or nodes, preserving data privacy and security. TL refers to the process of leveraging knowledge learned from one task or domain and applying it to a different but related task or domain. These categories were selected to encompass a broad range of approaches in the field of IoT data analytics, representing distinct methodologies and techniques.

3.3.1 Analytics Techniques

This section explores works that employed various analytics techniques in IoT data analytics with microservices, including statistical models, ML, and DL.

3.3.1.1 Approaches Based on Statistical Models

Time series data is commonly modeled using statistical models, as noted in a study by Khedkar et al. (2021) [84]. Essentially, a statistical model is a mathematical representation that elucidates the relationship between one or more random variables and other non-random variables, according to Dai et al. (2020) [85]. By using statistical modeling on raw data, IoT data scientists can more easily analyze data and uncover correlations between variables, which in turn can be used to generate predictions. These models rely on various regression techniques and are often implemented as data analytics services utilizing a microservices-based architecture.

Ortiz et al. [86] developed a microservice-based architecture that employs a statistical model for predictive data processing. This architecture enables the system to generate accurate and compelling predictions through predictive data analytics, which can be used to take real-time actions. Specifically, the authors utilized an Autoregressive Integrated Moving Average (ARIMA) algorithm to investigate time series data, leading to improved comprehension and forecasting. Furthermore, the implementation of a microservices-based architecture has resulted in significant improvements in terms of maintenance and scalability. The authors conducted experiments on a case study related to air quality to assess the architecture's performance, yielding promising results.

Mateo et al. [87] introduced a cloud-based software platform that utilizes microservices to simplify the analytics application in a wide of IoT scenarios. To satisfy the platform's essential needs of scalability, portability, interoperability, and usability, several technologies were integrated. The resulting platform has the capability to incorporate new sensing equipment, execute data processing tasks, merge various kinds of data, incorporate sophisticated statistical model improvements easily, and give an intuitive visual display. To test the suggested software architecture, the authors conducted a smart farming scenario, demonstrating the platform's effectiveness in real-world settings.

In their work [88], Vresk et al. proposed an architecture for a microservices-based middleware that aims to connect various IoT devices that may have different properties. The proposed approach focuses on addressing the complexity resulting from differences in end-device properties by using a consistent method for representing IoT devices and applications. The approach encapsulates multiple communication

methods to enhance interoperability and extensibility to suitable microservice units. This facilitates easy change of current system components and the addition of new ones, thereby creating an open and dynamic IoT ecosystem. The authors validated their approach with a use case involving the estimation of future energy usage, utilizing a simple linear regression model to identify correlations between outcomes and predictors. Once the regression model is trained, it can be applied to predict outcomes for new data points.

The works mentioned above utilized statistical modeling for IoT data analytics. These types of algorithms are a mathematical approach that focuses on identifying correlations between variables to predict an output. However, statistical models typically rely on coefficient estimation and are often applied to smaller datasets with fewer features. Additionally, they require the human designer to clearly understand the connections between variables before inputting them into the model.

3.3.1.2 Approaches Based on Typical ML Algorithms

ML is a branch of AI that includes a set of models that can be trained using historical data to classify, identify, or uncover hidden patterns [89]. The primary advantage of using ML is that once an algorithm has learned how to handle data, it can automatically perform the same tasks. With a wealth of IoT datasets available, the need for ML is increasing. Many industries rely on ML to extract meaningful insights from data. In this context, several experiments have been conducted to teach machines to get knowledge autonomously, without specific programming, using microservices-based architecture.

In their work [90], Jamil et al. presented a secure microservice system that utilizes ML and blockchain technology to enhance the predictive modeling of IoT fitness data. The proposed system offers microservice-based analytics capabilities, which offer trusted and dependable IoT services. Data privacy, security, and identity are guaranteed by a permissioned blockchain network. In addition, the system utilizes a recommendation model that generates customized workout and diet plans for fitness users based on a set of microservices, including data preprocessing, feature extraction, model selection, and visualization microservices. This proposed design ensures good performance, accurate results, scalability, and reliability of the fitness application.

The authors of [91] suggested a microservices-based approach to provide modularized data analytics services. Data analytics is decomposed into distinct stages and created in microservices for each step. The proposed approach was implemented using a prototype and evaluated with an IoT use case scenario. They employed a set of ML models, including Support Vector Machine (SVM), Linear discriminant analysis (LDA), K-Nearest Neighbours (KNN), Decision Tree (DT), and Linear Regression (LR) to analyze and learn from the data. Results showed that the proposed approach significantly improved the efficiency of IoT applications in terms of robustness, scalability, and stability.

In their work, Ribeiro et al. [92] proposed a new architecture, Machine Learning in Microservices Architecture (MLMA), for implementing intelligent applications in smart cities. The proposed architecture is based on a set of independent microservices, and it aims to develop an ML-based process that provides high-performance analysis and prediction, efficient scalability and usability, and easy maintenance of code. Two smart city use cases, a recommendation smart tourism application and a predictive policing system, were used to validate the feasibility of the MLMA. The findings revealed that the suggested architecture could provide numerous advantages in developing ML models for intelligent applications in smart city environments.

Dineva et al. [93] proposed a sophisticated and versatile architectural framework based on microservices for the delivery of ML solutions as services. This framework's primary goal is to support analytics using a collection of modular and scalable microservices, with each one providing a specific analytics function. The framework is composed of several key microservices, including the identity service, which manages user authentication and authorization, the data acquisition service, which retrieves data from IoT devices, the transformation service, which preprocesses and transforms raw data into usable formats, the ML service, which performs the actual analytics, the device management service, which monitors and manages the connected IoT devices, and the logging service, which logs and tracks the analytics process for future analysis and debugging.

In [94], a smart dairy farming platform was introduced to analyze animal behavior and monitor their health. The platform was designed using microservices and distributed computing to solve the challenge of limited Internet connectivity in remote farm locations. By using the RF and KNN algorithms, various analytics processes were conducted closer to the data source, enabling real-time processing and monitoring capabilities. These ML models provided farmers with accurate insights and effective recommendations.

In [95], a method for automatically configuring and evaluating ML algorithms for the quick identification of Android malware was presented. This method is built on DevOps concepts in addition to microservice architecture and utilizes ML models, including RF, SVM, KNN, and MultiLayer Perceptron (MLP), in the experiments. By following this method, the authors were able to systematically identify the optimal match to enhance the performance of Android malware detection algorithms.

The utilization of microservices and ML algorithms has shown that it is highly beneficial for improving the efficiency and scalability of IoT services. However, a major limitation of these approaches is their inability to process unstructured data effectively. This requires the creation of features through manual engineering, which can be a tedious and time-consuming task. Therefore, it is imperative to explore and develop innovative techniques that can automate feature extraction and effectively handle unstructured data to further improve the accuracy and effectiveness of ML-based microservices in the context of IoT applications.

3.3.1.3 Approaches Based on Deep Learning Models

DL models have demonstrated their effectiveness in modeling and analyzing spatial and temporal variations of IoT data in several disciplines. Unlike traditional ML algorithms, DL algorithms do not rely on hand-crafted feature engineering, but instead, they can learn feature representations directly from raw data. This allows DL algorithms to accurately capture spatial and temporal correlations in the data, leading to better performance and higher accuracy. Moreover, DL algorithms have been shown to be effective in handling unstructured data, like images, videos, and text, which are commonly found in IoT applications. Therefore, DL algorithms have become increasingly popular in the field of IoT, especially in applications like image and speech recognition and anomaly detection, where the ability to capture complex patterns and dependencies from data is essential.

In [96], a novel approach called Lightweight IoT Smart Public Safety (LISPS) was introduced, which employs a microservices architecture to support the safety and improve the security of public systems. The LISPS framework is designed as a distributed solution in which sub-tasks are hosted by different microservices. The

system consists of two key processes: feature extraction from videos and decision-making. To extract features from videos, a Lightweight CNN and a Kerman algorithm were employed. The authors also proposed a fuzzy logic-based decision-making technique for detecting aberrant activities. The LISPS approach demonstrated high accuracy and efficiency in detecting and reporting critical events and could be a promising solution for enhancing public safety and security.

In [97], Houmani et al. presented an innovative architecture that aims to optimize DL workflows, reduce implementation time, and improve analytics results. This architecture was evaluated using an object detection use case and deployed on edge and cloud resources. One of the key features of this approach is the data management method which limits the resolution of incoming data. By doing so, the architecture can reduce the volume of data sent across the edge and the cloud, leading to lower latencies and faster processing times.

In their paper [98], Roy et al. introduced Micro-safe, an innovative architecture that combines the power of microservices and DL to deliver security as a service in a 6G environment. The key feature of this architecture is its ability to generate personalized safety decisions and send them to registered end-users through a DNN. Through different experiments, the proposed architecture achieved improved latency, reduced energy consumption, and increased throughput, making it a promising solution for providing safety-related services in future 6G networks.

Utilizing DL models can enable the creation of IoT solutions that can efficiently tackle specific issues. Nevertheless, DL methods often require extensive datasets to deliver accurate outcomes, especially with supervised learning approaches that demand numerous images. This large dataset requirement can significantly increase the complexity and computational expense of the training process, leading to a considerable amount of time consumption [99].

Security and privacy are two significant challenges that face DL IoT applications. DL models are sensitive to a variety of challenges that might degrade the way they perform. For instance, false data injection is a growing threat that exploits IoT equipment in various locations sending misleading readings and data, compromising the accuracy of the models [100]. Another critical challenge is privacy, where sensors capturing data in public areas, such as images of people or their behavior, may raise privacy concerns when data is moved to the cloud for investigation and processing. In response to these issues, several privacy-preserving algorithms have been proposed that use the FL approach. These algorithms aim to protect users' privacy by keeping

their sensitive data local and sharing only model updates with the cloud.

This section focuses on different learning approaches used in IoT data analytics with microservices, specifically TL and FL.

3.3.2 Approaches Based on Federated Learning

Recent studies have highlighted the use of FL as an effective approach to address privacy and security concerns in DL IoT applications. FL has emerged as a promising solution to improve analytics performance while ensuring the privacy and security of the accumulated IoT data. Several research and development studies have been conducted in this direction to develop new techniques and frameworks for efficient and secure FL-based IoT analytics.

Abdel-Basset et al. provided a Fed-TH model that leverages both DL and FL to detect cyber attacks in industrial cyber-physical systems in [101]. The model was implemented as a microservice at the edge and validated on two public datasets, TON-IOT and LITNET-2020. An exploratory microservice was also included to handle latency issues. The proposed Fed-TH model obtained an accuracy of 92.97% and 92.84% using TON-IOT and LITNET-2020, respectively, demonstrating its effectiveness in detecting cyber attacks in industrial settings.

The authors of [102] improved the capabilities of IIoT applications by utilizing decentralized training and shared learning through container microservices. They employed various strategies to decrease energy consumption and application delay, including deadline-efficient task sequencing and scheduling, latency-efficient task planning, and energy-efficient task organization. The experimental results demonstrated that this approach improved energy efficiency by 30% and reduced training time by 50%.

Zhang et al. [103] presented an FL-based platform for cyber-physical systems. This platform is able to develop smart services with ML models trained through the FL paradigm.

There is a prevalent use of standard ML/DL techniques in proposed FL methods for IoT. However, a major concern associated with FL is the need for significant

learning approaches to achieve efficient outputs. To address this challenge, one contemporary trend is adopting the TL paradigm that utilizes previously acquired knowledge to solve new problems.

3.3.3 Approaches Based on Transfer Learning

Recently, the application of TL in IoT has received significant interest because of its particular characteristics and strengths. TL has the ability to leverage knowledge learned from a huge amount of data to address new challenges and problems with a small amount of data [104]. This has resulted in the proposal of several works that utilize TL to improve IoT applications.

In the context of smart surveillance, [105] proposed a microservices-based application with AI to analyze massive quantities of images from criminal evidence. The TL paradigm was used with pre-trained CNN architecture to classify those images and deduce crucial information (such as weapons, ammo, identity cards, etc.).

The authors of [106] have proposed a framework for supporting data analytics and maintenance procedures in industrial settings. Microservices and techniques for visualization are used to offer great levels of customization. The framework's design is organized into a collection of microservices, each performing a specific function. The platform can handle large amounts of data and provides adaptable data storage and management capabilities. Additionally, trained models are made available as a service to facilitate the use of predictive analytics.

In their work, Chen et al. [107] provided a re-identification system based on microservices, which performs identification calculations on edge computing to ensure efficiency and privacy protection. The system was designed to be flexible and accommodate various situations and demand loads. The experiments showed that the proposed design provided enough computing resources for Re-ID and ensured flexibility.

The application of TL has demonstrated significant improvements in delivering smart IoT services. However, most of the proposed methods are built on centralized structures in which devices transfer their locally generated data to cloud-based storage facilities or powerful servers for computing. Such centralized approaches may raise concerns regarding data privacy and security. Alternately, leveraging TL

and FL approaches can efficiently support IoT data analytics while preserving data privacy by performing computations locally on edge devices or using distributed computing across multiple edge devices.

3.4 Comparison of The Microservices-based IoT Data Analytics Approaches

For the purpose of comparing microservices-based IoT data analytics approaches, we have established the following criteria: the domain of the IoT application, the computing paradigm employed, the types of the data analytics performed, the analytical task carried out, the data analytics techniques utilized, and the key contributions of each approach.

Table 3.4 provides an overview of the reviewed studies based on microservices-based architecture/data analytics in IoT. The chosen criteria for comparing the related works were selected to provide a comprehensive analysis of the approaches. Each criterion was specifically chosen to highlight different aspects of the approaches and facilitate a thorough comparison. We start by the domain of the IoT application. This criterion focuses on the specific application domain to which each approach is applied. It helps understand the breadth and diversity of IoT applications addressed by the approaches and highlight their applicability in different domains. The computing paradigm emphasizes the paradigm adopted by each approach. It could include cloud computing, edge computing, or a combination of both. Understanding the computing paradigm used provides insights into the scalability, latency, and resource utilization aspects of the approaches. The types of data analytics performed focuses on the specific types of analytics performed by each approach. It could include descriptive analytics, diagnostic analytics, predictive analytics, or prescriptive analytics. By analyzing the types of analytics employed, we can gain insights into the depth and sophistication of the approaches' analytical capabilities. The analytical task carried out delves into the specific analytical tasks accomplished by each approach. It could include anomaly detection, classification, clustering, regression, etc. Understanding the analytical tasks undertaken by the approaches helps evaluate their suitability for different data analysis objectives. The data analytics techniques criterion highlights the specific techniques employed by each approach. It could include statistical models, ML algorithms, DL architectures, or other specialized techniques. Assessing the techniques utilized sheds light on the sophistication

and potential effectiveness of the approaches. The last criterion, key contributions, focuses on identifying the unique contributions of each approach. It helps to understand the novel aspects, innovations, or advancements introduced by the approaches in the field of IoT data analytics following the microservices-based approaches.

	Work	IoT Application domain	Computing paradigm	Type of data analytics	Analytical task	DA technique	Key contributions
Statistical models	[86]	Air quality	Centralized computing	Predictive analytics	Event pattern detection, real-time prediction, context awareness	ARIMA model	
	[87]	Smart Agriculture	Centralized computing	Descriptive/ Predictive analytics	Estimate the pigs fattening growth, Send alerts to farmers	Sigmoid function from the Gompertz family, A cubic polynomial function	+Facilitate digital transformation in agricultural fields.
	[88]	Smart environment	centralized computing	Descriptive analytics	Estimation of future energy consumption	Regression model	+An interoperability and extensibility architecture for IoT ecosystem
ML	[90]	Healthcare	Centralized computing	Prescriptive analytics	Recommendation of diet and workout plan	SVM, KNN, LR	
	[91]	Healthcare	Centralized computing	Predictive analytics	Classification of positive or negative diabetes patients	SVM, KNN, DT, LR	
	[92]	Smart City	Centralized computing	Predictive analytics	Policing system	ML	+Reusability of code. +The ease of separate services maintenance.
	[94]	Smart agriculture	Distributed computing (fog)	Perspective analytics	The analysis of animal behaviour and their health monitoring	KNN	+Precise insights and effective recommendations
	[95]	Cyber security	Centralized computing	Predictive analytics	Android malware detection	RF SVM, KNN MLP	+Increase the detection rate. +Optimizing resource use.

DL	[96]	Smart surveillance	Distributed Computing (fog+edge)	Predictive analytics	detect, track objects and identify suspicious activities	L-CNN	+Minimum delay of video processing +Easy to deploy and configure.
	[97]	Smart city	Distributed Computing (cloud+ fog+edge)	Predictive analytics	Object detection in images and videos	YOLO algorithm	+Efficient way of data management. + Simple workflow to implement a DL application
	[98]	Smart transportation	Distributed computing	Predictive analytics	Address the safety problems in transportation services.	DNN	+Improved latency, +Reduced energy consumption, +Increased throughput
	[108]	Smart city	Centralized computing	Descriptive analytics	Feature detection and image matching	CNN	+A low error rate, An excellent response time.
FL	[101]	Industrial threat hunting	Distributed computing	Predictive analytics	Cyber-threat detection	LSTM-AE	+Efficient cyber threat detection, +Preserve participants privacy
	[102]	Healthcare, smart home, transportation	Distributed computing	Descriptive analytics	Provide different IoT services using ML	ML	+Produce the best results in terms of latency and power consumption
	[103]	Cyber security	Distributed computing	Predictive analytics	Anomaly detection	ML algorithm	+Ensure the development of trustworthy Edge Computing
TL	[105]	Smart Surveillance	Distributed computing	Predictive analytics	The identification of crimes	YOLO, IneptionV3, Nudenet	+A flexible architecture, +Speed up the image classification process.
	[106]	Smart manufacturing	Distributed computing (edge)	Predictive analytics	Support smart predictive maintenance operations	Pretrained analytical models	+A lightweight, resilient, and scalable architecture + Enable data analytics on the edge
	[107]	Smart surveillance	Distributed computing (edge)	Predictive analytics	Person identification	OSNet,ResNet Inception, MobileNet SqueezeNet	Low-latency performance

By conducting a thorough analysis and comparison of the related works based on the defined criteria, we gain a comprehensive understanding of the various dimensions of these approaches, and we deduce the following limitations:

- Many of the examined works focus on specific types of data analytics, overlooking the potential benefits of integrating multiple data analytics techniques.
- Some approaches exhibit a lack of flexibility, particularly in terms of accommodating different types of IoT devices, data formats, and application domains. This rigidity can restrict the applicability and adaptability of analytics solutions in diverse IoT environments.
- Several approaches rely on a centralized architecture for data processing, which can introduce potential bottlenecks and single points of failure. The centralized nature may also lead to increased latency and network congestion in scenarios with large-scale deployments.
- While some works prioritize data protection, there is a need for a more comprehensive focus on privacy concerns in IoT applications.
- In some of the existing works, privacy concerns are addressed through the use of FL, which ensures data privacy by keeping sensitive information locally on edge devices. However, this focus on privacy often comes at the cost of sacrificing the performance of the analytical models.
- The reviewed works largely concentrate on a single type of data analytics or learning technique, failing to explore the advantages of hybrid architectures that combine various approaches. By adopting a more comprehensive and integrated approach, we can harness the strengths of different techniques and achieve enhanced performance in IoT data analytics.

At the end of this literature review on the utilization of microservices architecture for IoT data analytics, we can draw the following conclusions:

- ML and DL algorithms have demonstrated their effectiveness in modeling and analyzing the spatial and temporal fluctuations of IoT data in various domains. By learning feature representations, these algorithms can accurately capture spatial and temporal patterns. The model can distinguish between different features and provide precise results due to its ability to learn unique features

across multiple classes rapidly. Furthermore, these data analytics methods can account for the spatial and temporal diversity of the environment and efficiently classify different types of learned information related to the problem, even when the surroundings or time changes.

- TL is becoming increasingly popular in IoT data analytics research due to its ability to efficiently leverage previously gained knowledge to address new problems. TL requires fewer data and resources during the model training phase than traditional learning methods such as supervised, unsupervised, or reinforcement learning. The table shows that TL gives several improvements, including enhanced service performance, reduced effort and cost, improved accuracy, and support for real-time data analytics. Being able to handle real-time data analytics is particularly important for time-sensitive applications in the IoT environment, such as smart parking scenarios and elderly care applications.
  Experimental results of the reviewed approaches suggest that the features combination of multiple TL methods can achieve superior outcomes than using just one TL method for the same data. This is because different models may have different strengths and weaknesses, and by combining them, their strengths are utilized to enhance the system's overall performance.

- FL is a distributed approach to training ML models in which the training data is kept locally on the user's device, and the model is updated with the help of a central server without transferring the raw data. This approach is gaining popularity due to its ability to ensure data privacy and improve the performance of the analytical process. In the FL approach, the model is trained on the client side, and only updates are sent to the central server, where the updates from all clients are aggregated to generate a global model. FL allows for the efficient use of distributed resources and can handle huge quantities of data without the need for a centralized storage unit. This approach is particularly useful in scenarios involving sensitive data, such as healthcare or finance, where data privacy is crucial. Moreover, FL may be used for IoT data analytics to reduce latency and ensure data privacy while providing accurate results.

- Some of the studies analyzed aimed to improve data analytics capabilities by implementing microservices architecture. In one study [91], the researchers proposed breaking down the data analytics process into a set of steps that can be implemented using composable microservices. This approach improves the efficiency of smart data analytics services by allowing them to handle diverse

IoT data from various sources and process vast amounts of data. Additionally, these services are modular and flexible, enabling IoT systems to scale with analytic features. The other reviewed works aimed to improve specific IoT data analytics case studies in various application domains, such as smart surveillance, smart cities, manufacturing, and smart farming.

- The main objective of using microservices-based architecture is to suit the needs of end-users. This approach helps to improve the usability, personalization, and contextualization of IoT applications. The studies reviewed in this article have utilized microservices-based architecture to address the needs of end-users. Some of the research works, including [109], [110], have focused on Quality of service (QoS) requirements that microservices-based architecture must meet. As this architecture is dynamic and scalable, various QoS assurance mechanisms have been developed to ensure the performance, security, and privacy of the applications during their execution. Some studies, such as [101], [106], have investigated solution-oriented criteria to enable decentralized computing and stream processing and facilitate device management.

Based on current research on IoT, microservices, and data analytics, we discovered a research need in microservices-based architecture to support data analytics with effective performance and high privacy. Combining microservices-based architecture, TL, and FL with IoT applications boosts performance, privacy, speed, scalability, integrity, and throughput.

3.5 Conclusion

This chapter is devoted to the state-of-the-art on proposed approaches for data analytics in IoT environments. We introduced five categories of analytics methods developed through microservices architecture, including approaches based on statistical models, ML, DL, FL, and TL. We created a comparative analysis between the reported studies and several conclusions have been drawn: 1) the use of the TL paradigm improves the learning process benefiting from the accumulated knowledge, 2) the employment of FL has boosted the efficiency of the knowledge-sharing process and increased accuracy while preserving data privacy, 3) the adoption of microservices architecture improves the functioning of IoT applications, particularly in terms of availability and speed of response.

In the next chapter, we will propose a new approach for data analytics in an IoT environment. This approach combines both the TL and FL paradigms and follows the microservices-based architecture.

Chapter 4

Proposed Approach: An Effective Analytical Solution Based On Microservices For IoT Applications

Contents

4.1 Introduction

In this chapter, we introduce a novel approach to data analytics in IoT applications. Our approach is rooted in the microservices architecture and leverages a variety of advanced AI techniques to deliver great analytical performance, unparalleled privacy protection, and exceptional scalability.

We begin this chapter by introducing the architecture of the proposed approach. Next, we present how we have used different ensemble methods to improve the generalization of the analytical model. After that, we explain how we have optimizing the scalbility of the analytical process using microservices, highlighting how it is designed to effectively process and analyze IoT data.

4.2 Proposed Approach Architecture

This work aims to develop an intelligent approach that facilitates IoT data analytics. The approach is implemented using a set of microservices that are designed to be secure, flexible, scalable, and highly responsive to user demands. Each microservice provides a functionality independently. The microservices are merged via a pipeline of multiple microservices to work together and achieve the overall service objectives. To do this, the data analytics functionality is divided into discrete containerized microservices, and the analytics models are trained using appropriate AI methods before being deployed on edge computing nodes.

The proposed approach is a novel solution that is divided into two main nodes: the cloud node and the edge node, as depicted in Figure 4.1.

This architecture has been developed with the flexibility to support both centralized and distributed learning, depending on the specific requirements of the application. While the architecture is primarily designed for centralized learning on large datasets, it has also been explored for distributed learning. Centralized learning involves training the model on a large dataset that is aggregated in a central location. This approach is well-suited for scenarios where the dataset can be easily collected and managed in a centralized manner, and where the computing resources required for training the model are available in a single location. Distributed learning involves training the model on multiple machines, with each machine processing a subset of training data. This method of learning is more suitable for scenarios where the dataset is too large to be managed in a centralized manner, or where the computing resources required for training the model are distributed across multiple locations. In summary, the proposed architecture is versatile enough to support both centralized and distributed learning, and the choice between the two approaches depends on a number of parameters, such as the size and complexity of the dataset, computing resources availability, and the specific requirements of the use case.

In order to achieve highly efficient IoT data analytics with enhanced performance and increased scalability, the approach has been developed with a range of functionalities that are summarized below. Each component of the approach has been assigned specific responsibilities to ensure seamless integration and optimal performance. This intelligent approach is proposed to enhance IoT data analytics by providing highly efficient and scalable solutions. By leveraging the cloud and edge nodes and supporting both centralized and distributed learning, the proposed approach is well-suited for a large set of use cases.
The following subsections will outline the analytical process for both centralized and distributed learning methods.

4.2.1 Centralized Learning Method

Centralized learning in data analytics refers to a system where data is gathered from various sources and stored in a central location, such as a data warehouse, before being analyzed. It has several benefits. First, it allows for the processing of big amounts of data that may be difficult or impossible to analyze locally. Second,

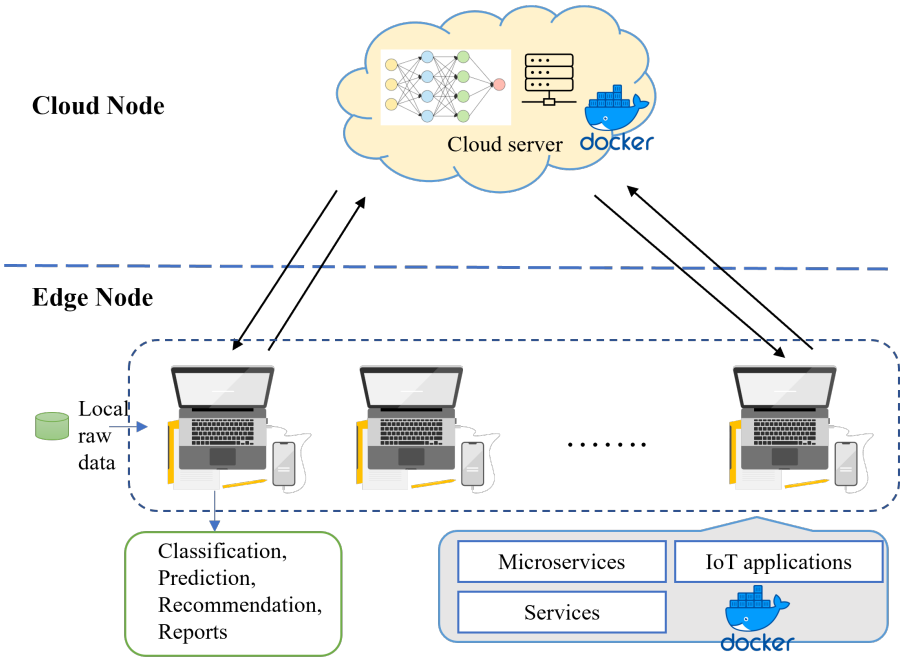


FIGURE 4.1: Abstract architecture of the proposed approach design

it can be more efficient and cost-effective than analyzing data locally, particularly when dealing with large datasets. Finally, centralized learning can lead to more accurate results, as the central location can leverage more powerful hardware and algorithms for data processing.

In this context, the data analytics process is performed through cloud nodes and edge nodes. The cloud node refers to a powerful remote server that is typically located in a data center or a cloud computing infrastructure. It is used for storing and processing big quantities of data and is often used for training ML/DL models. It is particularly useful when dealing with large datasets that require significant processing power to analyze. This node is responsible for data storage, preprocessing, model learning, and analytics.

On the other hand, an edge node refers to a local device that is connected to a network and is used to collect data from sensors or other devices. Edge nodes are often presented as IoT applications, where data is collected from multiple sensors located in various places. Edge nodes are typically less powerful than cloud nodes, but they are closer to the data source and can process data more quickly. In a centralized learning setup, data is gathered from edge nodes and transferred to a

cloud node for processing and analysis. The analytical findings are subsequently relayed back to the edge nodes, which can utilize the information to make decisions or take action.

In the proposed approach, to perform data analytics with a centralized learning method, these steps are followed:

1. Data collection: data is collected from different sources, like databases, websites, and social media platforms.
2. Data storage: data is stored in a central location, like a data warehouse, where it can be accessed and analyzed easily.
3. Data preprocessing: data is preprocessed to ensure that it is clean, consistent, and formatted correctly for analysis.
4. Data analytics: data is analyzed using various data analytics methods, like descriptive analytics, predictive analytics, and prescriptive analytics.
5. Data Visualization: the results of the analysis are illustrated using graphs, charts, and other visualization tools in order to understand the insights that have been generated.

This method is particularly useful when dealing with real-time data that requires quick analysis and decision-making. However, there are also some potential drawbacks to centralized learning. One concern is that it can raise privacy and security concerns, particularly when sensitive or personal data is being collected and analyzed. Additionally, centralized learning can be less resilient to system failures or attacks, because one single point of malfunction may bring the whole system down. The distributed learning method is proposed as a solution for some of the drawbacks of the centralized learning method.

4.2.2 Distributed Learning Method

Distributed learning is an approach to ML that involves training models on multiple devices or computers. This can be particularly useful for data analytics when dealing with big quantities of data which can not be processed using a single machine.

There are several benefits to distributed learning for data analytics. By leveraging multiple machines, distributed learning can decrease the required time to analyze large amounts of data and improve the accuracy of models by leveraging more data. Additionally, distributed learning can improve fault tolerance and reduce the risk of data loss by replicating data across multiple machines.

There are several approaches to distributed learning for data analytics, including parallel computing, distributed computing, and FL. We employ FL which is a newer approach that involves training models on devices like smartphones or other edge devices. The data stays at devices, and model parameters are transmitted back to the cloud server for aggregation. This approach is particularly useful for privacy-sensitive applications where the data cannot be sent to a central location.

The proposed approach for distributed computing employs intelligence coordination through the edge and cloud nodes. The functions of these nodes are described in the following.

4.2.2.1 Edge Node

In FL, the edge nodes are machines such as smartphones, tablets, or IoT devices that have local data and computational power. The role of edge nodes in FL is to participate in the training of ML/DL models without needing to transfer data to a centralized cloud server. Rather than transmitting data to a centralized server for training, edge nodes utilize their local data for model training and then send the model's updates to a centralized server. This process allows for the ML/DL models to train on decentralized data, without exposing sensitive data to external entities.

Edge nodes play a critical role in FL as they enable the training of models on data that is placed at the edge, near where it is generated. This approach reduces network latency, saves bandwidth, and protects data privacy, making it ideal for use in various applications such as mobile devices, smart homes, and IoT devices. In our suggested approach, the edge layer is a complex yet dynamic system that consists of several layers of service-oriented microservices that work in tandem to form an intelligent IoT application. The microservices are linkable and capable of inter-communicating to execute specific tasks. Figure 4.2 illustrates the edge layer in the distributed learning method and its underlying structure.

To better understand the functionalities of the edge layer, it can be summarized as a layered service-oriented system that performs tasks ranging from data acquisition, preprocessing, model learning, and interpretation. These tasks are carried out through a series of interconnected microservices that work together to provide intelligent decision-making capabilities. The layers of the edge layer are presented below:

- **Data Collection**

  Raw data with varying formats and types are generated from a multitude of smart devices like sensors and actuators. These objects collect data and information from their environment, with sensors providing real-time data that enables communication between physical objects and digital networks. The functions of various types of sensors vary depending on the goals of IoT applications. Wearable sensors provide precise information on human actions, whereas alternative sensors gauge various variables, including temperature, humidity, pressure, air quality, duration, time, velocity, motion, and pulse rate, among other factors. Despite the vast amounts of data collected, it is unstructured and comes in different formats and types. To make it usable, the data must be preprocessed, normalized, and filtered.

- **Data Preprocessing**

  Data preprocessing is an important step in the procedure of preparing data for further processing with a learning model. It involves applying various techniques to the raw data collected by multiple sensors and objects in different formats and types. The main methods used in preprocessing include data integration, normalization, transformation, and augmentation. The data integration function is used to combine heterogeneous data into a standardized format, rectify outliers and inconsistencies in the data, and smooth noisy data. Normalization is used to turn data into dimensionless and similar distributions. The dataset features are usually normalized between 0.0 and 1.0. Data transformation is employed to convert accumulated data into a format that matches the input type of the ML/DL model. Data augmentation is used to increase the size of the training dataset and improve the generalization abilities of the training model [111]. Different augmentation strategies such as random cropping and resizing, horizontal and vertical flipping, random rotation, color jittering, and adding noise are used for data transformations.
  Overall, these preprocessing techniques help to prepare the collected raw data to be used with ML and DL models.

- **Model learning**

  The model learning layer is a critical component of ML systems that is responsible for creating and refining the model. In an FL setting, this layer communicates with the cloud node to initiate the training process based on the settings defined in the cloud server. The cloud node provides the initial model to the model learning layer, which then begins training the model using the data stored on the local device.

  The model learning layer is in charge of implementing the appropriate learning algorithms to optimize the model for the specific use case. This includes selecting the appropriate loss function, defining the learning rate, and selecting the optimization algorithm. Additionally, the model learning layer may incorporate techniques such as regularization, early stopping, and hyperparameter tuning to improve model performance. It is essential to monitor the learning process and track the model's progress to ensure that it is converging toward the desired outcome. The model learning layer is a crucial component of ML systems and plays a significant role in developing accurate and reliable models that can be used to make informed decisions.

- **Application**

  This phase presents the process of using a trained ML model to produce predictions on new or unseen data. In other words, it is the process of using a model to infer the output or outcome of a given input data. During the training phase, an ML model learns patterns and relationships in the training data, which are used to make predictions on the new data during inference. The trained model can be used to make various types of predictions, depending on the problem being solved. For example, in a classification problem, the model can be used to predict the class label of a new data point. In a regression problem, the model can be used to predict a continuous value.

  During inference, the new data is fed into the trained model, which then processes the data and produces a prediction. The output of the model can then be used for various purposes, such as decision-making, reporting, or recommendations. It should be noted that the accuracy and effectiveness of the predictions made by a model during interpretation depend on the quality of training data, the model complexity, and the inference data quality. Therefore, it is important to continually track and assess the model performance during inference to ensure that it is producing accurate and reliable predictions.

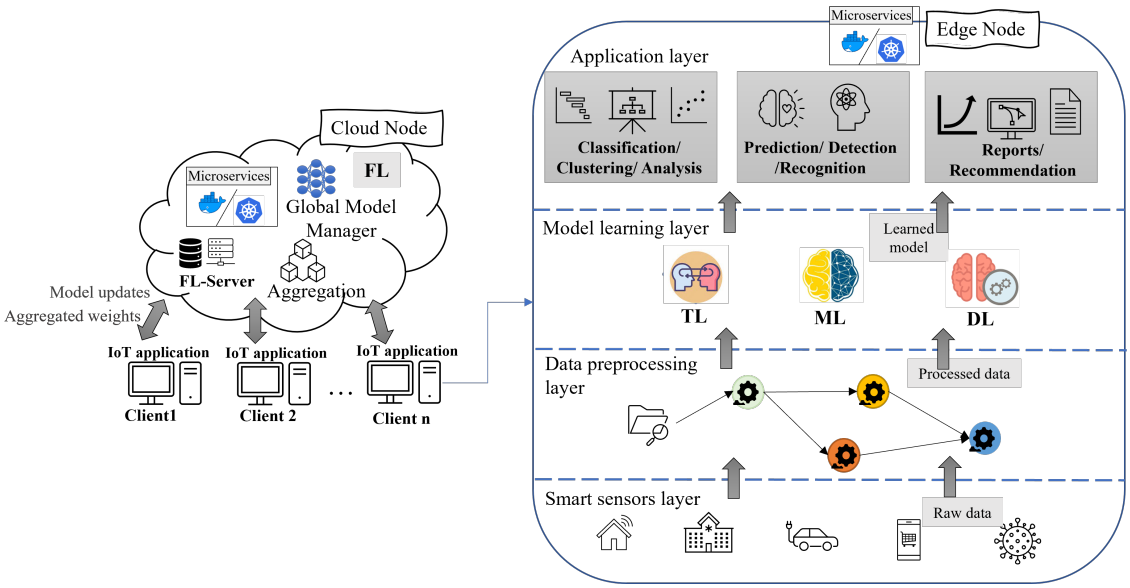


FIGURE 4.2: The Edge Node architecture in the decentralized learning method.

4.2.2.2 Cloud Node

The cloud node is located in the cloud and is responsible for processing functions from edge nodes. The model-building process occurs on edge and cloud nodes. In the centralized view, the entire model construction process occurs on the cloud, necessitating the movement of data from the edge to the cloud. This approach raises privacy concerns, as sensitive data could be compromised during data transmission.

However, by employing the FL paradigm, the proposed approach addresses these concerns. The cloud node aggregates the inference models from the edge nodes and performs evaluation before delegating the final models to the edge nodes. This approach ensures that model building occurs in a decentralized and secure manner, where data remains on the edge nodes and is not transmitted to the cloud. The proposed approach improves the coordination between edge and cloud nodes, increases efficiency, and ensures data privacy.

- **Model Manager**

The model manager is a key component in the FL pipeline that has an important role in the development, training, and deployment of ML/DL models. It is responsible for overseeing the entire model lifecycle, from the initial creation and selection of the appropriate model architecture to its deployment and evaluation. The model manager, in turn, is responsible for utilizing these resources effectively by handling model initialization, training configuration, and evaluation. In addition to model selection, the model manager is also essential in the development of the initial model architecture and its compilation. This process requires a deep understanding of the IoT use case and the specific data that will be utilized for training.

Once the local models trained by the clients have been aggregated, the model manager synthesizes them into a new global model that is more accurate and efficient than any individual local model. It then deploys these models to production and monitors their performance to ensure that it continues to perform optimally. The model manager is a highly skilled and specialized component that is essential to the success of any ML project. Its ability to handle the complex tasks of model selection, development, and deployment is critical to ensuring that ML models are accurate, efficient, and effective in solving real-world problems.

4.2.2.3 Federated Learning Process

FL is an effective paradigm for training ML models in a distributed environment. It involves the use of multiple virtual instances, located at the edge, to train models using their respective training data. The updated parameters are then sent to a central server in the cloud for aggregation, which allows for the creation of a final global model that incorporates the insights gained from all of the individual instances.

This final global model is then deployed on the edge for use in analytics. Ensemble approaches can be used to increase the model's reliability and accuracy even more.By leveraging the power of distributed learning and ensemble techniques, we can develop more accurate and reliable ML models that can drive better insights and decision-making.

The working process of the proposed FL is described in the following steps:

1. Data preparation: Each client collects and prepares its local data for training.
2. Model initialization and distribution: The initial ML/DL models are initialized in the cloud server and shared with the edge IoT clients.
3. Local training: Upon receiving the models, each client trains the models locally on its own data for multiple epochs.
4. FL Aggregation: The new weights of the locally trained models are uploaded to the cloud server to be aggregated.
5. Global model construction: The weights of the models are aggregated in the FL server to construct new global models. The FL training process (Steps 3 and 4) is repeated for multiple communication rounds until the models reach an effective performance.
6. Global model distribution: The new global model weights are transferred back to the IoT devices, replacing the existing local models with the final global models.
7. Interpretation: Analytics operations are performed on the client's local data using the final global models.
8. Fusion ensembling: A fusion ensembler is developed to receive the predictions from the final global models and select the final output.

4.2.2.4 Models development

The proposed approach for distributed ML involves constructing the model using both cloud servers and edge devices. Firstly, the model is created and initialized by the model manager in the cloud. Then, the learning process is carried out collaboratively through the edge clients using their own data.

The model-building process follows a three-step method. Firstly, system initialization and device selection take place, where the cloud server initializes the ML/DL model and chooses the edge devices to collaborate in the training procedure. Secondly, decentralized learning and updates occur, where selected edge devices do local training using their respective datasets, and transfer the updated model weights to the cloud server. This step is repeated for multiple communication rounds until the model performance is satisfactory. Finally, model aggregation and download

take place, where the cloud server aggregates the updated model weights received from edge devices to construct a novel global model. The new global model is then downloaded and distributed back to the edge devices for further use.

1. **Model Selection and System Initialization**

   The model manager selects an IoT job, such as intrusion detection or disease classification, and creates the appropriate ML/DL models. Multiple models can be selected to be trained using the clients' data. Additionally, TL techniques can be employed to leverage previously learned knowledge and provide models with better performance.

   Depending on the use case, the type of models to be used is identified, and the learning rates and communication rounds are selected. These parameters are crucial in ensuring that the models are trained effectively and efficiently. The IoT devices are also selected to participate in the FL process as clients.

   By selecting the appropriate models and configuring the learning parameters, the model manager can ensure that the FL process is optimized for the specific application case. The use of TL can also help to enhance the models' performance, while the selection of the IoT devices as clients can ensure that data privacy is maintained.

2. **Decentralized Learning and Updates**
   Once the model manager has configured the learning parameters and selected the appropriate models, the cloud server initializes the global models and transmits them to clients to begin the FL process. Every client, denoted as k, employs its own local models for training using its own dataset, denoted as D_k, and calculates a weight, denoted as w_k, through reducing the loss function $F(w_k)$:

   $$w_k = argminF(w_k), k \in K \tag{4.1}$$

   Subsequently, every client k transfers its calculated weight w_k to the server to be aggregated. Through the training process, the clients just share model weight updates with the server, ensuring that data privacy is maintained.

3. **Models Aggregation and Download**
   The cloud server combines locally trained models' weights from participating clients in each round and produces a new version of the global model. The communication rounds number in an FL process refers to how many times this aggregation technique is executed, i.e., how many times the server collects the

locally trained models, combines them, and refreshes the global model. Each round typically involves several communication iterations, during which the IoT devices send their locally learned weights, and the server sends back the updated global model. The communication rounds number in FL depends on various factors such as the convergence rate of the models, training data size, IoT devices number, and communication bandwidth.

In each training cycle, local weight updates provided by clients in FL are merged. Federated Averaging (FedAvg) is the most fundamental aggregation function in FL [112]. FedAvg updates the global model by averaging the weights provided by clients. Suppose that $W = (w_i)$ denotes the new aggregated weight, and $W^k = (w_i^k)$ denote learned weights by each client k, then:

$$w_i = \sum \frac{d_i}{D} w_i^k, \tag{4.2}$$

where D presents the whole data size and d_i presents the data size for each client. Technically, the aggregation process is done according to equations (4.3) and (4.4).

$$F_k(w) = \frac{1}{d_k} \sum_{i \in P_k} f_i(w), \tag{4.3}$$

$$f(w) = \sum_{k=1}^{K} \frac{d_k}{D} F_k(w). \tag{4.4}$$

where K denotes the number of included clients, the loss of prediction of sample $(x_i; y_i)$ is denoted as $f_i(w) = l(x_i;\ y_i w)$, and P_k is a collection of data sample indices for client k. Equation 4.3 determines the weight parameters of all devices based on the loss values experienced across training data points. Equation 4.4 scales the parameters and adds them all up component-by-component.
By exchanging just trained models' parameters rather than local data, this strategy decreases the device's communication overhead and ensures privacy.

Overall, the FL process allows for collaborative learning across multiple IoT devices, without the need for centralized data storage. By allowing each client to train on its own local data, the privacy of the data is maintained, while still allowing the development of accurate and effective models for a wide range of IoT applications.

4.3 Improving The Generalization of The Analytical model

Ensemble learning is a powerful technique that can significantly enhance the performance and robustness of ML models [113]. To further enhance the proposed approach, we choose to explore different ensemble learning techniques of stacking. Stacking can improve performance by combining the predictions of multiple models [114]. Additionally, it is essential to carefully select the DL models and their architectures to ensure that they complement each other and provide diverse predictions.

Besides, we employ different TL models for ensemble learning. TL is a technique commonly used in DL to leverage the knowledge learned by pre-trained models on a different but related task. In the field of computer vision, CNNs are commonly employed, and there are several architectures available, including MobileNetV2, DenseNet201, and InceptionV3. The mentioned DL models are excellent choices and can be integrated into different forms to generate an ensemble. Different architectures and combinations can be performed to find the suitable ensemble for a given use case.

Finally, it is crucial to assess the efficiency of the ensemble learning model using appropriate measurements, like accuracy, precision, recall, and F1-score. This will aid to identify any weaknesses in the model and guide further improvements.

The steps for developing an ensemble learning model are summarised in the following points:

1. Data preprocessing: Prepare the data set by cleaning, pre-processing, and feature selection.
2. Model Selection: Choose a set of base models that are diverse in terms of their algorithms, hyperparameters, and training data.
3. Model Training: Train each of the selected models.
4. Model Evaluation: Evaluate the trained models on a validation set to determine their individual strengths and weaknesses.
5. Ensemble Method Selection: Select an appropriate ensemble method, such as majority voting.

6. Ensemble model inference: Use the ensemble model to combine predictions.
7. Model Evaluation: Evaluate the efficiency of the ensemble model to determine its overall accuracy and other performance metrics.
8. Deployment: Following the completion of training and fine-tuning, the ensemble model is deployable in a production environment for actual utilization.

4.3.1 Decision Fusion Using Ensemble Strategies

After completing the training process, the final models are ready to be utilized for data analytics and interpretation. To validate their performance and accuracy, new data is used to test these models. However, combining the outputs of multiple models can result in a more accurate prediction. This is where ensemble learning comes in.

Ensemble learning is an excellent strategy for improving performance by combining the predictions of numerous analytical models [79], [80], [115]. This strategy relies on the concept combining the outputs of multiple models can provide better results than using a single model alone. To implement ensemble learning, different methods such as majority voting, and evidence theory can be used to merge final models' predictions.

4.3.1.1 Hard Voting

Hard voting is a simple and popular ensemble learning technique used for combining outputs generated by multiple classifiers. This method works by choosing the plurality decision generated from different classifiers [115]. In this approach, each classifier predicts the class of the input data, and the outcomes are stored in an array as follows: $[\mathcal{O}_1(x), \mathcal{O}_2(x), \ldots, \mathcal{O}_n(x)]$, where n denotes the total number of used classifiers.

To determine the final output class, hard voting applies a voting mechanism that selects the class with the highest frequency of prediction in the vector. Specifically, the category with the greatest number of voters is chosen to be the final output class for a given data instance. Mathematically, the final determined class y for input is

determined as follows:

$$y = \arg \max_{i \in 1,2,...,C} \sum_{j=1}^{n} I(\mathcal{O}_j(x) = i), \tag{4.5}$$

where C presents the involved classes number, $\mathcal{O}_j(x)$ is the output predicted class label of the j_th classifier for the input data x, and I stands for the indicator function that returns 1 if the prediction is correct and 0 otherwise.

4.3.1.2 Soft Voting

Soft voting is a classification technique that combines the predictions of multiple classifiers by averaging their probabilities for each class [115]. Given a set of classifiers $\mathcal{C}l = cl_1, cl_2, ..., cl_j$ which work for a multi-classification problem, soft voting obtains the mean probability for every class following the Equation 4.6.

$$\mathcal{P}_{mean}(i_j|x) = \frac{1}{n} \sum_{z=1}^{n} Pcl_z(i_j|x) \tag{4.6}$$

In this equation, $\mathcal{P}_{mean}(i_j|x)$ represents the average probability of class i_j for a given input x. It is calculated as the sum of the probabilities $Pm_z(i_j|x)$ produced by each classifier m_z for class i_j, divided by the total number of classifiers n.

Finally, the output class $\mathcal{Y}$ of an input x is selected by Equation 4.7. This equation selects the category that has the greatest average probability among the available classes. The argmax function returns the index of the class that obtains the highest probability value, which corresponds to the final selected class for classification.

$$\mathcal{Y} = argmax[P_{mean}(i_0|x), ..., P_{mean}(i_j|x)] \tag{4.7}$$

4.3.1.3 Random Forests-Based Voting

RF classifier is a powerful tool for combining multiple ML/DL models (M_j) to improve predictive accuracy [116]. Given an input network data x with n columns, each model m_j predicts a set of classes probabilities $Pr = pr_1, pr_2, ..., pr_n$. By employing the RF algorithm, the probabilities of the M_j models are integrated

through an ensemble prediction function $f(x)$.Equation 4.8 illustrates this process, where each model's likelihood is viewed as a vote. The label with the highest level of certainty is produced as the output.

$$f(x) = argmax_{y \in Y} \sum_{j=1}^{J} I(y = p_i(x)) \tag{4.8}$$

where Y denotes the total number of classes and y a single class.
The benefit of using the RF classifier is that it can combine the strengths of multiple models, thereby improving overall predictive accuracy. Additionally, it is able to handle large datasets and avoid overfitting, making it a robust tool for data analysis. By utilizing the RF classifier to combine probability values, it is possible to create an accurate and reliable prediction function that could be employed in a wide range of data types and use cases.

4.3.1.4 Evidence Theory

The Dempster-Shafer fusion theory is a widely utilized technique for enhancing decisions obtained from different classifiers by generating stronger and more accurate conclusions. The evidence theory is a useful decision-making technique, which handles the uncertainty associated with data, allowing for the modeling of both aleatory and epistemic uncertainty. Mass (m), belief (Bel), and plausibility (Pl) functions are utilized in this theory.

As defined by Equation 4.9, the mass function is a crucial element of this evidence theory. For any $S \subset D$, $m(S)$ represents the mass function of the proposition that precisely expresses the fundamental level of belief and support attributed to suggestion S. $D = (P_1, P_2, \ldots, P_n)$ represents the sets of predictions from the final global models.

$$2^D \rightarrow [0.1], m(\emptyset) = 0, and \sum_{S \subset D} m(S) = 1, \tag{4.9}$$

The Bel function exhibits a constant rise in value on $2^D \rightarrow [0.1]$, with Bel($\emptyset$)=0 and Bel(D)=1, as defined by Equation 4.13. Plausibility functions are then defined as $Pl(S) = 1 - Bel(S)$, where $Pl(S)$ and $Bel(S)$ are the highest and lowest boundary

functions of S, respectively.

$$\forall A \in 2^D, Bel(S) = \sum_{B \subseteq S, B \neq \emptyset} m(B) \tag{4.10}$$

To merge features and decisions from different DL models, the DST orthogonal method is employed, as seen in Equation 4.11. The maximum of belief rule is used in this study for decision-making, as presented in Equation 4.13.

$$m(S) = (m_1 \oplus m_2 \oplus ... \oplus m_l)(S) = \frac{\sum_{B_1 \cap ... \cap B_l} m_1(B_1) m_2(B_2)..m_l(B_l)}{1 - K} \tag{4.11}$$

where

$$K = \sum_{B_1 \cap ... \cap B_l =} m_1(B_1) m_2(B_2)...m_l(B_l), \tag{4.12}$$

K reflects the degree of disagreement among the l distinct DL algorithms.
In the present research, the decision-making strategy used is to select the greatest belief value as stated in Equation 4.13.

$$x \in C_i, if Bel(C_i)(x) = max[Bel(C_k)(x), 1 \leq k \leq n]. \tag{4.13}$$

4.4 Optimizing Data Analytics through Microservices

4.4.1 Analytical Process in Microservices

This section describes the analytical process involved in microservices, which analyzes and processes data collected from IoT devices.

The data analytics process is developed using a set of microservices that work together to analyze data and generate insights. These microservices are designed to perform specific tasks within the data analytics process, such as data preprocessing, analysis, and visualization.

The first step in this process is data enhancement, which involves improving the quality of the collected data by identifying missing or incorrect values and converting it into a consistent format. To achieve this, data preprocessing microservices are employed to examine the data and identify any missing or inconsistent information. The data is then normalized, transformed, or augmented to ensure better accuracy. Once the data has been processed, it is used as input for the model learning microservices, which apply ML/DL techniques to analyze the data and develop intelligent IoT applications.

The microservices involved in this process include preprocessing, model manager, model learning, interpretation, and results fusion. Once the model configuration is defined in the cloud microservices, the preprocessed data is then utilized for the learning process. This involves the communication among edge and cloud nodes, where microservices are responsible for selecting the appropriate learning models to be trained. The functionalities of these microservices are described in the following subsections.

4.4.1.1 Data Preprocessing

Data preprocessing serves as the primary stage in data analysis, where raw data is transformed into a format suitable for further analysis. We design a set of services for this step using the service-oriented computing paradigm. For this stage, the following microservices are designed:

- Data cleaning: This component defines the data filtering process, where the noise is eliminated, the missing values are resolved, and unnecessary data is removed. It ensures that the data is of high quality and ready for further analysis.
- Data integration: This component integrates and structures data from different sources with the same format. Since the data is gathered from various sources, this step ensures that the data is consistent and can be used for further analysis.
- Data scaling/transformation: This component normalizes and transforms the gathered data to a defined range to enhance the consistency of the data. This ensures that the data is consistent and can be used for further analysis.

4.4.1.2 Models Building

After the data preprocessing, the next stage is model building. This stage uses various analytical methods, such as ML, DL, and statistical analysis, to build models that can accurately predict future outcomes based on historical data. This stage consists of a set of microservices, each of which is responsible for a predefined function, as follows:

- Model creator: The model creator microservice is responsible for selecting the appropriate ML models based on the specific IoT use case and creating the initial models. It also handles the compilation of the models before training. The selection of the right models is crucial to ensure that the models are well-suited to the IoT use case and can achieve high accuracy with low computation costs. The model creator microservice can take into consideration a range of factors, including the type of data, the expected output, and the available computational resources to identify the suitable model for the task.

- Model training: Using the preprocessed data, the model begins the training process according to the configuration of the global model. The model training microservice is responsible for selecting the appropriate ML algorithm and tuning the hyperparameters to optimize the performance. In the centralized learning method, this microservice is deployed in the cloud, while in the distributed learning method, it is deployed on the edge nodes.

- Model evaluator: The model evaluator microservice is responsible for examining the effectiveness of the global model. This microservice evaluates the model's accuracy, precision, recall, and F1-score and provides feedback on the model's performance to the model manager. According to the evaluation findings, the model manager may initiate a new round of training or adjust the model's hyperparameters to improve its performance.

- Model uploader: In the FL setting, the model uploader microservice transfers the learned weights from edge nodes to the central server. This microservice ensures that the trained model parameters are transferred securely and efficiently to the central server.

- Model aggregator: The model aggregator microservice in the cloud is a key component in the FL setting. It receives the updated model parameters from the edge devices and aggregates them to create a new global model.

4.4.1.3 Interpretation

In our approach, the final stage of data analytics is to use the learned models to come up with predictions using new data. The predictions or decisions are then fused using a fusion method to obtain a final output. This output can be used to inform decision-making or provide insights into the data. It should be emphasized that the quality of the data preprocessing, analysis, and model training stages have a significant impact on the accuracy and reliability of the final output. Therefore, these stages should be carefully designed and executed to ensure the best possible results. This stage consists of the following microservices:

- Interpretation microservice: This microservice is responsible for making predictions or classifications based on the trained model, given new input data. This microservice takes in the input data and passes it through the model to generate prediction/ classification, which can then be used for decision-making or other downstream processes.
- The results fusion microservice: This microservice combines the results obtained from multiple learned models. It employs a fusion technique to merge the data analysis findings from models to enhance the accuracy and comprehensiveness of the results. This microservice can use a variety of fusion methods, such as majority voting, weighted averaging, and fuzzy logic, according to the use case to aggregate the results from different models. The fusion of results can help to overcome the limitations of individual models and provide more reliable and accurate insights for decision-making.

In Figure 4.3, the procedure flow between the FL server and edge client is illustrated in detail through a set of microservices to describe the analytical process following the decentralized learning method. Figure 4.4 describes the microservices connections among cloud and edge nodes to perform the analytical process through the centralized learning method.

Algorithm 1 outlines the entire process for the fusion-based FL approach. On the other hand, Algorithm 2 explains the analytical process of the proposed approach associated with the centralized learning method.

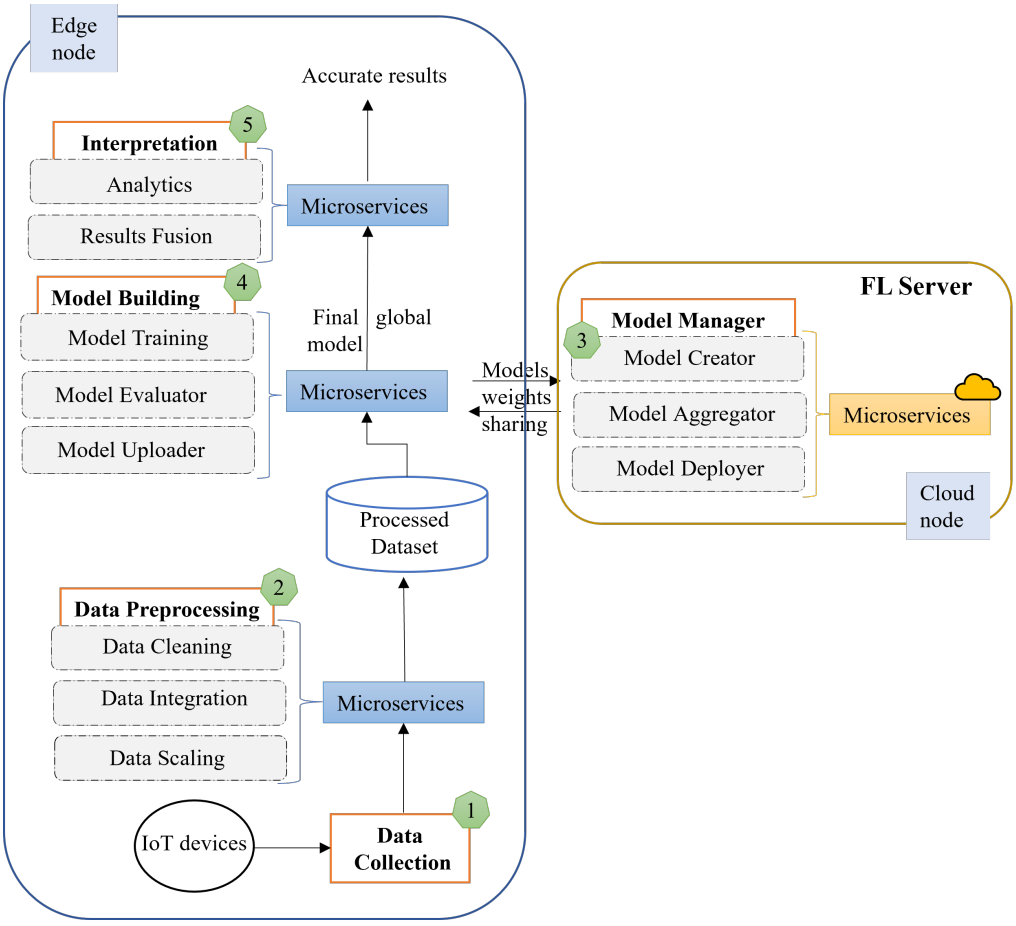


FIGURE 4.3: The generalized workflow between the FL server and the client following the FL method. The communication between Step 3 and 4 are done for predefined rounds.

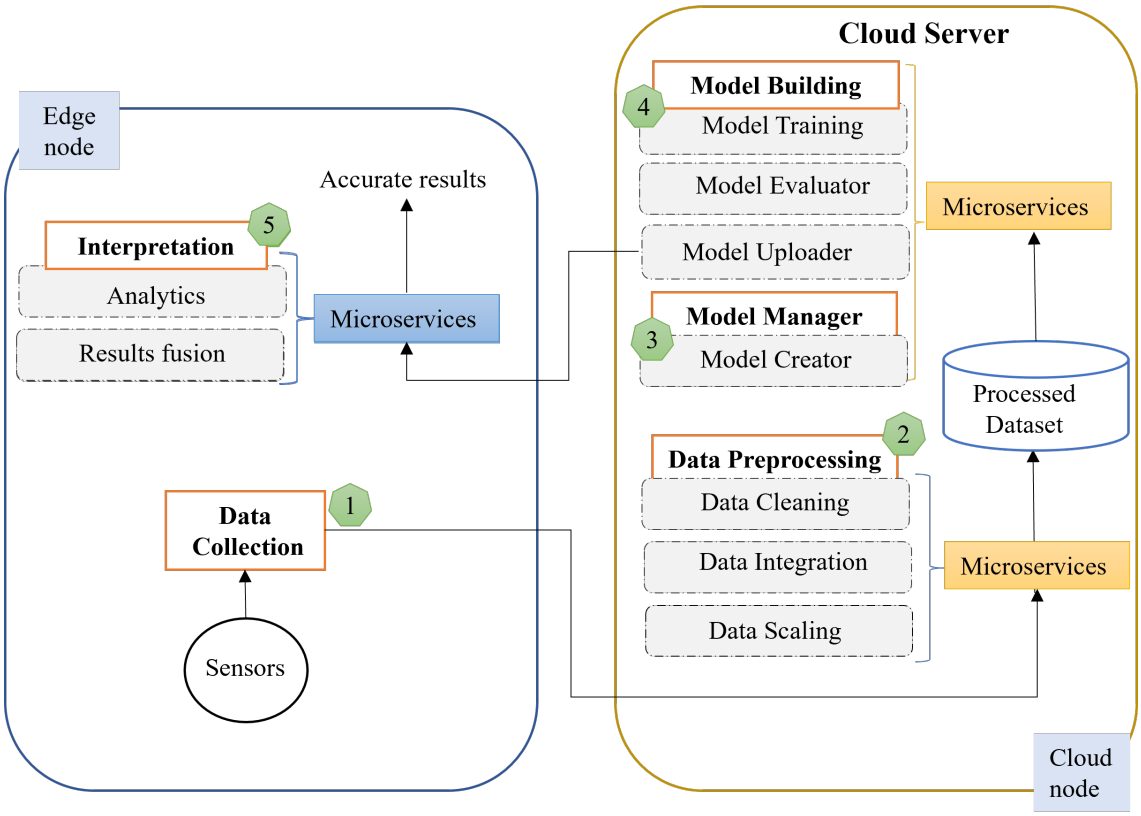


FIGURE 4.4: The generalized workflow between the cloud and edge following the CL method.

Algorithm 1 : Analytical process following the distributed learning method

Input: N different datasets DS_1, DS_2, .., DS_n.
Output: Prediction/detection .
Begin
Choose the appropriate models. (three models used in this example)
Set the models' initial parameters for all the clients
mw_1, mw_2, mw_3 /* The local models' weights */
C= c1, c2, . . . , cn /* Initializing the FL clients */
Reading input data
function F(L) training (CommunicationRrounds):
    while c_i in C do: **do**
        /* train the models with local data of each client */
        mw_1 = train(data in c_i)
        mw_2 = train(data in c_i)
        mw_3 = train(data in c_i)
        return mw_1, mw_2, mw_3
    end while
end function
function F(L)aggregation:
    /* Averaging parameters of each model from all devices using Eq(4.3) and(4.4)*/
    GMw_1 = FLaverage($c_1(mw_1)$, $c_2(mw_1)$, .., $c_n(mw_1)$)
    GMw_2 = FLaverage($c_1(mw_2)$, $c_2(mw_2)$, .., $c_n(mw_2)$)
    GMw_3 = FLaverage($c_1(mw_3)$, $c_2(mw_3)$, .., $c_n(mw_3)$)
    Replace local models with global models
    return GMw_1, GMw_2, GMw_3
end function
function E(n)sembler (GMw_1, GMw_2, GMw_3) :
    for `each client` **do**
        newdata /*to be used for data analytics*/
        predictions[] = Mw_1(newdata), Mw_2(newdata), Mw_3(newdata)
        final prediction = ensemble(predictions)
    end for
end function

Algorithm 2 : Analytical process following the centralized learning method

/*Step 1: Data collection*/
Collect the dataset
Upload the dataset to the cloud server.
/*Step 2: Data preprocessing*/
Clean data, remove redundant data, and solve the missing data problem.
Data scaling and integration.
/*Step 3: Model creation */
Initialize the model parameters.
/*Step 4: Model learning */
Loop through the training data:
a. Forward propagate the data through the model
b. Calculate the loss
c. Backward propagate the loss through the model
d. Update the model parameters using an optimization algorithm
/*Step 5: Model evaluation */
Loop through the testing data:
a. Forward propagate the data through the trained model
b. Calculate the loss and accuracy of the model with testing data.
/*Step 6: Model uploader
Save the trained model for later use
/*Step 7: Interpretation
Load the saved models
Process new data using the models for predictions
Combine the predictions using a fusion technique

4.4.2 Overview of the Proposed Approach

Figure 4.5 provides an overview of the operations involved in the proposed approach. The process starts with the collection of data by different IoT devices. This data is then gathered and stored for further processing.

The next step is the data processing phase, where the collected data undergoes various preprocessing and transformation steps to make it suitable for analysis. This processed data is then transmitted to the TL models generated by the model manager. The TL models utilize the FL methodology to train and update the parameters using the distributed data available on the edge. FL ensures that the training process preserves data privacy by keeping sensitive information on edge devices and only exchanging model updates.

Once the TL models are trained, they are being used to assess new data and provide valuable insights. The model's performance and accuracy are continuously monitored and evaluated to ensure its effectiveness in analyzing IoT data.

In the final phase, the results obtained from the TL model are combined using a suitable fusion method. This fusion method, chosen based on the specific use case, integrates the individual model outputs to provide more effective and reliable results.

Overall, the proposed approach encompasses data processing, TL model training using FL, result evaluation, and fusion of the obtained outputs. This comprehensive process aims to leverage the benefits of microservices, TL, FL, and fusion methods to enhance the performance of IoT data analytics and provide valuable insights for decision-making.

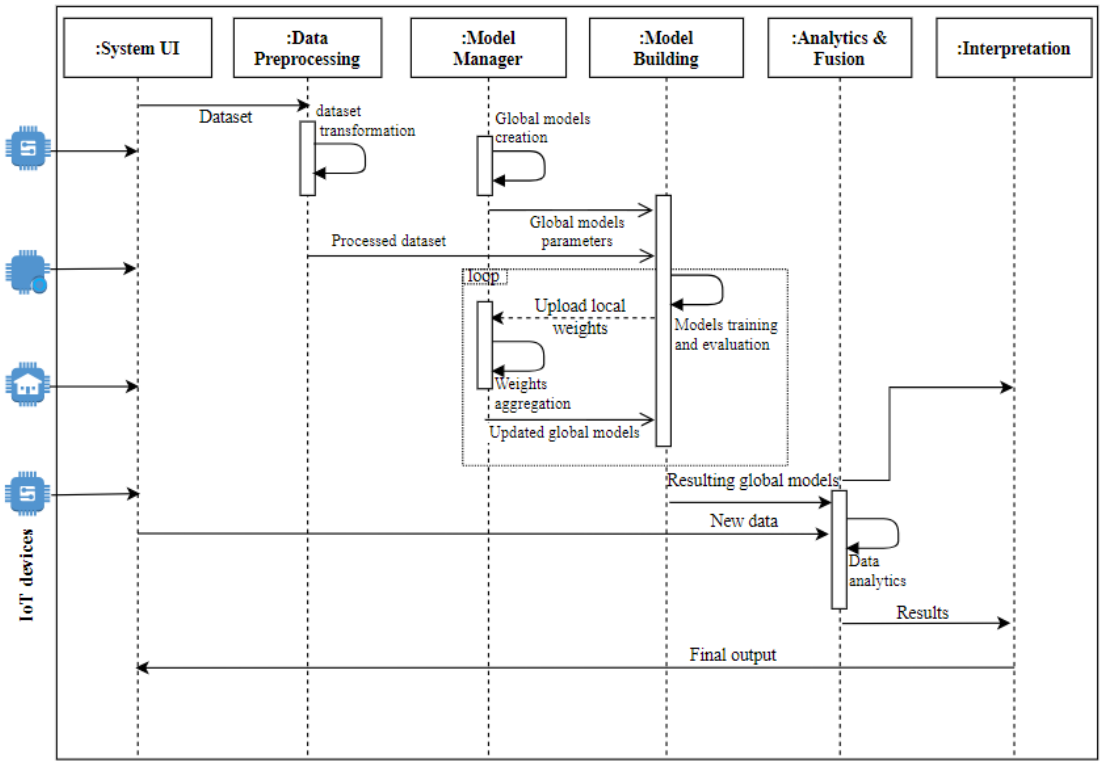


FIGURE 4.5: Overview of the proposed approach: processing, and analyzing IoT data using microservices, FL,TL, and Fusion Methods

4.4.3 Context-Aware Microservice Selection

In the proposed approach, the utilization of a variation of microservices for different steps, including data preprocessing, models building, and data fusion, enables a modular and flexible architecture for IoT data analytics. This modular approach allows for the customization and adaptation of the analytical pipeline based on the specific characteristics and requirements of the IoT system at hand.

Each type of input data, such as sensor data, images, or textual information, may have distinct processing needs. Similarly, different analytical models require specific algorithms and techniques to be applied effectively. The specific IoT context and requirements heavily influence the selection of microservices. Factors such as the nature of the data, the desired level of accuracy, and the available computational resources, play a role in determining the suitable microservices for each step of the analytics pipeline.

In addition, the fusion method employed in our approach is carefully selected based on the specific use case being considered. It primarily depends on the performance achieved during the model learning phase. In cases where the individual models exhibit high performance and satisfactory results, a simple ensemble method can be employed for fusion. This involves combining the predictions of multiple models and generating an aggregated output. However, in scenarios where the individual models' performance is not as impressive and there is a need for significant improvement, we adopt a more advanced fusion method. One such method is the Dempster-Shafer theory of evidence, which allows for the combination of uncertain and conflicting information from different sources. This method incorporates a more profound level of fusion, taking into account the reliability and credibility of each model's predictions to make a more informed decision.

By aligning the chosen microservices with the requirements of the IoT use case, the proposed approach can effectively address the challenges of data preprocessing, model learning, and data fusion in a way that meets the specific needs and objectives. Therefore, the approach can leverage the power of microservices to handle diverse data types, perform complex analysis tasks, and ultimately drive valuable insights for informed decision-making in the IoT ecosystem.

4.5 Conclusion

In summary, the chapter presents a novel approach to data analytics that addresses the challenges of privacy and scalability in IoT applications. The suggested solution employs the microservices architecture, that enables the creation of fine-grained and reusable entities that can be easily updated and combined. The process involves several stages, including data preprocessing, data analysis, model building and training, and ultimately, using learned models and fusing decisions to generate insights. The various microservices developed for each stage are responsible for carrying out specific functions, such as data cleaning, data integration, and model evaluation. The final stage emphasizes the importance of using the learned models and fusing decisions to acquire more precise and comprehensive outputs.

In the upcoming chapter, we will delve into the practical application and evaluation of the proposed approach. By showcasing how the microservices architecture can be implemented in real-world IoT scenarios, we intend to exhibit the efficiency and scalability of the proposed solution. Furthermore, the performance of our approach will be evaluated using appropriate metrics to measure the quality of the data analytics results.

Chapter 5

Experimentation

Contents

5.1 Introduction

In the previous chapter, we introduced our proposed approach to IoT data analytics based on microservices architecture. Our approach is centered around creating a collection of fine-grained, reusable entities that are loosely linked together to form a flexible and adaptable IoT application. We have designed this architecture to support both centralized and distributed learning, in accordance with the use case's particular demands. By leveraging FL techniques for model learning, our approach delivers intelligent microservices that can effectively analyze data while maintaining high levels of privacy.

In this chapter, we will dive deeper into the capabilities of our approach and evaluate its performance in a variety of different use cases and scenarios. Through the presentation and assessment of various IoT scenarios, we aim to show how versatile and adaptable our suggested strategy is.

5.2 Environment Setup

For the implementation of the experiments, we utilized a PC having these specifications: an Intel(R) Core(TM) i7-8565U CPU @ 1.80 GHz 1.99 GHz processor, 16 GB RAM, and an NVIDIA GeForce MX graphics card running on the latest version of Windows 11. We deployed the developed microservices using Docker images [117], which were orchestrated into a cluster of containers using swarm orchestration [118] for efficient management and execution.

To implement and test the various ML/DL models, we utilized the power of Python 3.8 [119] along with the versatile Jupyter notebook [120] provided by the Anaconda package [121]. This combination allowed us to leverage the benefits of Python's vast ecosystem of libraries and tools, while also providing an interactive and intuitive environment for model development and testing. Through the use of these technologies, we were able to achieve efficient and reliable experimentation, leading to robust and accurate results.

In Figure 5.1, we illustrate our simulation experiments intended for achieving the primary objective of our proposed approach - ensuring predictive analytics in IoT environments through the adoption of microservices and AI techniques. To design and validate the different scenarios using our approach, we utilized open-source components that were readily accessible to the public. Through the adoption of a microservices-based architectural style and FL/TL approaches, we were able to ensure the scalability and reliability of our suggested solution.

To facilitate the implementation of our proposed approach, we constructed three virtual machines using distributed Docker images [117]. Each virtual machine hosted a microservice, with the Docker images installed on each machine forming a cluster of containers. Swarm orchestration [122] was employed to manage and orchestrate the different components of this cluster, providing a high degree of availability for our applications. The first virtual machine was dedicated to supporting the model manager processes, which made use of microservices such as the model creator, aggregator, and deployer. The second virtual machine was set up for model-learning processes, which comprised microservices for training, evaluating, and uploading models. Finally, the third virtual machine was responsible for executing the data analytics algorithms, which were implemented using microservices for data preprocessing and results fusion.

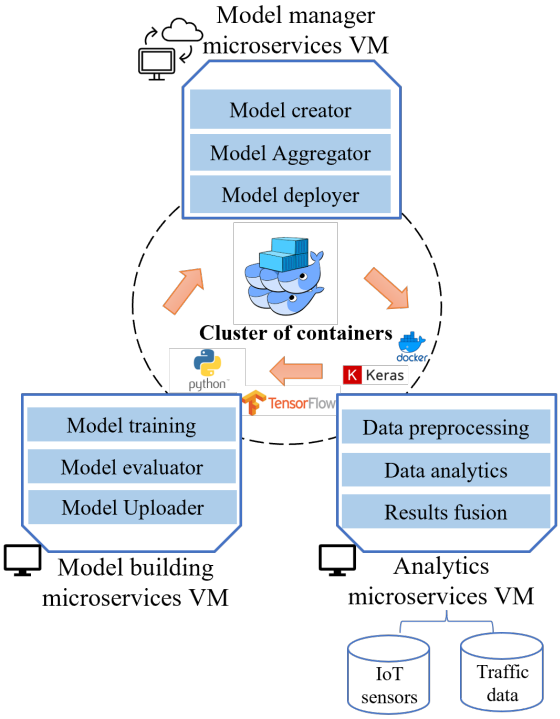


FIGURE 5.1: Experimental setup of the proposed approach used for different IoT use case scenarios.

This setup allowed us to achieve efficient and reliable execution of our proposed approach, with each virtual machine handling specific tasks in a distributed and scalable manner. By leveraging the power of Docker swarm and microservices, we were able to create a highly available and adaptable system capable of handling complex and diverse data sets.

5.3 Case Study 1: Cybersecurity

The proliferation of IoT devices has led to increased vulnerability to cyber-attacks. Weak passwords, unsecured network connections, and a shortfall in secure update methods are just some of the well-known vulnerabilities that make IoT devices attractive targets for malware. To combat these threats, data analytics models have emerged as powerful tools for cybersecurity. Data analytics models are able to

mine, investigate, and get knowledge from big volumes of data, making them well-suited for analyzing traffic flows and detecting attacks. By leveraging these models, cybersecurity professionals can ease the burden of manual analysis and more quickly identify and respond to threats. In this case study, we employ our proposed approach to enhance cybersecurity. Specifically, we will use data analytics models to analyze traffic flows, detect attacks, and classify them using different scenarios.

5.3.1 Experiment 1: Detection and Classification of IoT Malware

We implement an IoT-malware detection and classification use case to validate and assess our proposed approach through an FL setting. The different stages of the use case scenario are depicted in Figure 5.2. It comprises IoT sensors and devices, malware detection entities (clients), and FL servers as its high-level components. The data traffic is collected, preprocessed, and analyzed for each client using our proposed malware detection solution. The solution is composed of microservices that support data preprocessing, model building, and interpretation. These microservices encapsulate distinct capabilities, such as data transformation, model training, and model uploader. With the help of an FL server, the flow of data analytics is carried out to detect and classify various malware effectively.

5.3.1.1 Dataset

To evaluate and validate our proposed approach, we utilized the MaleVis dataset [123], which is a comprehensive dataset for evaluating IoT malware using image-based analysis. The dataset consists of 14,226 RGB-converted images from 26 distinct IoT malware families, covering 25 types of malware divided into six categories: adware, trojans, viruses, worms, backdoors, and one benign class. To ensure the robustness of our approach, we partitioned the dataset into three subgroups for training, validation, and testing. This allowed us to train and fine-tune our model on a large portion of the dataset, while still having enough data for validation and testing to accurately assess the effectiveness of our approach.

Below, we demonstrate the outcomes of implementing our suggested method to identify and categorize IoT malware, utilizing the MaleVis dataset.

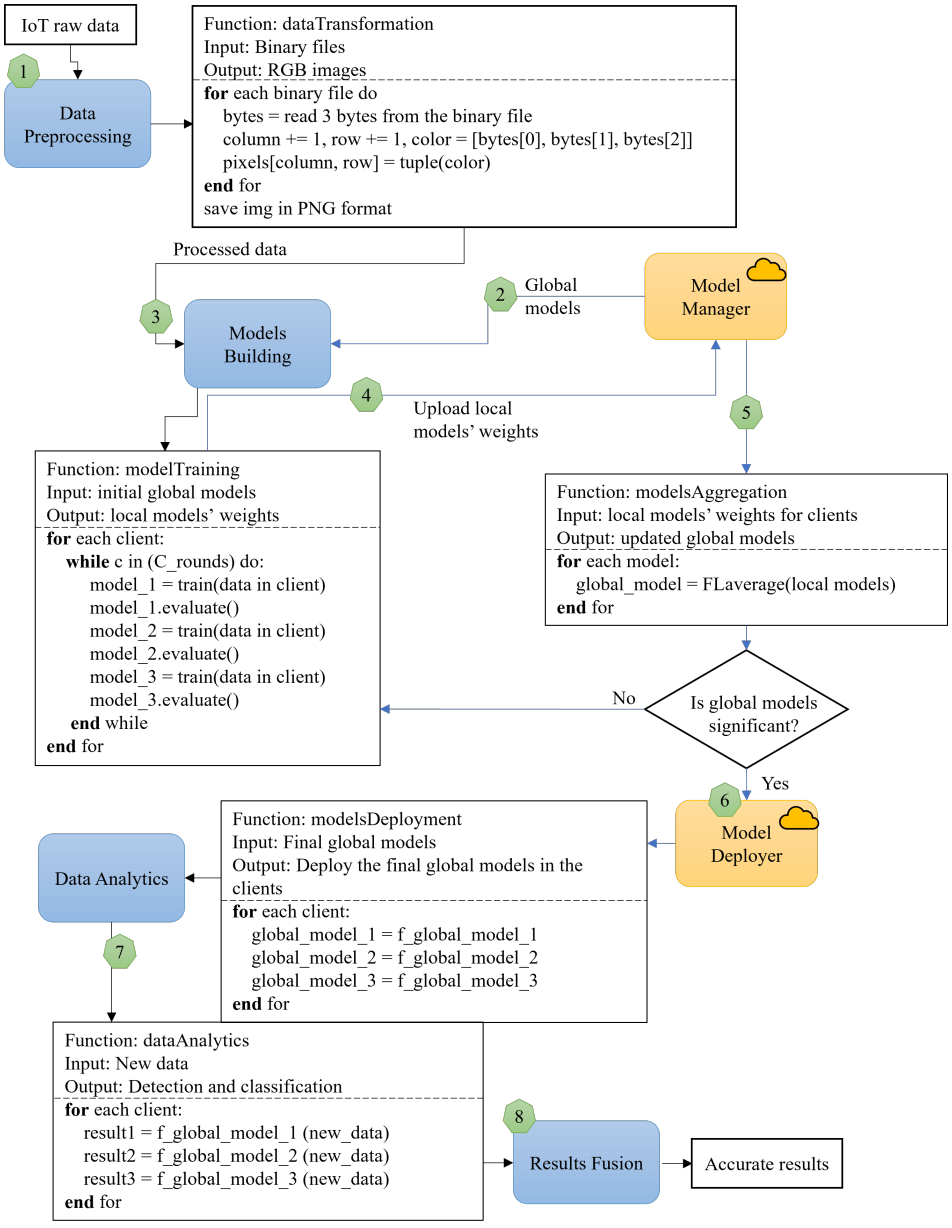


FIGURE 5.2: IoT malware detection and classification Scenario.

5.3.1.2 Data Preprocessing

During this stage, we utilized the bin2png algorithm to transform the PE files acquired from malfunctioning software into RGB images. This step was essential to preprocess the data and make it compatible with the pre-trained CNN models, which require images as inputs. In order to convert binary files into images, we used a three-byte encoding scheme, where each pixel in the output image represents three bytes. Specifically, the initial byte is encoded and displayed in the red channel of the resulting image, while the second byte is shown in the green channel and the third byte in the blue channel. Using this encoding scheme, we generated RGB images for all the collected PE files. These images were then utilized as input for the models to perform further analysis and detection of malware in the collected data. This approach allowed us to leverage the power of pre-trained CNN models, which have been successfully demonstrated to be efficient in detecting and classifying malware.

5.3.1.3 Models Building

To evaluate our proposed approach, we conducted experiments in a federated setting with 10 virtual IoT devices. We selected three pre-trained CNN architectures, including MobileNetV2, DensNet201, and InceptionV3, and used TL to detect and efficiently classify IoT malware. To train the CNN models, we selected hyperparameters based on prior research and experimentation. Table 5.1 gives details about the hyperparameters chosen for the training of the considered CNNs, including learning rate, batch size, and epochs number. We conducted several experiments with different combinations of hyperparameters to ensure the best performance of the models.

Table 5.1: Hyperparameters' values of the considered CNNs.

Hyperparameter	**Value**
Batch size	32
Number of clients	10
Number of epochs	10
Number of communication rounds	35
Optimizer	Adam
Learning rate	1e-3

5.3.1.4 Interpretation

Table 5.2 summarizes the findings of fine-tuned CNNs trained using FL. The results show that the FL paradigm yields excellent model performance. Figure 5.3 illustrates that the MobileNetV2 model achieved the best performance by applying the FL, with accuracy, precision, recall, and F1-score metrics surpassing those obtained with the CL approach.

After completing this experiment, the evidence theory was used for decision fusion, and the results of the generated classifiers are presented in Table 5.2. In terms of efficiency, the combined classifier beat individual models, earning 99.24% accuracy, 99.12% precision, 99.36% recall, and 99.45% F1-score.

5.3.1.5 Experiments Evaluation

1. **Performance Against Centralized Learning:**

   To conduct a comparative analysis, we also implemented our proposed approach's Centralized Learning model using the TensorFlow DL framework [124]. The MobileNetV2, DenseNet201, and InceptionV3 were fine-tuned and trained using MaleVis data. In contrast to the FL implementation, the knowledge produced by the centralized approach is shared in the cloud, and the amount of data utilized for training is enormous compared to the amount of data used in the FL approach. The CNNs were trained over 35 epochs. Comparing the CNN models' performance results emphasizes that the FL implementation outperforms the CL implementation.

   Table 5.2 includes the experimental results of the fine-tuned CNNs trained using the CL and FL in addition to the fusion-based classifiers of these models. The results presented by Table 5.2 indicate that the performance of models trained in FL is better than those trained in CL approach. It is clear from Figure 5.3 that the accuracy, precision, recall, and F1-score metrics of the MobileNetV2 model, which provided the highest performance results by applying the FL, surpass the performance results of the same model by using the CL approach.

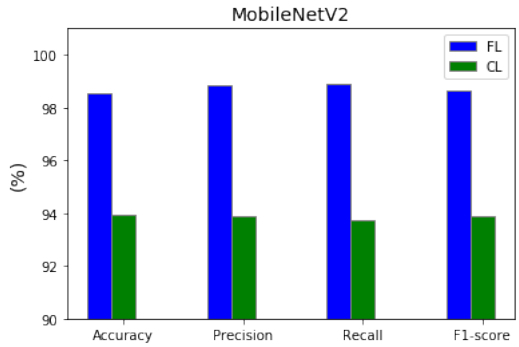


FIGURE 5.3: Performance results of the MobileNet CNN resulting from the adoption of FL and CL approaches.

TABLE 5.2: Experimental results of the single CNN models, the centralized Dempster-based fusion classifier, and the proposed approach.

Model \ Evaluation metric	**MobileNetV2**		**DenseNet201**		**InceptionV3**		**CL Dempster-based fusion classifier**	**Proposed approach**
	FL	***CL***	***FL***	***CL***	***FL***	***CL***		
Accuracy(%)	98.56	93.95	98.27	95.12	97.98	93.49	97.15	99.24
Precision(%)	98.57	93.98	98.35	95.14	97.87	93.65	96.87	99.12
Recall(%)	98.68	93.84	98.12	95.18	97.78	93.23	97.02	99.36
F1-score(%)	98.52	93.81	98.24	95.21	98.94	93.21	97.48	99.45
Loss	0.1072	0.2939	0.1251	0.2483	0.1538	0.2462	0.1541	0.08471
Specificity(%)	98.78	93.25	98.15	95.09	97.74	93.15	97.44	99.31
MCC(%)	98.48	93.89	98.18	95.23	97.57	93.41	97.27	99.18

2. **Performance Analysis of Microservices:**

The suggested framework has been evaluated in terms of microservices performance. For this purpose, we looked at the execution time of microservices based on their functionality. The end-to-end response time of the data analytics function (f) refers to the entire procedure from start to finish. It is defined by Equation 5.6:

$$T_f = T_{Pre} + T_{Pro} + T_{Fus} \tag{5.1}$$

The time spent in data pre-processing, data analytics, and results fusion stages are denoted as $T_{Pre}, T_{Pro}, T_{Fus}$, respectively.

The total time cost (T_{total}) of the data analytics function corresponds to the response time of its composed microservices and is calculated using Equations 5.2 and 5.3 [97]:

$$T_{total} = \sum_{i=1}^{n} RT_{\mu}, \tag{5.2}$$

Where

$$RT_{\mu} = Trans(M_j, M_i) + T_{Exec} \tag{5.3}$$

T_{Exec} presents the time needed for a microservice execution. $Trans(m_j, m_i)$ represents the duration of data transfer from microservice M_j to microservice M_i and it is computed using Equation 5.4:

$$Trans(M_j, M_i) = \begin{cases} \frac{size}{bw(m_j, m_n)} + \frac{size}{bw(m_n, m_i)} & if r_j \in edge, r_i \in cloud \\ 0 & r_i = r_j \\ \frac{size}{bw(m_j, m_i)} & otherwise \end{cases} \tag{5.4}$$

Where $size$ is the data size transferred in Mbits, and $bw(r_j, r_n)$ is the bandwidth of the link between the data creator and data consumer in Mbits.

Figure 5.4 depicts the execution time for each microservice in the malware detection/classification process, starting with data preparation and finishing with the fusion of the results. The data preprocessing microservice converts the recorded traffic from Portable Execution (PE) file to an RGB image. This process takes only approximately three seconds to complete. Because the data is processed through three different CNN models, the data analytics function is the most time-consuming. These models analyze the transformed image, and the classification results, expressed as probabilities for each class, are the provided outputs.

Because it includes the selection of the best output based on the collected results, the fusion microservice operates significantly faster than the preceding microservices. In a monolithic architecture, all the components of an application are tightly coupled and must be deployed together, which can make it difficult to scale individual components to handle the increased load. This can result in longer response times and decreased performance. In contrast, a microservices-based architecture allows for the development of smaller, more specialized services that can be scaled independently. This allows for faster response times and improved performance, as the application can better adapt to changing loads and resource demands.

3. **Impact of Number of FL Clients:**
Let us now replicate the previous learning process with a larger number of data owners. Instead of decomposing the train set into 10 clients, we will breakdown it down into 15, 20, 25, and 30 clients. Figure 5.5 depicts the average

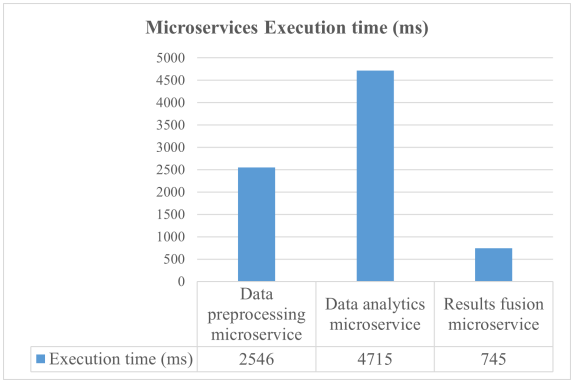


FIGURE 5.4: The execution time of different microservices in the process of malware detection.

calculation time of the learning process for the number of clients considered. As a result, varying the number of clients in the FL process doesn't significantly worsen the overall execution time. Here are the main reasons for this conclusion:

- Asynchronous Communication: clients communicated with the server independently, without waiting for other clients. This can help reduce the overall execution time, as clients do not have to wait for slow or unresponsive peers. As a result, adding more clients may not necessarily increase the overall execution time since each client can operate independently;
- Parallel Processing: the local computations for multiple clients were performed in parallel, which helped reduce the overall time required to complete a training round;
- Sample Size: each client trained the models on a subset of its local data. Here the sample size is relatively small; adding more clients may not significantly impact the overall execution time since each client's contribution to the final model update was relatively small;
- Efficient Aggregation: The server aggregated the model weights obtained by clients to update the global model. Here an efficient aggregation strategy, which is federated averaging, was used, so the overhead of aggregating updates from a large number of clients was minimized.

4. **Comparison with Existing Research:**
We compare the acquired findings with those of previous studies published that worked on the same use case scenario and used the same dataset in

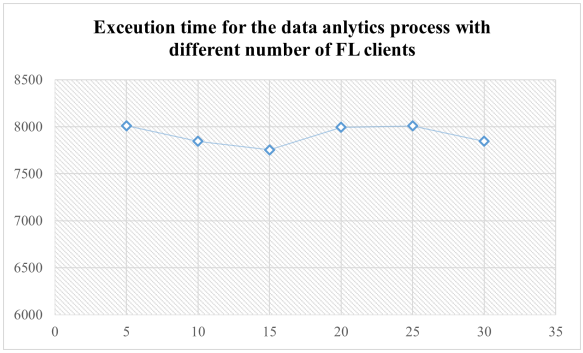


FIGURE 5.5: Execution time in ms for data analytics process with a different number of FL clients.

order to verify the validation of the proposed approach.As seen from the accomplishments and outcomes in Table 5.7, our approach achieves the highest performance results compared to the related studies that rely on centralized ML/DL models and use monolithic architectures.

TABLE 5.3: Evaluation of our approach against previous methods utilizing MaleVis dataset

Ref	**Method**	**Precision(%)**	**Recall(%)**	**F1-score(%)**	**Accuracy (%)**
[125]	TL (DenseNet201)	not stated	not stated	not stated	97.48
[126]	Deep RF approach	97.43	97.32	97.42	97.43
[127]	Hybrid CNN approach using AlexNet and ResNet152	97.1	94.9	94.5	96.6
[128]	TL with ShuffleNet and DenseNet201)	99.80	95.61	95.37	95.01
[129]	Pretrained DenseNet model	98.56	97.74	98.15	98.21
[130]	RF-based voting classifier	98.74	98.67	98.70	98.98
Proposed approach	FL and microservices based framework for IoT data analytics	99.12	99.36	99.45	99.24

5.3.2 Experiment 2: Detection and Classification of Denial of Service Attacks in IoT Networks

According to Mittal et al., [131], Wireless Sensor Networks (WSNs) are extremely susceptible to Denial-of-Service (DoS) threats, especially in hostile environments where sensor nodes can be physically accessed and controlled. DoS attacks, as described by Bhatt et al. [132], can severely impact internet bandwidth and cause traditional connections to terminate by overwhelming the network with large amounts of unnecessary data. WSN jamming threats have been categorized into different categories of DoS threats, such as Blackhole, Grayhole, Flooding, and Scheduling [133].

WSNs and IoT systems are susceptible in almost every layer, which renders them an attractive target for a variety of DoS assaults. Data-driven methods based on DL/ML-based approaches have been used to develop efficient defensive tactics and remedies for security breaches, notably DoS assaults.

Based on our proposed approach, we propose a solution for the detection and classification of DoS attacks [134]. The following points highlight the primary findings and contributions of our experiment:

- Put forth a deep TL technique to detect and categorize DoS assaults in WSNs based on microservices;
- Transform tabular data into images using a visual method;
- Train different pre-trained CNN architectures to detect and classify DoS attacks;
- Employ the WSN-DS dataset for learning and experiments.
- Use the ensemble learning method to provide accurate decisions;
- Compare findings with related works.

5.3.2.1 Dataset

This study uses the WSN-DS dataset [135] to evaluate the proposed solution. WSN-DS was gathered using the Low Energy Aware Cluster Hierarchy (LEACH) protocol

[136], which is a well-known hierarchical routing protocol in WSNs. This dataset is very huge and includes 19 attributes and four DoS attack categories, namely Blackhole, Grayhole, Flooding, Scheduling, and also normal (benign) samples.

5.3.2.2 Data Preprocessing

1. **Data Integration:**

   The labels for different sorts of assaults must be transformed to numerical values to ensure that they do not impact the efficiency of the CNNs utilized in the trials. The WSN-DS dataset comprises Blackhole, Flooding, Grayhole, Normal, and Scheduling assaults, which are denoted in the results of the investigations as 0, 1, 2, 3, and 4.

2. **Data Normalization:**

   The id and time features are removed from the dataset. Using Equation 5.5, each of the values of the other sixteen attributes is normalized around 0.0 and 1.0.

$$A' = \frac{(A - A_{min})}{(A_{max} - A_{min})} \tag{5.5}$$

   where A represents the unprocessed attribute value, while A_{max} and A_{min} refer to the maximum and minimum feature values, respectively.

3. **Data Transformation:**

   Following dataset preparation, The Image Generator for Tabular Data (IGTD) technique is applied to transform the WSN-DS data towards image representations [137]. Figure 5.6 shows several examples of the created images.

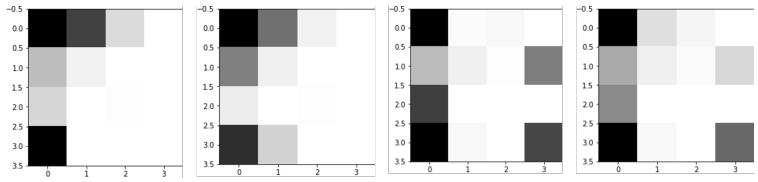


FIGURE 5.6: Visual representations of the dataset using IGTD technique.

5.3.2.3 Models Building

The objective of this step is to tackle the issue of detecting and categorizing new DoS attacks using existing knowledge. Instead of creating and teaching DL models from the beginning, which is time-consuming and requires a significant amount of computational resources and data, the use of already learned CNNs can enhance and expedite the learning process by reusing their layers' weights. To detect and classify different DoS threats, ResNet18, DenseNet161, VGG16, SqueezeNet, and EfficientNetB3 were employed. The five CNN architectures were learned over 25 epochs with Adam optimizer, cross-entropy loss function, and SoftMax activation function. The hyper-parameters employed for the configuration of model training are presented in Table 5.4.

TABLE 5.4: Specified hyper-parameters for optimal models performance

Hyper-parameters	**Values**
Batch size	64
Epochs	25
Images size	150, 150
Optimizer	Adam
Loss function	Cross-entropy

5.3.2.4 Interpretation

The CNN models were learned using the preprocessed data, and their effectiveness was evaluated using the evaluation metrics and training time measures. Test images were employed to assess the overall performance of the models, and Table 5.5 shows that the DenseNet161 CNN achieved high performance and reached an accuracy of 99.995%. The VGG16 was the model with the lowest performance. Overall, the learned models performed well and accurately, with no notable variances in their outcomes.ResNet18, SqueezeNet, and EfficientNetB3 all functioned similarly, having an accuracy of around 99.9%.

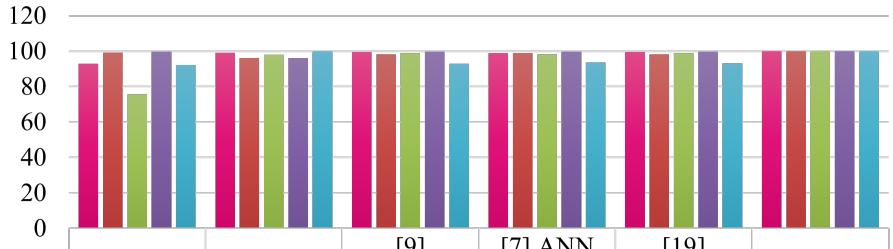
Due to the impressive performance of achieving 99.9%, we conducted a thorough examination of overfitting to ensure the reliability of the results. We assessed the loss obtained from the learned models, which served as an indicator of the model's ability to generalize beyond the training data.

The following stage was to use ensemble learning by integrating the findings of CNNs utilizing the hard voting approach. The performance provided by the ensemble classifier was evaluated using the test set of data, and Table 5.5 compares the outcomes of the TL-based DL models. The proposed ensemble classifier produced higher classification results and outperformed individual CNNs with 100% precision, recall, F1-score, and accuracy. The TL-based ensemble classifier performed admirably in identifying and classifying DoS assaults.

These experimental findings emphasize the importance to use TL as well as ensemble learning in detecting and classifying DoS assaults correctly and robustly.

TABLE 5.5: Evaluation results of TL-based classifiers on the WSN-DS dataset.

Model	Class	Precision	Recall	F1-score	Accuracy	Loss
ResNet18	Blackhole	99.90	99.95	99.92	99.983	0.0021
	Flooding	98.95	100	100		
	Grayhole	100	99.96	99.98		
	Normal	100	99.95	99.97		
	Scheduling	99.77	99.84	99.81		
DenseNet161	Blackhole	100	100	100	99.995	0.0035
	Flooding	100	99.84	99.92		
	Grayhole	100	100	100		
	Normal	99.99	100	99.99		
	Scheduling	100	100	100		
VGG16	Blackhole	99.90	99.95	99.92	99.951	0.0016
	Flooding	98.95	100	100		
	Grayhole	100	99.96	99.98		
	Normal	100	99.95	99.97		
	Scheduling	99.77	99.84	99.81		
SqueezeNet	Blackhole	99.95	100	99.97	99.959	0.0013
	Flooding	99.39	99.54	100		
	Grayhole	99.93	100	99.65		
	Normal	99.98	99.97	99.98		
	Scheduling	100	99.77	99.88		
EfficientNetB3	Blackhole	100	100	100	99.991	0.0021
	Flooding	100	99.69	99.84		
	Grayhole	100	100	100		
	Normal	99.98	100	99.99		
	Scheduling	100	100	100		
Proposed Approach	Blackhole	100	100	100	100	0
	Flooding	100	100	100		
	Grayhole	100	100	100		
	Normal	100	100	100		
	Scheduling	100	100	100		



	[8] ANN	[24] RF	[9] GXGBoost model	[7] ANN based entity embedding	[19] LightGBM algorithm	Proposed approach
Recall, Blackhole	92.8	99	99.5	98.9	99.5	100
Recall, Flooding	99.4	96	98.2	98.9	98.1	100
Recall, Grayhole	75.6	98	98.9	98.1	98.9	100
Recall, Normal	99.8	96	99.9	99.9	99.9	100
Recall, Scheduling	92.2	100	92.9	93.6	93.07	100

FIGURE 5.7: Comparison of the proposed approach's performance with existing studies in the literature.

5.3.2.5 Comparison with Similar Studies

The suggested solution for detecting DoS attacks in WSNs using DL, TL, and ensemble learning techniques was compared to previously published studies using the same data. The results demonstrated that this solution obtained a high recall percentage compared to the other reviewed studies, which are based on traditional neural networks. By transforming tabular data into images and using pre-trained CNN architectures, the approach efficiently dealt with the dataset's unbalanced distribution among classes. The TL and ensemble learning paradigms had a considerable influence on the effectiveness of the implemented models in coping with the WSN-DS dataset's extreme unbalance and achieving excellent recall, F1-score, and accuracy values across each DoS class.

5.4 Case Study 2: Healthcare

The fast rise of IoT and big data analytics has transformed the medical sector, opening the door for the creation of intelligent healthcare systems. By utilizing cutting-edge technology like IoT and big data analytics, healthcare providers can

now offer personalized, real-time care to patients, leading to improved health outcomes. In this view, we use the proposed approach to help in a smart and effective disease prediction.

5.4.1 Diagnosis of Pneumonia Using Chest X-Ray Imagery

The smart healthcare sector has greatly benefited from the rapid advancements in IoT and big data analytics, leading to improved patient care and well-being. Various IoT applications have been developed using data analytics techniques for various medical services such as illness forecasts, nutrition evaluation, and elderly care.

For pneumonia detection, we utilized TL with a fine-tuning method on 5 previously learned CNN architectures (VGG16, InceptionV3, RestNet50, Xception, and DenseNet201) trained through the FL. Our solution also involves combining the decisions extracted from these models using the Dempster-Shafer Theory (DST) to achieve high-performance image classification results. The last categorization choice is reached by combining the outcomes of the trained models used. Figure 5.8 illustrates the different stages of the pneumonia detection use case.

5.4.1.1 Dataset

The study conducted experiments on the Pneumonia Chest X-Ray dataset, which was gathered by the Guangzhou Women and Children's Medical Center in China and is accessible on Kaggle [138]. The data set includes 5855 images for training, validation, and testing groups, containing 1591 normal images and 4273 infected ones.
Although the dataset itself may not be directly generated from IoT devices, its application in the context of healthcare and medical monitoring aligns with the broader theme of IoT.

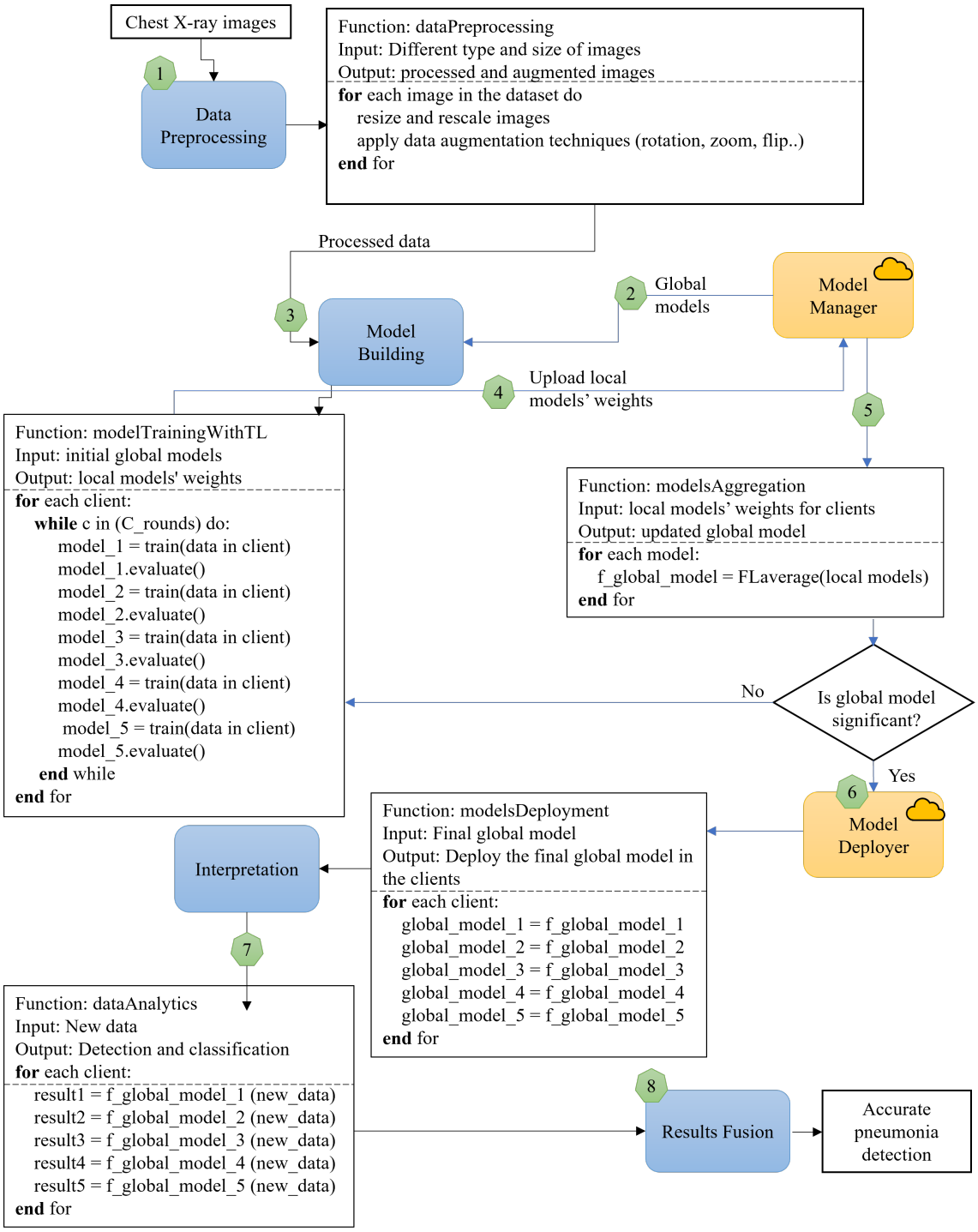


FIGURE 5.8: Pneumonia detection scenario following the proposed approach.

5.4.1.2 Data Preprocessing

To enhance the performance of classifiers, we utilized seven different techniques to augment the dataset. These included resizing the images to 224x224 and employing methods like rescaling, horizontal flipping, random rotation, width and height shift, and adjusting zoom and brightness levels.

5.4.1.3 Models Building

The proposed approach was implemented using the TensorFlow Federated framework on 5 virtual IoT devices in a federated setting. Five pre-trained CNN architectures were utilized with TL for efficient pneumonia detection, including VGG16, Xception, InceptionV3, ResNet50, and DenseNet201. Models' aggregation was performed over 15 communication rounds, with each CNN trained for 20 epochs using local data. The Adam optimizer with a learning rate of 1e-4 and the cross-entropy loss function were used for model configuration. Input images were resized to 224x224 pixels and normalized, and a batch size of 32 was used during training. To facilitate the self-learning process, the dataset was divided into training and test data sets across 5 different clients.

5.4.1.4 Interpretation Stage

After training the five DL classifiers, their performance was evaluated using the testing set of chest X-ray images. The classification results of each classifier are shown in Table 5.7. The DenseNet201 model achieved the best performance. The performance difference between the models is not significant, indicating relatively balanced performance.
Following the training phase, Dempster's theory was applied to combine the models' outputs. The performance outcomes of the generated classifiers are presented in Table 5.6. The combined classifier outperforms the individual models in terms of accuracy with 98.1%.

TABLE 5.6: Performance results of the learned DL models and the proposed classifier.

Model// Metric	VGG16	Xception	InceptionV3	ResNet50	DenseNet201	Proposed classifier
Accuracy	94.8	95.6	96.2	95.1	96.4	98.1
Precision	94.2	95.4	96.4	94.6	96.8	97.9
Recall	95.0	96.2	96.9	95.3	96.5	98.3
F1-score	94.6	95.8	96.6	94.9	96.6	98.1

5.4.1.5 Experimental Evaluation

Microservices Performance Assessment

Moreover, the performance of the proposed approach was assessed by evaluating the execution time of microservices based on their functionality. The end-to-end response time of the pneumonia detection function f is defined as the sum of the time spent in data pre-processing T_{Pre}, Interpretation T_{Inter}, and results fusion T_{Fus} stages, as shown in Equation 5.6.

$$T_f = T_{Pre} + T_{Inter} + T_{Fus} \tag{5.6}$$

Figure 5.9 showcases the execution time of each microservice in the pneumonia detection process, encompassing data preparation, interpretation, and results fusion. The data preprocessing microservice took 198 ms, while the interpretation microservice took 564 ms. On the other hand, the results' fusion microservice enhanced performance with an execution time of only 198 ms. As a result, the total time for the pneumonia detection process was calculated to be approximately 960 ms.

Comparison

To validate the proposed approach, we compared our previously published study, which focused on the same use case scenario and utilized the same dataset but adopted a centralized learning setting with monolithic architectures [65]. The performance results, as depicted in Table 5.7, clearly demonstrate that our approach outperforms the previous study in terms of achieving the highest performance results. This validates the effectiveness of our approach that utilizes decentralized DL models in an FL architecture.

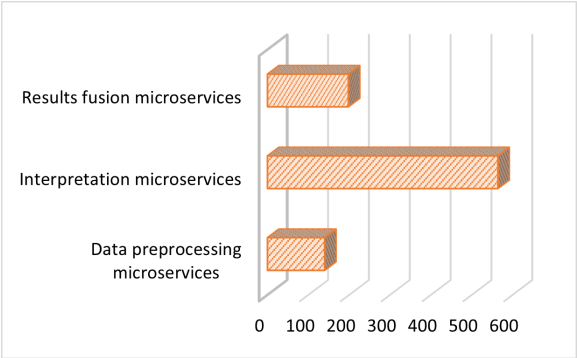


FIGURE 5.9: Microservices execution time for the pneumonia detection process.

TABLE 5.7: Comparison of the proposed approach results with our previously published study

Model	Type of learning	Architecture	Accuracy	Precision	Recall	F1-score
[65]	Centralized	Monolithic	97.5	97.5	98	97.8
Proposed solution	FL	Microservices	98.1	97.9	98.3	98.1

Discussion

To ensure accurate pneumonia diagnosis, it is essential to identify all relevant features present in the chest X-ray images. To enhance model generalization, ensemble learning was employed to combine the predictions of multiple TL models to make more accurate and robust predictions.

Privacy concerns arise when dealing with sensitive medical data. For this, FL was used to address these concerns by enabling the training of models on distributed data while keeping the data securely on their local devices. This allowed for collaborative model training across multiple institutions without sharing the raw data, preserving privacy and confidentiality.

In addition to privacy considerations, achieving high scalability and response time is crucial in healthcare. Microservices architecture was leveraged to address these requirements by breaking the data analytics function into smaller, specialized microservices that can be deployed and scaled independently. This allowed for efficient resource utilization and improved scalability. Moreover, microservices architecture enabled faster response times as each microservice can be optimized for performance and response time, leading to quicker results for end-users.

A robust and efficient pneumonia diagnosis solution was achieved by combining ensemble learning for model generalization, FL for privacy preservation, and microservices architecture for scalability and response time optimization. This approach ensures accurate predictions, protects patient privacy, and enables efficient and scalable data analytics in the healthcare environment.

5.5 Case Study 3: Environment

It is essential for smart cities to offer their residents a healthy living environment. This can be achieved by utilizing modern and innovative technologies to enhance the quality of air and water, monitor pollution levels, protect natural habitats, and identify and manage waste effectively [24]. By implementing these measures, a city can create a favorable environment for its inhabitants.

5.5.1 Smart Monitoring of Water Environments

Recently, there is a growing need for sophisticated water management solutions to effectively regulate the quantity and quality of drinking water [139]. Monitoring water is crucial in our world as it permits immediate time monitoring of measurements of water quality as well as efficient resource administration in a smart city to ensure sufficient water access to inhabitants [140]. To address this need, an intelligent solution based on IoT has been suggested to facilitate the monitoring of water zones.

We proposed a SmartWater solution following our presented approach with the centralized method and using cutting-edge technologies, such as sensor clouds, ML, and knowledge reasoning [141]. The proposed solution is presented in Figure 5.10 in detail. The water network is semantically and multi-relationally represented using knowledge graphs, and incremental network embedding is employed to build detailed representations of water objects, with a focus on damaged water zones [142]. A decision process is implemented to develop a smart management plan based on the existing condition of the water zones.

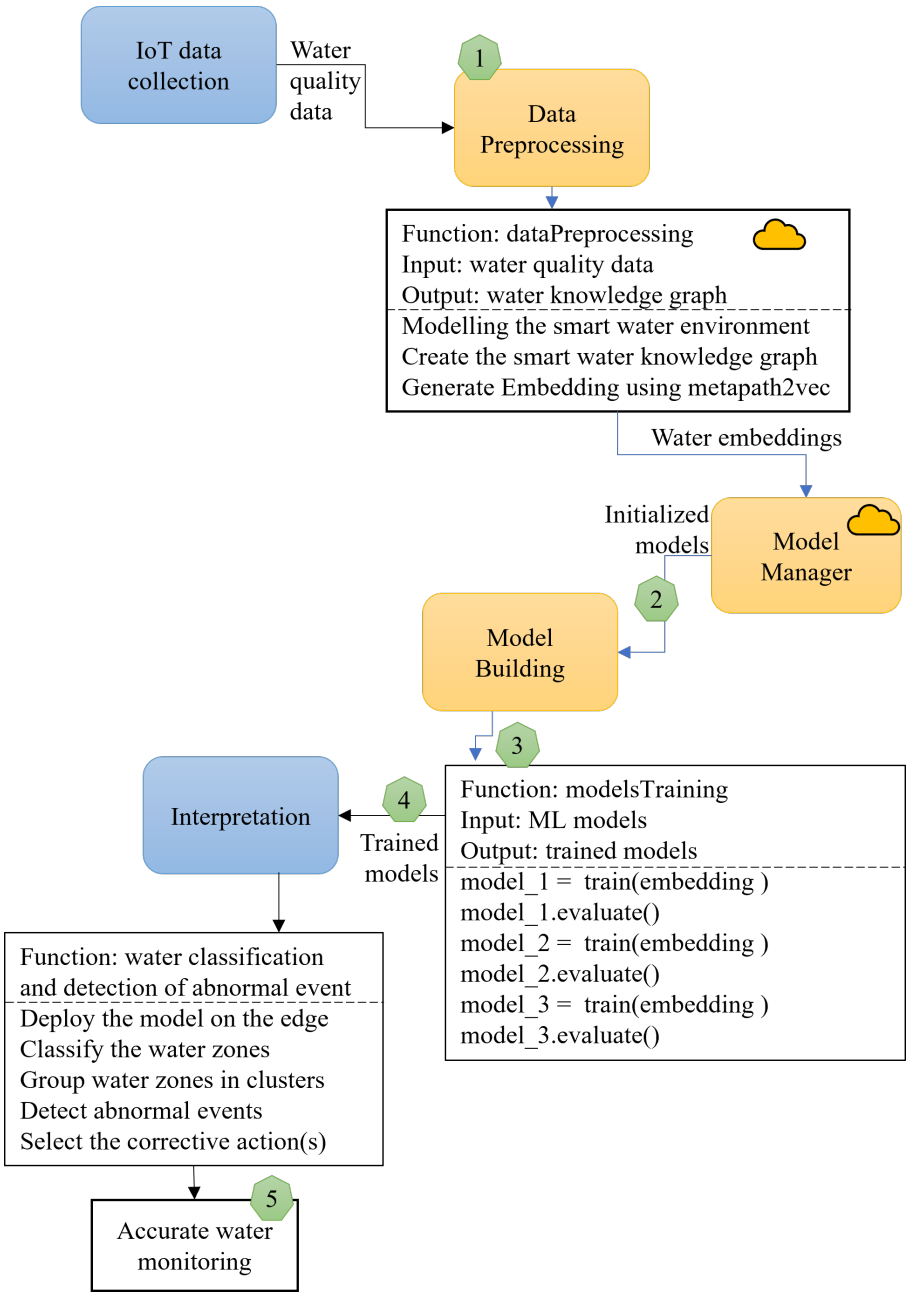


FIGURE 5.10: The proposed smart monitoring of water environment scenario following the proposed approach (Centralized method).

5.5.1.1 Dataset

Experiments were conducted utilizing a dataset including water quality data from several sites to examine the efficiency of the proposed SmartWater management solution. Temperature, pH, turbidity, dissolved oxygen, conductivity, biological oxygen demand, nitrate, fecal coliform, and total coliforms are among the 9 metrics included in this collection of data. Because the dataset lacked information about trigger events and their associated situations, therefore this was provided by generating the Water Quality Index (WQI) and classifying water instances depending on their WQI scores.

5.5.1.2 Data Preprocessing

The information network for the proposed approach was built using the Water Knowledge Graph (WKG), which consists of three types of nodes: water entity, event, and action. To represent water entities, the id of locations was used, and changes in the WQI were utilized to represent the event entities. The management rules were constructed at random to reflect the actions. The structure of the water network is reflected in the relations between the entities, while water zones and event nodes are labeled based on WQI values. The metapath2vec algorithm is utilized to continuously update the WKG, with a $P : W - E - C - E - W$ guided metapath used to generate random walks. The proposed approach utilizes metapath2vec as an incremental embedding technique to accurately represent both semantic and structural connections between distinct zone locations. It employs a two-step incremental embedding process, incorporating guided random walks and Skip-Gram learning to extract node sequences and capture structural relationships. The resulting embeddings facilitate tasks such as clustering, classification, anomaly detection, and decision-making in the water network. Algorithm 3 provides a summary of the complete process for embedding and classification.

Algorithm 3 : Water zones' embedding

Input: $\mathcal{G}$ - Water knowledge graph, $\mathcal{L}$ - Set of labels.
Output: $\mathcal{V}_l^p$ - Node embeddings in the water network.
Begin
$\{\mathcal{E}_t^p\}_{t=1}^T \leftarrow \emptyset$
for each node $v_i \in V$ **do**
    X = MetaPathRandomWalk(G, P, v_i, l)
    X = HeterogeneousSkipGram(X, k, MP) ;
    for each node type $t \in T$ **do**
        Learn the representations of node $\boldsymbol{v}_i$
        $\mathcal{L} \leftarrow$ Minimize relation's inference loss for v_i
        $\mathcal{V}_l^p \leftarrow \mathcal{V}_l^p \cup v_i$
    end for
end for
Return $\{\mathcal{V}_l^p\}_{l=1}^{|\mathcal{L}|}$

5.5.1.3 Models Building

The aim of this stage is to compare the classification performance of water zones. To achieve this, three different ML models were utilized: SVM, LR, and KNN. The knowledge graph was built using the dataset and node representations were learned from it. These representations were then fed into the classifiers.

5.5.1.4 Interpretation

Results of the classification of the used models based on the nine parameters of data are illustrated in Table 5.8, along with the confusion matrices 5.11. The SVM model demonstrates better performance than the other models. Furthermore, incorporating incremental network embedding led to an enhancement in the classification, demonstrating the effectiveness of latent representations learned through embedding in an accurate water zone classification. The metapath2vec algorithm was deemed successful in generating suitable embeddings.

TABLE 5.8: Performance evaluation of water zones classification using embedding techniques

Model	Accuracy	Precision	Specificity	F1-score
SVM	100	100	100	100
LR	99.75	99.87	99	99.76
KNN	99.95	99.97	99	99.94

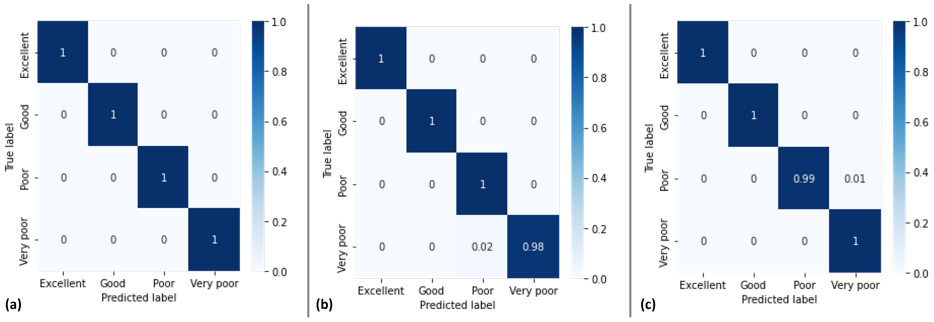


FIGURE 5.11: Normalized confusion matrices for the water zones classification without embedding using: (a) SVM, (b) LR, (c) KNN

5.6 Limitations of The Proposed Approach

While the proposed approach showcases several advantages, it is crucial to acknowledge its limitations for a comprehensive evaluation. One notable limitation is the dependence on data quality and availability. The accuracy and reliability of the analytics models heavily rely on the quality of the IoT data, and incomplete, noisy, or unreliable data can adversely affect the performance of the approach. Furthermore, limited data availability or data that inadequately represents the target application domain can hinder the approach's effectiveness.

Another significant limitation arises from the computational resource requirements of the approach. The utilization of AI methods, such as ML and DL, can be computationally intensive, posing challenges when deploying the approach on resource-constrained IoT devices or edge computing nodes. Limited processing power, memory, or energy constraints may impede scalability and real-time responsiveness.

Lastly, the approach may necessitate domain expertise and customization, especially in developing accurate and effective ML models for specific IoT applications.

The need for expert knowledge and limited labeled data in certain domains can pose challenges, potentially affecting the applicability and performance of the approach.

5.7 Conclusion

This chapter showcases various applications of the proposed approach using different learning settings. Firstly, we demonstrate the use of FL for IoT malware detection and multi-classification. Secondly, we discuss the proposed solutions for healthcare and environmental case studies.
To measure the efficacy of our proposed approach, we evaluate its efficiency among its three key stages: data preprocessing, model building, and interpretation, and the results indicate that our approach performs well.

Chapter 6

Conclusion and Future Lines of Research

Contents

6.1 Summary and Conclusions

The rapid growth and increasing complexity of the IoT have made it necessary to use multi-source and heterogeneous IoT data to facilitate efficient data analytics for a broad variety of important applications. Because the results of data analytics are often time-sensitive, it is crucial to generate them with minimal latency and high reliability. Therefore, it would be beneficial to reuse efficient architectures that have been validated through a large number of challenging test cases. By implementing this approach, we can ensure the reduction of these concerns while still maintaining effective, scalable, and secure data privacy. Additionally, it enables the provision of precise and timely insights to support crucial decision-making processes.

In this thesis, we proposed a solution using a microservices-based architecture that allows an IoT application to be structured as a collection of fine-grained, loosely coupled, and reusable entities. The proposed solution uses the promising capabilities of centralized and FL to provide intelligent microservices that ensure efficient, flexible, and extensible data analytics. This solution aims to deliver cloud calculations to the edge to reduce runtime and bandwidth bottlenecks in addition to maintaining data privacy. A series of IoT case studies were used to validate the suggested approach. In terms of performance, the findings demonstrated that our suggested strategy beat existing current techniques while also ensuring data confidentiality and lowering transmission costs.

Using an architecture built on microservices to incorporate learning processes has resulted in various benefits, including scalability, flexibility, reliability, modularity, and integration. These advantages have simplified the development, implementation, and management of intricate ML and DL systems in comparison to conventional monolithic architectures.

The following contributions were made in the context of this thesis work:

- **The proposition of a microservices-based approach that supports the data analytics in IoT application:** We propose a microservices architecture that allows for the creation of reusable and fine-grained entities, which can be easily combined and updated. The process includes a number of stages, starting with data preprocessing, followed by model development and training. The final step involves utilizing the learned models and then fusing decisions to produce insightful results. Each stage is managed by specialized microservices responsible for carrying out specific functions, such as data cleaning, integration, and model evaluation. In summary, our approach illustrates the advantages of utilizing microservices to improve the efficiency and efficacy of data analytics in IoT applications.
- **Enhance privacy using the FL paradigm:** We utilize FL to ensure data privacy by allowing multiple clients to collaboratively train a model to avoid revealing raw data. Instead, the client's data stays on their devices, and just model updates are exchanged between clients and the cloud server. This approach enhances privacy in several ways. First, it reduces the risk of data breaches or leaks, as the raw data does not go outside of the client machines. Second, it limits the amount of sensitive data that is exposed to the cloud central server, further reducing the danger of unauthorized access. Finally,

FL can also help address regulatory compliance requirements related to data privacy, as it provides a mechanism for organizations to train ML models on sensitive data while preserving privacy.

- **Improved model generalization using TL:** To enhance our model's generalization, we utilized TL, which allows the model to discover patterns from a diverse set of data. This technique is especially useful when working with smaller datasets, where training a model from scratch can be challenging. TL could also minimize the amount of time and computational resources required to train a model, as the pre-trained model already contains knowledge about the data. By incorporating TL, we were able to improve the performance of our solutions on smaller datasets while reducing training time and resources.

- **Improved performance using ensemble learning:** To achieve more accurate and comprehensive outcomes, we combined the learned models and decisions. Ensemble learning was employed to enhance our approach by reducing the impact of errors in individual models and leveraging the strengths of multiple models. It also increased the robustness of the analytical model, making it more resistant to overfitting as well as noise in data. By incorporating different types of models, ensemble learning created a diverse set of predictions that captured a wide set of patterns in data. Overall, the utilization of ensemble learning was instrumental in improving our approach by enhancing prediction accuracy, increasing model robustness, and capturing a more diverse range of patterns in the data.

- **Application and evaluation of the proposed approach:** The proposed approach was enhanced by leveraging the cloud and edge nodes, which support both centralized and distributed learning, making it suitable for a wide range of use cases. To demonstrate the feasibility and validity of the approach, a set of experiments were implemented and conducted on real use-case scenarios. These experiments were designed to test the proposed approach in a practical setting and to validate its effectiveness. Furthermore, a comparative study was conducted, which compared the suggested solutions with related state-of-the-art methods. The comparative study provided valuable insights into the strengths and weaknesses of the proposed approach and highlighted its advantages over other approaches. Overall, these experiments and comparative studies played an important role in providing evidence of the proposed approach's effectiveness and practicality.

In summary, the proposed approach exhibits a significant degree of intelligence. It incorporates cognitive abilities such as perception, action control, and deliberative reasoning through the use of AI methods like ML and DL. By training analytics models using these techniques, the approach demonstrates its ability to understand and process complex IoT data. The approach also follows behavioral principles based on rationality and social norms. It leverages centralized and distributed learning techniques, microservices, and edge computing nodes to ensure efficient data analytics. By adhering to these principles, the approach aims to achieve rational and socially acceptable outcomes.
Furthermore, the approach has the capacity to adapt through learning. It utilizes TL and ensemble learning techniques to improve model generalization and prediction accuracy. This adaptability enables the approach to continually enhance its performance and adapt to changing data patterns and scenarios.

In general, while the proposed approach may not encompass the entirety of human-level intelligence, it demonstrates a considerable degree of intelligence by incorporating cognitive abilities, following behavioral principles, and exhibiting adaptability through learning.

6.2 Future Work and Research Directions

Many areas of research can be anticipated from our work. These areas are both methodological and practical:

As part of our future work, we plan to extend our research by integrating other types of DL models, such as RNN and DRL. By incorporating RNNs, we aim to leverage their sequential modeling capabilities, making them suitable for IoT tasks involving time-series data or sequences. This integration will enable us to capture temporal dependencies and effectively process sequential information. Additionally, the inclusion of DRL in our research holds promise for enhancing our understanding of the interaction between DL and reinforcement learning. By combining the power of DNNs with reinforcement learning algorithms, we can develop intelligent agents capable of learning and making decisions in complex and dynamic environments.

By integrating these diverse DL models, it is essential to establish a selection

strategy that ensures an efficient composition of microservices within the architecture. The selection strategy plays a crucial role in determining the appropriate microservices to include in the composition, considering factors such as functionality, performance, and scalability.

Our models are highly efficient, but it's essential to use them for decision-making to learn about their decision-making process. However, the lack of transparency in these models can lead to mistrust, highlighting the importance of explanations that enhance the overall dependability and comprehensibility of ML systems. Therefore, we aim to concentrate our research on the emerging area of Explainable Deep Learning (XDL), which involves developing innovative approaches and tools to learn a deeper understanding of data, variables, and decisions incorporated in DL models and to make them understandable to humans [143], [144]. Our ultimate goal is to build transparent models that promote trust in their decision-making capabilities.

As a potential avenue for future research, we plan to explore the application of zero-shot learning techniques in our models. Zero-shot learning is a subset of ML that allows models to generalize to new, unseen tasks by leveraging prior knowledge or information [145]. Incorporating zero-shot learning into our models could potentially improve their adaptability and make them more robust in handling novel tasks. Therefore, we see this as an exciting area to investigate in the future.

List of Publications

My PhD. and research cooperation resulted in multiple international publications, which are mentioned in the following:

Journal Publications

[P1] Safa Ben Atitallah, Maha Driss, and Henda Ben Ghezela. "FedMicro-IDA: A Federated Learning and Microservices-based Framework for IoT Data Analytics." Internet of Things, Elsevier. (Quartile: Q1, Impact Factor: 5.711) [Accepted on 10/6/2023]

[P2] Maha Driss, Wadii Boulila, Haithem Mezni, Mokhtar Sellami, Safa Benatitallah, Nouf Alharbi. "Enhancing Decision-Making in IoT-based Water Environments through Probabilistic and Evidence Theory-based Knowledge Graph Embedding." Sensors, MDPI, 23(10), pp.4672. (Quartile: Q1, Impact Factor: 3.847).

[P3] Safa Ben Atitallah, Maha Driss, and Iman Almomani. "A Novel Detection and Multi-Classification Approach for IoT-Malware Using Random Forest Voting of Fine-Tuning Convolutional Neural Networks." Sensors, MDPI, 22, p.4302, 2022. (Quartile: Q1, Impact Factor: 3.847)

[P4] Haithem Mezni, Maha Driss, Wadii Boulila, Safa Ben Atitallah, Mokhtar Sellami, and Nouf Alharbi. "Smartwater: A service-oriented and sensor cloud-based framework for smart monitoring of water environments." Remote Sensing, MDPI,

14(4), p. 922, 2022. (Quartile: Q1, Impact Factor: 5.349)

[P5] Maha Driss, Safa Ben Atitallah, Amal Albalawi, and Wadii Boulila. "ReqWSComposer: a novel platform for requirements-driven composition of semantic web services." Journal of Ambient Intelligence and Humanized Computing, Springer, 13(2), p.849-865, 2021. (Quartile: Q1, Impact Factor: 3.662)

[P6] Safa Ben Atitallah, Maha Driss, Wadii Boulila, Anis Koubaa, and Henda Ben Ghézala, "Fusion of Convolutional Neural Networks based on Dempster-Shafer Theory for Automatic Pneumonia Detection from Chest X-Ray Images," International Journal of Imaging Systems and Technology, John Wiley, 32(2), 2021. (Quartile: Q2, Impact Factor: 2.177)

[P7] Safa Ben Atitallah, Maha Driss, Wadii Boulila, Henda Ben Ghézala, "Randomly Initialized Convolutional Neural Network for the Recognition of Covid-19 using X-ray Images." International Journal of Imaging Systems and Technology 32(1), John Wiley, p. 55-73, 2021. (Quartile: Q2, Impact Factor: 2.177)

[P8] Safa Ben Atitallah, Maha Driss, Wadii Boulila, and Henda Ben Ghézala. "Leveraging Deep Learning and IoT Big Data Analytics to Support the Smart Cities Development: Review and Future Directions." Computer Science Review, Elsevier, 38, p.100303, November 2020. (Quartile: Q1, Impact Factor: 8.757)

Conference Publications

[P9] Safa Ben Atitallah, Maha Driss, and Henda Ben Ghézala, " Revolutionizing Disease Diagnosis: A Microservices-Based Architecture for Privacy-Preserving and Efficient IoT Data Analytics Using Federated Learning," in 27th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES2023), 2023. (Rank: B, Publisher: Elsevier) [Accepted on 30/5/2023]

[P10] Safa Ben Atitallah, Maha Driss, Wadii Boulila, and Iman Almomani. "An Effective Detection and Classification Approach for DoS Attacks in Wireless Sensor Networks using Deep Transfer Learning Models and Majority Voting," in 14th International Conference on Computational Collective Intelligence (ICCCI2022), 2022. (Rank: B, Publisher: Springer)

[P11] Safa Ben Atitallah, Maha Driss, and Henda Ben Ghézala, "Microservices for

Predictive Analytics in IoT Applications: Benefits, Challenges, and Future Research Directions," in 26th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES2022), 2022. (Rank: B, Publisher: Elsevier)

[P12] Safa Ben Atitallah, Maha Driss, Wadii Boulila, Anis Koubaa, Nesrine Atitallah, and Henda Ben Ghézala, "An Enhanced Randomly Initialized Convolutional Neural Network for Columnar Cactus Recognition in Unmanned Aerial Vehicle Imagery", in 25th International Conference on Knowledge Based and Intelligent information and Engineering Systems (KES2021), 2021. (Rank: B, Publisher: Elsevier)

Bibliography

[1] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and iot-based applications in smart environments: A systematic review," *Computer Science Review*, vol. 39, p. 100 318, 2021.

[2] C. Santana, L. Andrade, F. C. Delicato, and C. Prazeres, "Increasing the availability of iot applications with reactive microservices," *Service Oriented Computing and Applications*, vol. 15, no. 2, pp. 109–126, 2021.

[3] S. B. Atitallah, M. Driss, W. Boulila, and H. B. Ghézala, "Leveraging deep learning and iot big data analytics to support the smart cities development: Review and future directions," *Computer Science Review*, vol. 38, p. 100 303, 2020.

[4] D. P. Acharjya and K. Ahmed, "A survey on big data analytics: Challenges, open research issues and tools," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 2, pp. 511–518, 2016.

[5] M. K. Saggi and S. Jain, "A survey towards an integration of big data analytics to big insights for value-creation," *Information Processing & Management*, vol. 54, no. 5, pp. 758–790, 2018.

[6] N. Tariq, M. Asim, F. Al-Obeidat, *et al.*, "The security of big data in fog-enabled iot applications including blockchain: A survey," *Sensors*, vol. 19, no. 8, p. 1788, 2019.

[7] T. J. Saleem and M. A. Chishti, "Data analytics in the internet of things: A survey," *Scalable Computing: Practice and Experience*, vol. 20, no. 4, pp. 607–630, 2019.

[8] D. C. Nguyen, M. Ding, P. N. Pathirana, A. Seneviratne, J. Li, and H. V. Poor, "Federated learning for internet of things: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2021.

[9] A. Whitmore, A. Agarwal, and L. Da Xu, "The internet of things—a survey of topics and trends," *Information systems frontiers*, vol. 17, no. 2, pp. 261–274, 2015.

[10] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[11] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial internet of things: Architecture, advances and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2462–2488, 2020.

[12] S. C. Mukhopadhyay, "Wearable sensors for human activity monitoring: A review," *IEEE sensors journal*, vol. 15, no. 3, pp. 1321–1330, 2014.

[13] M. Mohammadi and A. Al-Fuqaha, "Enabling cognitive smart cities using big data and machine learning: Approaches and challenges," *IEEE Communications Magazine*, vol. 56, no. 2, pp. 94–101, 2018.

[14] E. Ahmed, I. Yaqoob, I. A. T. Hashem, *et al.*, "The role of big data analytics in internet of things," *Computer Networks*, vol. 129, pp. 459–471, 2017.

[15] S. Li, L. D. Xu, and S. Zhao, "The internet of things: A survey," *Information systems frontiers*, vol. 17, pp. 243–259, 2015.

[16] I. Chebbi, W. Boulila, and I. R. Farah, "Big data: Concepts, challenges and applications," *Computational collective intelligence*, pp. 638–647, 2015.

[17] D. Singh and C. K. Reddy, "A survey on platforms for big data analytics," *Journal of big data*, vol. 2, no. 1, pp. 1–20, 2015.

[18] K. Rose, S. Eldridge, and L. Chapin, "The internet of things: An overview," *The internet society (ISOC)*, vol. 80, pp. 1–50, 2015.

[19] M. Alaa, A. A. Zaidan, B. B. Zaidan, M. Talal, and M. L. M. Kiah, "A review of smart home applications based on internet of things," *Journal of Network and Computer Applications*, vol. 97, pp. 48–65, 2017.

[20] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *Ieee Access*, vol. 5, pp. 26 521–26 544, 2017.

[21] F. Zantalis, G. Koulouras, S. Karabetsos, and D. Kandris, "A review of machine learning and iot in smart transportation," *Future Internet*, vol. 11, no. 4, p. 94, 2019.

[22] P. Panchatcharam and S. Vivekanandan, "Internet of things (iot) in healthcare–smart health and surveillance, architectures, security analysis and data transfer: A review," *International Journal of Software Innovation (IJSI)*, vol. 7, no. 2, pp. 21–40, 2019.

[23] K. Lakhwani, H. Gianey, N. Agarwal, and S. Gupta, "Development of iot for smart agriculture a review," in *Emerging Trends in Expert Applications and Security: Proceedings of ICETEAS 2018*, Springer, 2019, pp. 425–432.

[24] S. L. Ullo and G. R. Sinha, "Advances in smart environment monitoring systems using iot and sensors," *Sensors*, vol. 20, no. 11, p. 3113, 2020.

[25] S. B. Atitallah, M. Driss, W. Boulila, A. Koubaa, N. Atitallah, and H. B. Ghézala, "An enhanced randomly initialized convolutional neural network for columnar cactus recognition in unmanned aerial vehicle imagery," *Procedia Computer Science*, vol. 192, pp. 573–581, 2021.

[26] M. Marjani, F. Nasaruddin, A. Gani, *et al.*, "Big iot data analytics: Architecture, opportunities, and open research challenges," *ieee access*, vol. 5, pp. 5247–5261, 2017.

[27] C. Surianarayanan, G. Ganapathy, and R. Pethuru, *Essentials of Microservices Architecture: Paradigms, Applications, and Techniques*. Taylor & Francis, 2019.

[28] M. Driss, S. Ben Atitallah, A. Albalawi, and W. Boulila, "Req-wscomposer: A novel platform for requirements-driven composition of semantic web services," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–17, 2022.

[29] S. B. Atitallah, M. Driss, and H. B. Ghzela, "Microservices for data analytics in iot applications: Current solutions, open challenges, and future research directions," *Procedia Computer Science*, vol. 207, pp. 3938–3947, 2022.

[30] E. Wolff, *Microservices: flexible software architecture*. Addison-Wesley Professional, 2016.

[31] M. Driss, "Ws-advising: A reusable and reconfigurable microservices-based platform for effective academic advising," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, no. 1, pp. 283–294, 2022.

[32] D. Hasan and M. Driss, "Sublμme: Secure blockchain as a service and microservices-based framework for iot environments," in *2021 IEEE/ACS 18th International Conference on Computer Systems and Applications (AICCSA)*, IEEE, 2021, pp. 1–9.

[33] T. J. Saleem and M. A. Chishti, "Deep learning for internet of things data analytics," *Procedia computer science*, vol. 163, pp. 381–390, 2019.

[34] T. Saheb and L. Izadi, "Paradigm of iot big data analytics in the healthcare industry: A review of scientific literature and mapping of research trends," *Telematics and informatics*, vol. 41, pp. 70–85, 2019.

[35] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[36] C. Stergiou, K. E. Psannis, B.-G. Kim, and B. Gupta, "Secure integration of iot and cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 964–975, 2018.

[37] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and internet of things: A survey," *Future generation computer systems*, vol. 56, pp. 684–700, 2016.

[38] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[39] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826–1857, 2018.

[40] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[41] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM computer communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[42] W. Yu, F. Liang, X. He, *et al.*, "A survey on the edge computing for the internet of things," *IEEE access*, vol. 6, pp. 6900–6919, 2017.

[43] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[44] J. Qiu, Q. Wu, G. Ding, Y. Xu, and S. Feng, "A survey of machine learning for big data processing," *EURASIP Journal on Advances in Signal Processing*, vol. 2016, no. 1, pp. 1–16, 2016.

[45] R. Mitchell, J. Michalski, and T. Carbonell, "An artificial intelligence approach," *Machine learning. Berlin, Heidelberg: Springer*, 2013.

[46] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[47] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[48] M. Chen, S. Mao, Y. Zhang, V. C. Leung, *et al.*, *Big data: related technologies, challenges and future prospects*. Springer, 2014, vol. 100.

[49] D. Learning, "Deep learning," *High-dimensional fuzzy clustering*, 2020.

[50] H. Li, M. Krček, and G. Perin, "A comparison of weight initializers in deep learning-based side-channel analysis," in *International Conference on Applied Cryptography and Network Security*, Springer, 2020, pp. 126–143.

[51] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[52] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2019.

[53] S. Ben Atitallah, M. Driss, W. Boulila, and H. Ben Ghezala, "Randomly initialized convolutional neural network for the recognition of covid-19 using x-ray images," *International journal of imaging systems and technology*, vol. 32, no. 1, pp. 55–73, 2022.

[54] M. Al-Sarem, W. Boulila, M. Al-Harby, J. Qadir, and A. Alsaeedi, "Deep learning-based rumor detection on microblogging platforms: A systematic review," *IEEE access*, vol. 7, pp. 152 788–152 812, 2019.

[55] J. Gu, Z. Wang, J. Kuen, *et al.*, "Recent advances in convolutional neural networks," *Pattern recognition*, vol. 77, pp. 354–377, 2018.

[56] R. Al Saleh, M. Driss, and I. Almomani, "Cbilstm: A hybrid deep learning model for efficient reputation assessment of cloud services," *IEEE Access*, vol. 10, pp. 35 321–35 335, 2022.

[57] I. Sohn, "Deep belief network based intrusion detection techniques: A survey," *Expert Systems with Applications*, vol. 167, p. 114 170, 2021.

[58] H.-I. Suk, S.-W. Lee, D. Shen, and A. D. N. Initiative, "Latent feature representation with stacked auto-encoder for ad/mci diagnosis," *Brain Structure and Function*, vol. 220, pp. 841–859, 2015.

[59] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[60] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[61] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems*, 2021.

[62] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[63] W. Boulila, M. Driss, E. Alshanqiti, M. Al-Sarem, F. Saeed, and M. Krichen, "Weight initialization techniques for deep learning algorithms in remote sensing: Recent trends and future perspectives," *Advances on Smart and Soft Computing: Proceedings of ICACIn 2021*, pp. 477–484, 2022.

[64] M. Loey, G. Manogaran, M. H. N. Taha, and N. E. M. Khalifa, "A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the covid-19 pandemic," *Measurement*, vol. 167, p. 108 288, 2021.

[65] S. Ben Atitallah, M. Driss, W. Boulila, A. Koubaa, and H. Ben Ghezala, "Fusion of convolutional neural networks based on dempster–shafer theory for automatic pneumonia detection from chest x-ray images," *International Journal of Imaging Systems and Technology*, vol. 32, no. 2, pp. 658–672, 2022.

[66] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," in *International conference on artificial neural networks*, Springer, 2018, pp. 270–279.

[67] G. Vrbančič and V. Podgorelec, "Transfer learning with adaptive fine-tuning," *IEEE Access*, vol. 8, pp. 196 197–196 211, 2020.

[68] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[69] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[70] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[72] A. S. B. Reddy and D. S. Juliet, "Transfer learning with resnet-50 for malaria cell-image classification," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, 2019, pp. 0945–0949.

[73] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[74] B. X. Jie, M. A. Zulkifley, and N. A. Mohamed, "Remote sensing approach to oil palm plantations detection using xception," in *2020 11th IEEE Control and System Graduate Research Colloquium (ICSGRC)*, IEEE, 2020, pp. 38–42.

[75] X. Chao, X. Hu, J. Feng, Z. Zhang, M. Wang, and D. He, "Construction of apple leaf diseases identification networks based on xception fused by se module," *Applied Sciences*, vol. 11, no. 10, p. 4614, 2021.

[76] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information processing & management*, vol. 59, no. 6, p. 103 061, 2022.

[77] M. Driss, I. Almomani, J. Ahmad, *et al.*, "A federated learning framework for cyberattack detection in vehicular sensor networks," *Complex & Intelligent Systems*, pp. 1–15, 2022.

[78] B. Dash, P. Sharma, and A. Ali, "Federated learning for privacy-preserving: A review of pii data analysis in fintech," *International Journal of Software Engineering & Applications (IJSEA)*, vol. 13, no. 4, 2022.

[79] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[80] O. Sagi and L. Rokach, "Ensemble learning: A survey wiley interdisciplinary reviews: Data mining and knowledge discovery," 2018.

[81] H. R. Tizhoosh and L. Pantanowitz, "Artificial intelligence and digital pathology: Challenges and opportunities," *Journal of pathology informatics*, vol. 9, no. 1, p. 38, 2018.

[82] M. Driss, A. Aljehani, W. Boulila, H. Ghandorh, and M. Al-Sarem, "Servicing your requirements: An fca and rca-driven approach for semantic web services composition," *IEEE Access*, vol. 8, pp. 59 326–59 339, 2020.

[83] M. Driss, D. Hasan, W. Boulila, and J. Ahmad, "Microservices in iot security: Current solutions, research challenges, and future directions," *Procedia Computer Science*, vol. 192, pp. 2385–2395, 2021.

[84] S. P. Khedkar, R. A. Canessane, and M. L. Najafi, "Prediction of traffic generated by iot devices using statistical learning time series algorithms," *Wireless Communications and Mobile Computing*, vol. 2021, 2021.

[85] H.-N. Dai, H. Wang, G. Xu, J. Wan, and M. Imran, "Big data analytics for manufacturing internet of things: Opportunities, challenges and enabling technologies," *Enterprise Information Systems*, vol. 14, no. 9-10, pp. 1279–1303, 2020.

[86] G. Ortiz, J. A. Caravaca, A. Garcıa-de-Prado, J. Boubeta-Puig, *et al.*, "Real-time context-aware microservice architecture for predictive analytics and smart decision-making," *IEEE Access*, vol. 7, pp. 183 177–183 194, 2019.

[87] J. Mateo-Fornés, A. Pagès-Bernaus, L. M. Plà-Aragonés, J. P. Castells-Gasia, and D. Babot-Gaspa, "An internet of things platform based on microservices and cloud paradigms for livestock," *Sensors*, vol. 21, no. 17, p. 5949, 2021.

[88] T. Vresk and I. Čavrak, "Architecture of an interoperable iot platform based on microservices," in *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2016, pp. 1196–1201.

[89] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, pp. 381–386, 2020.

[90] F. Jamil, F. Qayyum, S. Alhelaly, F. Javed, and A. Muthanna, "Intelligent microservice based on blockchain for healthcare applications," *CMC-Comput. Mater. Contin*, vol. 69, pp. 2513–2530, 2021.

[91] S. Ali, M. A. Jarwar, and I. Chong, "Design methodology of microservices to support predictive analytics for iot applications," *Sensors*, vol. 18, no. 12, p. 4226, 2018.

[92] J. L. Ribeiro, M. Figueredo, A. Araujo, N. Cacho, and F. Lopes, "A microservice based architecture topology for machine learning deployment," in *2019 IEEE International Smart Cities Conference (ISC2)*, IEEE, 2019, pp. 426–431.

[93] K. Dineva and T. Atanasova, "Architectural ml framework for iot services delivery based on microservices," in *International Conference on Distributed Computer and Communication Networks*, Springer, 2020, pp. 698–711.

[94] M. Taneja, N. Jalodia, J. Byabazaire, A. Davy, and C. Olariu, "Smartherd management: A microservices-based fog computing–assisted iot platform towards data-driven smart dairy farming," *Software: practice and experience*, vol. 49, no. 7, pp. 1055–1078, 2019.

[95] Y.-D. Bromberg and L. Gitzinger, "Droidautoml: A microservice architecture to automate the evaluation of android machine learning detection systems," in *IFIP International Conference on Distributed Applications and Interoperable Systems*, Springer, 2020, pp. 148–165.

[96] S. Y. Nikouei, R. Xu, Y. Chen, A. Aved, and E. Blasch, "Decentralized smart surveillance through microservices platform," in *Sensors and Systems for Space Applications XII*, International Society for Optics and Photonics, vol. 11017, 2019, 110170K.

[97] Z. Houmani, D. Balouek-Thomert, E. Caron, and M. Parashar, "Enabling microservices management for deep learning applications across the edge-cloud continuum," in *2021 IEEE 33rd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, IEEE, 2021, pp. 137–146.

[98] C. Roy, R. Saha, S. Misra, and K. Dev, "Micro-safe: Microservices-and deep learning-based safety-as-a-service architecture for 6g-enabled intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[99] M. Badar, M. Haris, and A. Fatima, "Application of deep learning for retinal image analysis: A review," *Computer Science Review*, vol. 35, p. 100 203, 2020.

[100] B. Bostami, M. Ahmed, and S. Choudhury, "False data injection attacks in internet of things," in *Performability in internet of things*, Springer, 2019, pp. 47–58.

[101] M. Abdel-Basset, H. Hawash, and K. Sallam, "Federated threat-hunting approach for microservice-based industrial cyber-physical system," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1905–1917, 2021.

[102] A. Lakhan, M. A. Mohammed, S. Kadry, S. A. AlQahtani, M. S. Maashi, and K. H. Abdulkareem, "Federated learning-aware multi-objective modeling and blockchain-enable system for iiot applications," *Computers and Electrical Engineering*, vol. 100, p. 107 839, 2022.

[103] C. Zhang, X. Liu, X. Zheng, R. Li, and H. 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)*, IEEE, 2020, pp. 1–4.

[104] F. Zhuang, Z. Qi, K. Duan, *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2020.

[105] I. Silva, J. M. do Valle, G. Souza, *et al.*, "Using micro-services and artificial intelligence to analyze images in criminal evidences," *Forensic Science International: Digital Investigation*, vol. 37, p. 301 197, 2021.

[106] N. Nikolakis, A. Marguglio, G. Veneziano, *et al.*, "A microservice architecture for predictive analytics in manufacturing," *Procedia Manufacturing*, vol. 51, pp. 1091–1097, 2020.

[107] C.-H. Chen and C.-T. Liu, "Person re-identification microservice over artificial intelligence internet of things edge computing gateway," *Electronics*, vol. 10, no. 18, p. 2264, 2021.

[108] I. Sergi, M. Leo, P. Carcagnı, M. La Franca, C. Distante, and L. Patrono, "A microservices architecture based on a deep-learning approach for an innovative fruition of art and cultural heritage," *Journal of Communications Software and Systems*, vol. 18, no. 2, pp. 182–192, 2022.

[109] S. P. R. Asaithambi, R. Venkatraman, and S. Venkatraman, "Mobda: Microservice-oriented big data architecture for smart city transport systems," *Big Data and Cognitive Computing*, vol. 4, no. 3, p. 17, 2020.

[110] S. Trilles, A. González-Pérez, and J. Huerta, "An iot platform based on microservices and serverless paradigms for smart farming purposes," *Sensors*, vol. 20, no. 8, p. 2418, 2020.

[111] A. Mikołajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 international interdisciplinary PhD workshop (IIPhDW)*, IEEE, 2018, pp. 117–122.

[112] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, PMLR, 2017, pp. 1273–1282.

[113] M. A. Khan, M. A. Khan Khattk, S. Latif, *et al.*, "Voting classifier-based intrusion detection for iot networks," in *Advances on Smart and Soft Computing: Proceedings of ICACIn 2021*, Springer, 2022, pp. 313–328.

[114] Z. Ullah, M. I. Mohmand, M. Zubair, *et al.*, "Emotion recognition from occluded facial images using deep ensemble model.," *Computers, materials and continua*, vol. 73, no. 3, 2022.

[115] K.-L. Du and M. Swamy, "Combining multiple learners: Data fusion and ensemble learning," in *Neural Networks and Statistical Learning*, Springer, 2019, pp. 737–767.

[116] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble machine learning*, Springer, 2012, pp. 157–175.

[117] *Docker documentation*, `https://docs.docker.com/`, Accessed: November 6, 2022.

[118] *Docker orchestration*, `https://docs.docker.com/get-started/orchestration/`, Accessed: January 19, 2023.

[119] *Python programming language*, `https://www.python.org/`, Accessed: April 23, 2022.

[120] *Jupyter: Free software, open standards, and web services for interactive computing across all programming languages*, `https://jupyter.org/`, Accessed: May 19, 2022.

[121] *Anaconda*, `https://www.anaconda.com/`, Accessed: May 19, 2022.

[122] R. Smith, *Docker Orchestration*. Packt Publishing Ltd, 2017.

[123] *Malevis dataset*, `https://web.cs.hacettepe.edu.tr/~selman/malevis/`, Accessed: April 23, 2022.

[124] *Create production-grade machine learning models with tensorflow*, `https://www.tensorflow.org/overview`, Accessed: January 19, 2023.

[125] A. S. Bozkir, A. O. Cankaya, and M. Aydos, "Utilization and comparision of convolutional neural networks in malware recognition," in *2019 27th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2019, pp. 1–4.

[126] S. A. Roseline, S. Geetha, S. Kadry, and Y. Nam, "Intelligent vision-based malware detection and classification using deep random forest paradigm," *IEEE Access*, vol. 8, pp. 206 303–206 324, 2020.

[127] Ö. Aslan and A. A. Yilmaz, "A new malware classification framework based on deep learning algorithms," *Ieee Access*, vol. 9, pp. 87 936–87 951, 2021.

[128] W. Wong, F. H. Juwono, and C. Apriono, "Vision-based malware detection: A transfer learning approach using optimal ecoc-svm configuration," *IEEE Access*, vol. 9, pp. 159 262–159 270, 2021.

[129] J. Hemalatha, S. A. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient densenet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, p. 344, 2021.

[130] S. B. Atitallah, M. Driss, and I. Almomani, "A novel detection and multi-classification approach for iot-malware using random forest voting of fine-tuning convolutional neural networks," *Sensors*, vol. 22, no. 11, p. 4302, 2022.

[131] M. Mittal, K. Kumar, and S. Behal, "Deep learning approaches for detecting ddos attacks: A systematic review," *Soft Computing*, pp. 1–37, 2022.

[132] S. Bhatt, P. R. Ragiri, *et al.*, "Security trends in internet of things: A survey," *SN Applied Sciences*, vol. 3, no. 1, pp. 1–14, 2021.

[133] T. Park, D. Cho, H. Kim, *et al.*, "An effective classification for dos attacks in wireless sensor networks," in *2018 Tenth international conference on ubiquitous and future networks (ICUFN)*, IEEE, 2018, pp. 689–692.

[134] S. Ben Atitallah, M. Driss, W. Boulila, and I. Almomani, "An effective detection and classification approach for dos attacks in wireless sensor networks using deep transfer learning models and majority voting," in *Advances in Computational Collective Intelligence: 14th International Conference, ICCCI 2022, Hammamet, Tunisia, September 28–30, 2022, Proceedings*, Springer, 2022, pp. 180–192.

[135] I. Almomani, B. Al-Kasasbeh, and M. Al-Akhras, "Wsn-ds: A dataset for intrusion detection systems in wireless sensor networks," *Journal of Sensors*, vol. 2016, 2016.

[136] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd annual Hawaii international conference on system sciences*, IEEE, 2000, 10–pp.

[137] Y. Zhu, T. Brettin, F. Xia, *et al.*, "Converting tabular data into images for deep learning with convolutional neural networks," *Scientific reports*, vol. 11, no. 1, pp. 1–11, 2021.

[138] *Chest x-ray dataset*, `https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia`.

[139] M. Driss, W. Boulila, H. Mezni, M. Sellami, S. Ben Atitallah, and N. Alharbi, "An evidence theory based embedding model for the management of smart water environments," *Sensors*, vol. 23, no. 10, p. 4672, 2023.

[140] A. D. Gupta, P. Pandey, A. Feijóo, Z. M. Yaseen, and N. D. Bokde, "Smart water technology for efficient water resource management: A review," *Energies*, vol. 13, no. 23, p. 6268, 2020.

[141] H. Mezni, M. Driss, W. Boulila, S. B. Atitallah, M. Sellami, and N. Alharbi, "Smartwater: A service-oriented and sensor cloud-based framework for smart monitoring of water environments," *Remote Sensing*, vol. 14, no. 4, p. 922, 2022.

[142] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE transactions on neural networks and learning systems*, vol. 33, no. 2, pp. 494–514, 2021.

[143] A. Das and P. Rad, "Opportunities and challenges in explainable artificial intelligence (xai): A survey," *arXiv preprint arXiv:2006.11371*, 2020.

[144] F. K. Došilović, M. Brčić, and N. Hlupić, "Explainable artificial intelligence: A survey," in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, IEEE, 2018, pp. 0210–0215.

[145] F. Pourpanah, M. Abdar, Y. Luo, *et al.*, "A review of generalized zero-shot learning methods," *IEEE transactions on pattern analysis and machine intelligence*, 2022.