Investigating AI-Based Approaches for Data Processing on IoT Edge Devices

by

Mahmud ALOSTA

MANUSCRIPT-BASED THESIS PRESENTED TO ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
IN PARTIAL FULFILLMENT FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
Ph.D.

MONTREAL, JUNE 3, 2021

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

FOREWORD

This thesis presents my research carried out between 2015 and 2021 under the supervision of Professor Abdelouahed Gherbi. The main goal of the research is to address and provide solutions that enable best practices for efficient data processing on IoT resource-limited edge devices using AI based techniques.

The dissertation is organized in the form of integrated articles. The articles are included in their original published and submitted form; no changes have been made to them.

For this research work, three articles are included. The articles are inter-related as they describe efficient AI based solutions for data processing on IoT edge devices. A separate chapter is devoted to the background and literature review. The three articles are presented in Chapters 2, 3, and 4. Furthermore, in each of these chapters, a separate section is devoted to the background and related work. Also, a special review is presented on the research problems and the contributions relevant to that particular work.

ACKNOWLEDGEMENTS

I would like to convey my deep appreciation and sincere gratitude to those who have assisted me, in one way or another, through my passionate journey pursuing and realizing my Ph.D. dream.

First of all, I would like to express my sincere and heartfelt gratitude to my advisor Professor ***Abdelouahed Gherbi***, for having faith and trust in me, and for giving me this incredible opportunity to work under his supervision. His guidance, encouragement, and patience during the course of my study have enabled me to fulfill my Ph.D. dream. Dear Gherbi, working with you has been the most rewarding experience of my life and without your support, this dissertation would not have been possible.

Earning a Ph.D. also involves cooperating with other researchers. This sometimes opens up new perspectives and leads to more decent outcomes. I would like to thank Mr. ***Ahmed Bali***, a Ph.D. student and co-author in our lab. Our discussions not only lead to common research results and publications but, more importantly, also to a deep friendship. He kept my motivation up when I needed some backup.

I would like to express my deep gratitude to the committee, Professor Mohamed Cheriet, Professor Alain April, and Professor Rachida Dssouli for their efforts reviewing my thesis and providing insightful feedback.

Many thanks also go out to the Libyan Ministry of Higher Education and Scientific Research for providing the financial support that made this work possible.

I remain indebted to my beloved wife ***(Ahlam)***. Your comprehension, dedication, encouragement, endless support, and ceaseless love during the past years of my Ph.D. journey have sustained me through the challenges and filled my life with joys and tenderness. A special thanks filled with love to my precious children ***(Tiem, Fatima, Yazan, and Hala)*** for cheering me up and bringing joy and happiness to my life.

I am deeply indebted to my beloved mother ***(Fatima)***, father ***(Abdallah)***, brothers and sisters for their sincere prayers and wishes. In spite of the distance, they were an incessant source of encouragement and unconditional support during these years.

Finally, a special thanks to all my friends. The time we spent together and the unconditional support is always remembered.

This work is dedicated to the memory of Ahmed Busbiha (1965 - 2017), big brother, friend, and former ETS Ph.D student.

Étude des approches de l'application des techniques basées sur l'IA pour le traitement des données sur les dispositifs en périphérie de l'IdO

Mahmud ALOSTA

RÉSUMÉ

La diffusion sans précédent des technologies des capteurs et leurs applications dans différents aspects de la vie quotidienne ont largement contribué à l'émergence de l'internet des objets (IdO). L'IdO permet l'intégration transparente des mondes numérique et physique, créant ainsi le formidable réseau de dispositifs connectés dont on ne cesse d'être témoin. Ces dispositifs ont été exploités pour servir une diversité d'applications, notamment la domotique, les transports, la santé et l'agriculture. Fondamentalement, la collecte de données est l'objectif principal de ces réseaux déployés. Ces données sont constamment générées, ce qui conduit à des volumes énormes de données sensorielles brutes qui nécessitent une manipulation supplémentaire pour les convertir en informations utiles.

En général, les données sont transmises au nuage pour être traitées et pour en déduire un résultat utile. En outre, plusieurs cas d'utilisation de l'IdO exigent que le nuage renvoie des instructions pour répondre à des besoins de contrôle. Compte tenu du très grand nombre d'appareils connectés à l'IdO et de la capacité limitée des réseaux, plusieurs défis se posent. Tout d'abord, la transmission sans contrôle des données vers le nuage a amplifié le stress du réseau, ce qui a conduit à un goulet d'étranglement. En outre, la dépendance à l'égard de la topologie du nuage central a augmenté le coût de la manipulation des données de l'IdO. Ce coût peut être considéré comme le coût de la bande passante, en plus du coût opérationnel nuage. En outre, plusieurs applications de l'IdO dépendantes du temps peuvent ne pas tolérer le délai d'attente imposé par l'aller-retour entre les nœuds de capteurs et le nuage. Tous les éléments mentionnés ci-dessus ont mené à la recherche de solutions alternatives permettant de pousser ces capacités de calcul à la périphérie du réseau et à proximité des sources de données. De manière exceptionnelle, cela est devenu possible grâce à la diversité et aux progrès récents des dispositifs de périphérie de l'IdO, qui leur confèrent la capacité de gérer une variété de cas d'utilisation. Cette évolution s'est accompagnée d'une vaste enquête sur la capacité de plusieurs techniques basées sur l'IA à effectuer des tâches de manipulation de données de l'IdO. Plus précisément, plusieurs techniques basées sur l'IA, dont le web sémantique, l'apprentissage automatique et la logique floue, ont été largement étudiées et considérées comme des outils essentiels pour révéler l'ambiguïté des données brutes des capteurs et pour faciliter la prise de décision. Bien que les avantages introduits par ces techniques aient été largement reconnus dans le domaine de l'IdO, la tendance croissante à étendre cette intelligence à la périphérie du réseau a révélé plusieurs défis. Il s'agit notamment de leur applicabilité dans des plates-formes à ressources limitées, ainsi que de leur consistance avec l'instabilité et les fluctuations auxquelles les solutions basées sur la périphérie de l'IdO sont fortement exposées.

L'objectif principal de cette thèse est d'examiner à travers plusieurs études de cas la capacité des techniques basées sur l'IA à maintenir une performance balancée dans les tâches de traitement et d'analyse des données. Cette efficacité est assurée en tenant compte de la limitation des ressources qui caractérise une grande partie des dispositifs périphériques de l'IdO. En outre, chaque étude de cas représente un problème de recherche qui a été traité séparément dans cette thèse. Ainsi, les contributions introduites dans ce travail correspondent aux études de cas présentées et elles sont structurées en objectifs séquentiellement liés entre eux. La première contribution est proposée pour évaluer la faisabilité des solutions de périphérie dans l'exécution de tâches de détection d'événements efficaces et à temps, basées sur le web sémantique et les techniques de traitement d'événements complexes. Dans cette contribution, nous avons examiné la faisabilité du processus d'annotation sémantique dans l'exécution de tâches de détection et de modélisation efficaces des événements, en plus des tâches de filtrage des données sur les dispositifs périphériques de l'IdO. Deuxièmement, afin d'aider les solutions de périphérie de l'IdO non seulement à détecter les événements à temps, mais aussi à les prévoir et à les anticiper, nous avons examiné la combinaison de techniques d'apprentissage automatique et de logique floue. Plus précisément, nous avons étudié et appliqué les meilleurs principes pour le déploiement efficace de solutions basées sur l'apprentissage automatique et la logique floue dans des environnements restreints en ressources. Troisièmement, compte tenu des défis posés par l'utilisation de modèles d'apprentissage automatique en temps réel, nous avons examiné une approche basée sur la logique floue pour assurer le suivi des performances du modèle. Le mécanisme proposé vise à maintenir une performance acceptable du modèle en temps réel conformément à une utilisation balancée des ressources sur les dispositifs périphériques de l'IdO.

Dans tous les cas analysés, nous avons mené une enquête approfondie sur les sujets de recherche connexes et appuyé nos hypothèses par une analyse approfondie de cas d'utilisation réels. En outre, nous avons validé nos hypothèses par plusieurs mises en œuvre de preuves de concept. Cette méthodologie a permis de mettre en évidence une variété de perspectives et de reconnaître les questions ouvertes et les problèmes de recherche potentiels dans le cadre de la recherche présentée.

Mots-clés: internet des objets (IdO), informatique de périphérie, apprentissage automatique, web sémantique, logique floue, analyse de périphérie

Investigating AI-Based Approaches for Data Processing on IoT Edge Devices

Mahmud ALOSTA

ABSTRACT

The unprecedented dissemination of sensor technologies and their applications in different aspects of contemporary life has significantly contributed to the emergence of the Internet of Things (IoT). IoT envisions seamless integration of digital and physical worlds fabricating the ever witnessed stupendous network of connected devices. These devices have been harnessed to serve a diversity of applications including smart buildings, transportation, health, and agriculture. Fundamentally, collecting data is the main objective of such deployed networks. These data are constantly being generated leading to enormous volumes of raw sensory data that require further manipulation to convert it to valuable insight.

Typically, IoT data are transmitted to the cloud for processing and to deduce a meaningful outcome. Additionally, several IoT use-cases require the cloud to send back some instruction to fulfill actuation purposes. Considering the ever-increasing number of IoT connected devices and the limited network capacity, several challenges have arisen. First, blindly transmitting data to the cloud has amplified the stress in the network, leading to network bottlenecks. In addition, the reliance on the central cloud topology has raised the cost of IoT data manipulation. This cost can be envisioned in the bandwidth cost, in addition to the cloud operational cost. Furthermore, several time-sensitive IoT applications might not tolerate the latency imposed by the round-trip from sensor nodes to the cloud. All aforementioned issues have stimulated investigating alternative solutions that push such computing capabilities to the network edge and close to data sources. Auspiciously, this has become feasible thanks to the recent diversity and advancement that has been seen in the IoT edge devices endowing them the capability to handle a variety of use-cases. This also has been simultaneously accompanied with a broad investigation of the applicability of several AI-based techniques to perform IoT data manipulation tasks. More specifically, several AI-based techniques including semantic web, machine learning, and fuzzy logic have been widely investigated and accepted as key enabler to reveal the ambiguity of raw sensor data and to support decision-making tasks. Although the advantages introduced by these techniques have been widely acknowledged in the IoT domain, the growing tendency to extend such intelligence to the network edge has revealed several challenges. This includes their applicability to perform under resource-restricted platforms, in addition to, their consistency in unstable environments that the IoT edge based solutions are highly exposed to.

The main objective of this dissertation is to examine through several case studies the applicability of AI-based techniques in maintaining balanced efficiency in data processing and analysis tasks. This efficiency is obtained taking into consideration the resource-constrained aspect that is featured in a considerable portion of IoT edge devices. In addition, each case study represents a research problem that has been addressed individually in this thesis. Thus, the contributions introduced by this work correspond to the presented case studies and are structured sequentially interrelated objectives. The first contribution is proposed to assess the feasibility of edge-based solutions in performing timely and efficient event detection tasks based on the semantic web and complex event processing techniques. The feasibility of the semantic annotation process in performing efficient event detection and modeling, in addition to data filtration tasks on IoT edge devices was examined. Second, in order to support IoT edge solutions to not only detect events on time but also to predict events and contemplate them in advance, we have examined the combination of Machine Learning and fuzzy logic techniques. More specifically, we have investigated and applied best practices for efficient ML and FL-based solution deployment in resource-limited environments. Third, in view of the challenges of using ML models in real-time, we have investigated a fuzzy logic based approach to keep track of model performance. The proposed mechanism is intended to maintain an acceptable real-time model performance in accordance with balanced resource utilization on IoT edge devices.

In all analyzed cases, we conducted a deep exploration of the related subjects of research and supported our assumptions by a thorough analysis of real-life use cases. In addition, we have validated our assumptions with several proof-of-concept implementations. Such methodology demonstrated progress in highlighting a variety of perspectives and recognizing open issues and potential research issues within the scope of the presented research.

Keywords: Internet of Things (IoT), edge computing, machine learning, semantic web, fuzzy logic, edge analytics

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

AI	Artificial Intelligence
AC	Air Conditioner
ANNs	Artificial Neural Networks
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
BNs	Bayesian networks
CEP	Complex Event Processing
CO2	Carbon Dioxide
CoAP	Constrained Application Protocol
CPU	Central Processing Unit
DT	Decision Tree
DRL	Deep Reinforcement Learning
DUL	DOLCE Ultralite
EC	Edge Computing
EMA	Exponential moving average
EPL	Event Processing Language
FL	Fuzzy Logic
FLFM	Fuzzy Logic Framework Manager
HTTP	Hypertext Transfer Protocol
IAQ	Indoor Air Quality

ICT	Information and Communication Technology
IDC	International Data Corporation
IMO	Indoor Monitoring Ontology
IoT	Internet of Things
ITU	International Telecommunication Union
JSON	JavaScript Object Notation
KNN	K-nearest neighborhood
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MLFM	Machine Learning Framework Manger
MLP	Multilayer Perception
MQTT	MQ Telemetry Transport
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
PH	Prediction Horizon
PM	Particulate Matter
QoS	Quality of Service
RAM	Random Access Memory
RDF	Resource Description Framework
REST	Representational State Transfer

RFID	Radio Frequency Identification
RMSE	Root Mean Square Error
SBCs	Single-Board Computers
SOAP	Simple Object Access Protocol
SOS	Sensor Observation Service
SSN	Semantic Sensor Network
SPARQL	Simple Protocol and RDF Query Language
SVM	Support Vector Machine
SVR	Support Vector Regression
SW	Semantic Web
SW	Sliding Window
SWoT	Semantic Web of Things
TB	Terabyte
UoM	Unit of Measurement
VOCs	Volatile Organic Compounds
WiFi	Wireless Fidelity
WSNs	Wireless Sensor Networks
XML	Extensible Markup Language

INTRODUCTION

Overview

Nowadays, we are witnessing an extraordinary invasion from smart devices. These devices could be attached to almost everything we use, and their capabilities of sensing, gathering, and processing information from the surroundings are constantly improving, which accelerates the movement toward the awareness of smart environments. To achieve broader benefits from these smart environments, a recent trend known as the Internet of Things (IoT) has been widely researched and adopted by both academia and industry. This novel paradigm represents a new era of Information and Communication Technology (ICT) that is foreseen to be one of the future's life pillars Manyika (2015).

IoT refers to a new model of connection that seeks to facilitate seamless integration of the physical world with the cyber-world. IoT networks are expected to connect a plethora of physical objects which have capabilities to interact with their surroundings either by sensing or actuating, and are able to communicate with each other, other machines, or computers. These devices collect data, which can be analyzed further and shared in order to extract knowledge and improve the accuracy of decision-making systems by augmenting them with contextual information that represents the real world. This enables the development of new applications that can potentially contribute to improving several areas of our daily life such as public safety, transport logistics, and environmental management Manyika (2015); Evans (2011).

As shown in Figure 0.1, the adoption of IoT is anticipated to increase, as the number of devices connected to the Internet has rapidly amplified. Several industry mainstream corporations such as Cisco and Ericsson expect that tens of billions of devices will be connected to the Internet by 2020 Ericsson (2015); Evans (2011). Moreover, Gartner (2013) forecasts that IoT will have an economic impact of 1.9 trillion US $ by the same year. This enormous growth is likely to

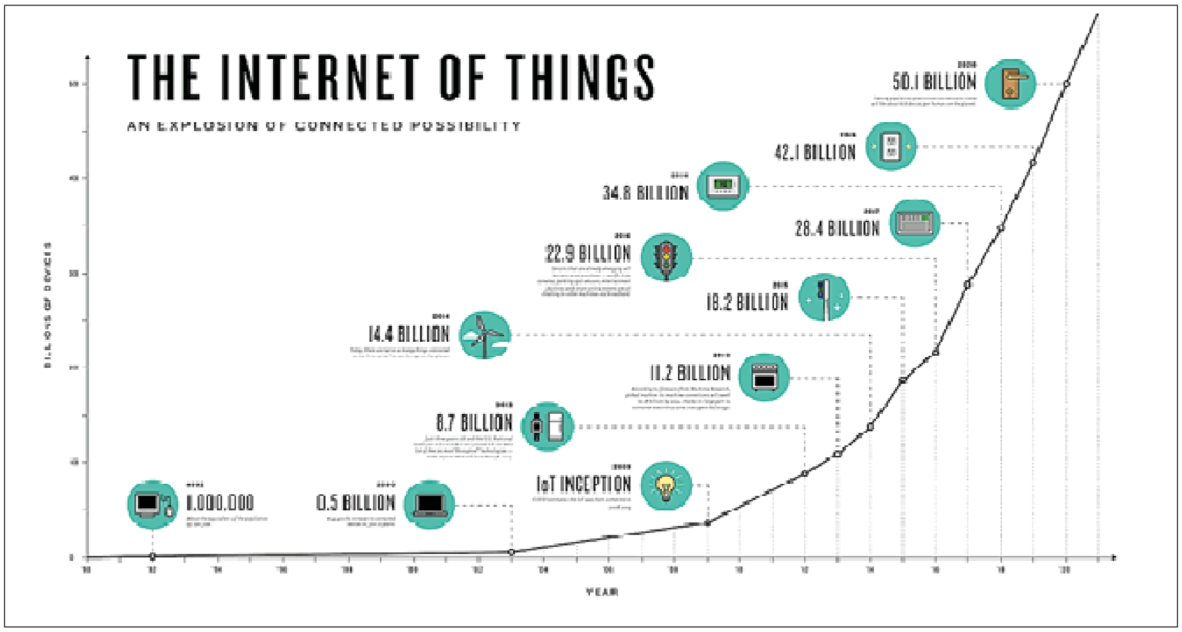


Figure 0.1 The Evolution Stages of IoT networks and Applications
Taken from Evans (2011)

expand the implementation of IoT applications in various areas of our daily life resulting in numerous opportunities for researchers and industries to collaborate to take full advantage of this potential. In addition, recent estimates by the International Data Corporation IDC (2019) shows that by 2025, 55.9B devices will be connected around the globe. These devices are anticipated to contribute to the generation of 79.4ZB of data compared to 13.6ZB of data reported in 2019.

With the large-scale deployment of connected IoT objects and the diversity of their characteristics, new requirements have been imposed to enable the creation of value-added services and applications from the sensor data. Recently, more interest has been shown to develop software and services to enable the capture, interpretation, and action on heterogeneous data packets obtained from IoT endpoints. One key enabler of IoT applications development is to facilitate seamless integration between its heterogeneous components. This heterogeneity has resulted from the diversity of the technology stack that underpins such systems. More specifically, with the rapid growth of IoT, more and more vendors are involved in the development of its systems leading to IoT becoming a collection of different platforms, protocols, and data formats. Mainly, three levels of heterogeneity have arisen due to this diversity of IoT infrastructure. First, device-level heterogeneity emerged as a result of using different communication capabilities, protocols, and hardware platforms. Second, syntax level heterogeneity, which indicates the variety of data type, structure, and formats. Finally, semantic heterogeneity, which refers to the lack of a common understanding of the exchanged data content. Thus, this heterogeneity has become a critical challenge to enable the development of interoperable IoT applications leveraging the diversity of the infrastructure underneath Rahman & Hussain (2019); Razzaque *et al.* (2015).

On the other hand, IoT data packets by most of the current solutions are sent to cloud platforms for further analysis to reap the value which might turn into actionable knowledge. This knowledge can be delivered to IoT applications after extensive analysis using several data mining methods such as machine learning, semantic web, and other intelligent decision-making algorithms. It is conceivable to integrate them into IoT applications to facilitate further analysis tasks and to manifest the ambiguity behind large collected sensory data sets. This type of task is likely to require relatively extensive computing resources, which are mostly available on cloud platforms. This would lead to the development of IoT applications that rely on transferring a large portion of sensor data to the cloud. Taking into consideration the aforementioned IDC estimations, several issues have been triggered as a result of blindly transferring IoT data to central cloud platforms through limited network infrastructure. Moreover, this topology may not efficiently serve some IoT use cases. For example, IoT-based gas monitoring systems might not tolerate the delay from the round-trip of data to be processed on the cloud. Furthermore, increases in the amounts of sensor data to be communicated to the cloud have been strongly linked to some issues concerning network performance including congestion, reliability, and availability, not to mention the cost of moving huge amounts of data and the on-demand cloud resources for processing and storage.

A recent trend that has been widely gaining popularity is to delegate a portion of IoT data processing tasks to devices at the network edge. More specifically, the edge computing paradigm is conceived as a new model for handling the burden of data processing close to their sources. The edge layer already exists in most IoT architecture proposals and it is used as a middle component to connect devices at the sensor node level to the cloud. A considerable portion of devices at the sensor node level are featured by extremely limited resources and their main task is to sense the surroundings and collect data. Collected data is further transferred to the cloud either directly or through gateway devices at the edge layer. Recently, gateway devices have seen significant improvements in terms of enhanced computing resources which have enabled them to carry out some relatively resource-intensive processing tasks besides their main tasks. This leap has contributed to the development of edge-based IoT solutions. These solutions are widely adopted to handle situations where IoT cloud solutions cannot efficiently apply. In addition, edge solutions are featured for their ability to provide a faster response in some critical situations since most of the data are processed and decisions are made close to their sources.

In this thesis, more emphasis has been dedicated to investigate and develop approaches to cope with the increased demand for edge-based solutions. The philosophy behind these solutions is to "bring algorithms to data rather than bringing data to algorithms". The proposed approaches in this thesis are directed to handle several issues related to IoT data processing. First, the data heterogeneity issue is addressed by employing semantic web (SW) techniques at the edge. In addition, SW is also used to perform data filtration tasks and to set the priority of data processing by identifying the most critical events using some complex event processing techniques. Second, machine learning (ML) algorithms have been investigated to identify events prior to their occurrence, thus, the appropriate actions can be planned and taken in advance. The latter task is assigned to some fuzzy logic (FL) based inference engine. Third, to improve the efficiency of ML applications on IoT edge devices, an FL logic-based approach has been investigated and applied to monitor the prediction performance of the ML models used in real-time. These solutions were developed to run on IoT edge platforms taking into consideration their limited resources.

Problem Statement

Despite the significant growth of Internet-connected devices and the span of IoT networks, some challenges still limit the ability to take full advantage of this novel paradigm. Mainly, IoT applications are more likely data-driven and the efficiency of their performance relies on the quality of the collected data. IoT applications are required to convert raw sensory data to information that further might be transformed to some kind of actionable knowledge needed to enable the automation of decision-making tasks. These data are significantly impacted by the underlying infrastructure establishing the IoT systems. More specifically, the heterogeneity characteristic spanned on different levels of IoT systems and devices is reflected in their generated data. This heterogeneity of data format has contributed to the rise of the lack of interoperability issue between IoT system components. This issue has mainly hindered the task of IoT high-level applications and end services to integrate and efficiently interpret raw sensory data collected from a variety of resources. Furthermore, the lower the level of semantic interoperability between the subsystems, the more time-consuming, costly, and error-prone the effort will be to integrate and maintain these systems. This integration and interpretation of data is a prerequisite to enable horizontal integration among vertical silos of IoT applications. More specifically, to realize interoperability among different IoT systems' components and domains.

For further clarification, let's assume a smart building where several IoT gateways are deployed to collect data from different types of sensors. These gateways are connected to central distributed servers deployed in the cloud in charge of large-scale data collection and analysis. It is likely that sensors connected to these gateways come from different vendors and use different technology stacks for communicating their collected observations to a higher level. Furthermore, these gateways which represent sub IoT systems are likely to have their own information model and semantics and each of them evolves individually over time. Thus, it becomes a cumbersome and time consuming task to create new services that can understand information coming from these subsystems. For instance, an energy consumption application might be interested in calculating the average temperature in a building, where a set of temperature sensors are deployed and periodically report their observations to IoT gateways, which further transmit them to the cloud. A major concern would be integrating data from different IoT gateways, where in some cases temperature might be defined as "Temp" while in another gateway it might be defined as "Temperature". Another example, a healthcare application could use humidity sensor readings to issue notifications about the risk of getting infected by some viruses as a result of low levels of humidity. In this case, the application should be able to understand the meaning of sensor humidity readings to integrate them with data coming from external resources, such as the web. All of the aforementioned emphasizes the necessity to present raw sensory data to end-services with the most possible unified and expressive shape that can be smoothly consumed and shared between different IoT systems.

In addition, a considerable portion of IoT solutions is cloud-based. This implicitly involves transmitting IoT raw data, which is estimated to be in the tens of zettabytes by 2025 (according to IDC (2019)), to servers deployed on central cloud platforms for further analysis. The knowledge extracted from such analysis is essential to better perceive the received raw sensory data. Thus, supporting and/or automating the decision-making and planning tasks demanded to be accomplished by IoT systems. However, the ever increasing amount of sensory data generated by large scale deployed networks, and such huge volumes of raw sensory data transferred through network infrastructure has led to some issues. First, blindly sending data forth and back between the cloud and IoT gateways has exacerbated the network bottleneck issue. More specifically, most IoT applications require real time analysis and support real time decision making tasks; hence, any delay caused by transferring data over long distances and congested networks might not to be tolerated. This is significant especially in some time sensitive use cases such as gas monitoring for safety purposes application Fang *et al.* (2019). Another example can be clearly spotted in smart transportation systems, where vehicles generate a large amount of data every second. In such a situation, smart cars need to make the correct decisions on the spot to avoid severe consequences. To enhance real time processing, it is not practical in such a case to send data to the remote cloud for processing because the response time would be too long. Latency resulting from the long transmission of data packets to remote cloud platforms is strongly related to the inefficiency of the end service. Second, a major concern is related to the cost of transferring data to cloud platforms. This issue can be decomposed into two sub issues. Firstly, the cost of the high level of bandwidth consumed to transmit these huge amounts of data to the cloud. In addition, most cloud platforms offer their services based on the concept of pay-as-you-go which means that charges are based on the used resources. Although these platforms have the capabilities to manipulate such an amount of data, this process consumes a considerable amount of resources, where in some cases users would have to expand their storage facility to meet the increasing amount of data. Finally, in some IoT applications such as healthcare, data are very sensitive and users might not be attracted to expose their data to privacy and security breaches that are more likely to accompany cloud-based solutions. The aforementioned challenges have paved the way to investigate new solutions for IoT systems to enhance their ability to process raw data and react to events in a faster real-time fashion. These solutions mostly emphasize providing data processing tasks locally close to their sources.

The aforementioned challenges have paved the way to investigate more efficient solutions by extending cloud capabilities to the network edge, thus enhancing the ability of IoT systems to process raw data and react to events in real time. These solutions are mostly emphasized on delegating a portion of data processing tasks to the devices at the network edge close to their sources. Thanks to the recent efforts to improve storage, computation, and networking capabilities of edge devices, more IoT use cases have become reliant on them not only to maintain communication between cloud and sensor nodes but also to perform some data processing tasks. For instance, edge devices can be utilized to reduce the amount of data transferred to the cloud by performing some kind of filtration algorithm. Also, these devices can improve the quality of services by decreasing latency especially in scenarios where a quick response is essential. However, a considerable bunch of these devices are featured by their relatively limited resources, which limits their capabilities to perform some data analysis tasks that rely on some heavyweight artificial intelligent technologies such as machine learning and the semantic web. These technologies are of significant vitality to decode the meaning behind raw sensory data and to improve IoT service's abilities to detect and identify events in real time, especially at the edge. Events in real life can be deduced from sensed data and identified to facilitate the task of IoT systems in initiating instantaneous decisions. However, in some use cases such as gas monitoring, critical events need to be given priority processing to avoid severe consequences. Furthermore, IoT systems are required to be reactive and also proactive. Thus, IoT edge-based solutions should be equipped with proactive modules to enable them to plan decisions in advance based on real-time predictive analysis over streamed sensory data.

Research Question

The advancement of IoT technologies and the wide span of their application has accompanied a revolutionary advancement in AI technologies and their adoption in different technological contexts. In this thesis, AI technologies namely SW and ML in combination with edge computing have been applied to alleviate the challenges mentioned above. The main objective is to improve the efficiency of IoT edge solutions in performing reliable data processing that leads to accurate event extraction and detection and thus, support IoT systems' decision-making tasks. Gateway devices at the edge are considered as a core component of IoT edge-based solutions. Taking into consideration the challenges introduced in this research domain, this study attempts to shed light on the significance of IoT gateway devices to carry out some relatively resources intensive tasks required to extract and detect events from raw sensory data by trying to answer the following research question (RQ1): To what extent can IoT gateway devices contribute to empowering IoT edge-based solutions and render them suitable in a wide range of pervasive IoT environments?

Amid the plethora of IoT use cases, we investigate four distinct problems as listed below:

- data filtration and semantic annotation (Alleviate heterogeneity and network bottleneck);
- event detection and classification (Improve system responsitivity to critical situations);
- event prediction and decision making (Improve system proactivity and controlling tasks);
- maintaining efficient edge-based real-time prediction tasks (to better reflect any possible change in the streaming sensor data distribution).

Although each use case raises several research questions, this research study concentrates on those aspects in which IoT gateways can be assigned a key role to improve the overall efficiency of IoT applications. This dissertation assumes the capacity of typical gateway devices are rather limited, comparable to Single-Board Computers (SBCs). SBCs are usually not battery-driven devices, characterized by low-power consumption when compared to general-purpose computers. Cost-efficient SBCs like Raspberry pi and BeagleBone are typical candidates for several IoT use cases such as a smart home where relatively light processing is needed. Further, they may be grouped in clusters and become a key enabler of the edge vision where intelligence is pushed out toward data sources, reducing bandwidth and latency as the systems can react to local events Johnston *et al.* (2018). Finally, it is worth mentioning that our thesis does not consider the use of the proposed solutions on microcontrollers or devices characterized by very constrained resources.

Research question (RQ1), as illustrated in Figure 0.2, can be conceived as a meta question for all the remaining RQs. Investigating RQ2 mostly leads to fulfilling the answer of the subsequent RQs (RQ3 to RQ5), each of which can be correlated to a separate research problem.

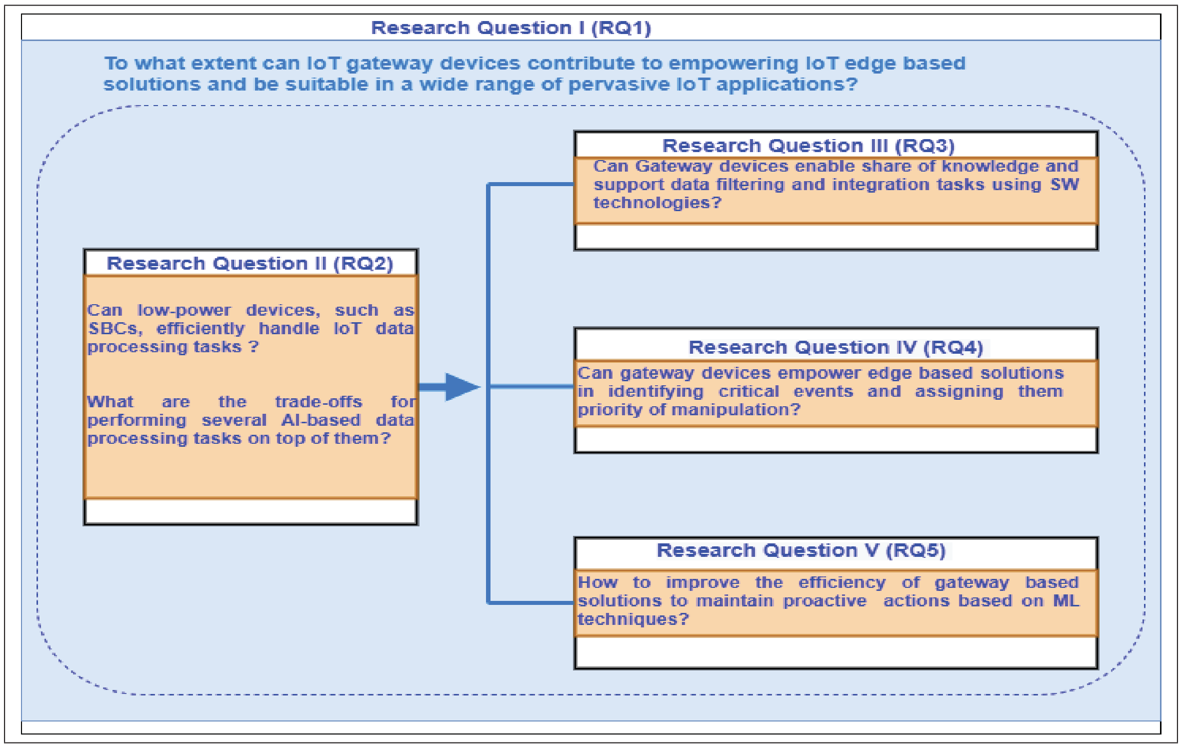

Research Question I (RQ1)
To what extent can IoT gateway devices contribute to empowering IoT edge based solutions and be suitable in a wide range of pervasive IoT applications?
Research Question II (RQ2)
Can low-power devices, such as SBCs, efficiently handle IoT data processing tasks ?
What are the trade-offs for performing several AI-based data processing tasks on top of them?
Research Question III (RQ3)
Can Gateway devices enable share of knowledge and support data filtering and integration tasks using SW technologies?
Research Question IV (RQ4)
Can gateway devices empower edge based solutions in identifying critical events and assigning them priority of manipulation?
Research Question V (RQ5)
How to improve the efficiency of gateway based solutions to maintain proactive actions based on ML techniques?

Figure 0.2 Research questions addressed in this dissertation

Research Aims and Objectives

Taking into consideration the above-mentioned issues related to IoT cloud-based solutions, this research is aimed at proposing IoT gateway-based solutions to improve the overall efficiency of IoT systems. In particular, this study seeks to move the intelligence into the network edge closer to data sources to alleviate the burden resulting from transmitting huge amounts of data to the cloud and to reduce latency, thus improving IoT systems' responsivity. Mainly, in this dissertation, we are keen to investigate the suitability of several AI-based solutions in addressing issues related to IoT data processing tasks. More specifically, the proposed work is intended to meet the increasing demand to maintain data integration and filtration, as well as, identifying events from raw sensory data and acting accordingly. Furthermore, to support prior event manipulation, this research aims to embed predictive models into the IoT data processing pipeline at the network edge.

This research can be conceived as a step forward in the efforts seeking to adopt decentralized IoT edge-based solutions as an approach to distribute the intelligence throughout the network, thus avoiding the reliance on central cloud platforms and their consequent disadvantages. However, it is worth noting that this research is not seeking to completely neglect the role that can be played by the cloud. Rather, the cloud is essential for performing some resource-intensive tasks such as training and testing predictive models where complete and large data sets are available. Also, the cloud is essential in providing large-scale remote device management and controlling tasks.

To summarize, we seek to meet the following sub-objectives:

- proposing and implementing a semantic web based approach for data annotation on IoT gateway devices. The proposed approach should take into consideration the resource limitations of IoT gateways. Thus, it has been equipped with a filtration step that serves two purposes. First, it mitigates and accelerates the semantic data annotation process where not all data need to be annotated. Second, it reduces the amount of data to be later transferred to the cloud, which lightens network traffic and lowers latency. The filtering step is based on a rule engine to filter out redundant and unnecessary data;
- empowering IoT edge based solutions to perform real-time extraction and detection of events from raw sensory data. The intention in this phase is to present an event-driven and semantic based approach for data processing on IoT gateways. The processing pipeline is presented in two levels to first, facilitate event detection and classification, and second, to assign a priority of handling events based on the class assigned in the previous level. This, in turn, enables the development of gateway-based IoT solutions to handle critical and sensitive situations within less time;
- embedding intelligent predictive models into IoT edge based solutions to empower them to perform proactive tasks and to support the automation of edge-based decision-making tasks. Mainly, ML models are trained and deployed on gateway devices to run in real-time and to

deliver as many accurate predictions as possible. These predictions are fed in a further step to fuzzy-based decision-making module. In addition, the fuzzy concept has been employed to monitor and ensure the efficiency of the predictive module. By doing so, we aim to keep our used predictive models up-to-date in terms of their adaptability to any possible change that may influence the distribution of the real-time sensory data.

Research Scope

This research concentrates on providing IoT gateway solutions to emphasize the role that can be played by such cost-efficient and yet simple platforms in empowering the edge layer to improve the overall IoT system performance. More specifically, in this research, we seek to investigate the adoption of several AI technologies and their suitability in performing under resources constrained environments. Namely, SW, ML, CEP, and Fuzzy logic techniques have been evaluated in different use cases to perform several IoT data processing tasks. Their roles are varied from maintaining data filtration and integration to alleviating the heterogeneity and network congestion challenges in addition to improving IoT systems in converting raw sensory data into recognizable events to achieve faster response especially in critical situations. They were also used to enhance the IoT systems' abilities to perform proactively and plan in ahead for actuation instructions to prevent some consequences. This study has also considered assessing the proposed approaches in simulated real-life scenarios, namely gas detection and air quality IoT based applications. The hardware platform used in this research is Raspberry pi 3 equipped with a raspbian operating system, which based on Linux distribution Debian.

Methodology

The following methodology steps were pursued to investigate and accomplish the objectives:

- an initial study was conducted to identify IoT data characteristics, the significance of data integration, the role that can be played by edge devices to process data, advantages of using them, and main technologies that can be used for data processing, as well as, the trade-offs and benefits of running each one of these technologies on IoT gateway devices. Furthermore, this step revealed the major factors that contribute to the exacerbating the data integration process; thus raising the lack of interoperability issue between IoT systems. This step has guided us to identify the heterogeneity of IoT data format and structures, as well as, the absence of unified and shareable understanding as a key obstacle toward achieving interoperability. Also, most investigated solutions rely on the cloud to perform data processing tasks, which leads to increase the pressure on the network;

- an extended review study of the current solutions that are mostly or partly IoT edge oriented. These solutions are intended to run the semantic-based annotation process since it is the key enabler to data integration on higher application levels. In this step, we first identified the limitations of the existing gateway-based semantic annotation approaches. More specifically, the extensive investigation in this step brought our attention to more severe issues that had not been highlighted in the literature. For instance, the current approaches do not consider lightening the annotation process, especially given that the semantic annotation process is mostly dependent on the relatively resource-greedy technology stack. In this case, all sensory data are considered to be annotated regardless of their significance to the application, which would lead to more pressure on the limited computing resources on IoT gateway devices. Furthermore, all these data will be sent in a further step to the cloud, which leads to more bandwidth consumption and in sometimes to network bottleneck. Furthermore, the current literature does not consider the priority of treatment while converting raw sensory data into recognizable and meaningful events. The latter issue impacts the reaction of IoT systems especially in critical events;

- a semantic-based annotation approach for sensory data annotation on IoT resource limited gateway devices was developed. In this work, we sought to provide answers for both RQ2 and RQ3. More specifically, a lightweight semantic based annotation approach has been proposed that is intended to meet the conditions of limited resources on gateway devices and following requirements. First, minimizing resources required for the annotation process; to achieve this, we have designed data aggregation and filtering mechanisms to filter-out unnecessary data based on a rule engine reducing the amount of data to be annotated and further sent to the cloud. Second, lighten the annotation process by concentrating on annotating the data that is mostly queried and used by end applications;

- in order to respond to RQ2, RQ3, and RQ4, we have developed and assessed an event-driven and semantic-based approach for IoT data processing on gateway devices. This approach tackles IoT data processing tasks on two levels. The first level is intended to convert raw sensory data to recognizable events and classify these events based on predefined classes. These classes are used to assign a priority of treatment to the data flows streamed to the gateway, thus, enabling IoT systems to react faster to the events and in real-time. On the second level, data events generated from the first level are transformed, and two types of processing are applied. The first type is filtering incoming data using a set of semantic rule to filter out redundant and insignificant data. The second type of processing is where the semantic annotation process takes place. It aims at tagging the raw sensory data with a shared set of vocabularies; thus, it can be smoothly consumed by different high-level IoT applications and services;

- in order to respond to RQ2 and RQ5, in this methodology step we focused our attention on the investigation of ML algorithms to improve IoT edge-based solutions in predicting future events from sensed data. These predictions will be used to feed a fuzzy logic based inference engine intended to convert these predictions into actionable knowledge based on predefined fuzzy rules. Also, the fuzzy logic is intended to perform monitoring tasks that keep track of the performance of the used ML models. The significance of the latter step lay on its efficiency in maintaining the quality of the model that might be impacted by the change of the distribution of the data generated by different sensors connected to the gateway. Several comparison experiments were launched to empirically assess the performance of the selected ML models. An indoor air quality dataset was chosen to demonstrate the feasibility of the proposed approach. It was divided into two parts. The first is used to train and test the ML models under assessment, while the second is used to simulate virtual sensors that send their data to the gateway device. Several benchmarks were evaluated such as resources, namely CPU and RAM, consumption, as well as, inference time and model performance degradation time.

Each of the above-listed methodology steps is reflected in the collection of publications that form the basic content of the presented thesis. The used research methodology is more inclined toward system design and empirical experimentation. Typically, the design of the approaches presented in this study are scrutinized based on the practical insight learned by prototyping and then the shortcomings of the design are analyzed. Moreover, the methodology employed to assess the feasibility of the implemented prototypes under different cases consider several aspects including response time, data size, cost, and scalability, as well as, resource consumption. Finally, the overall methodology has been devoted to addressing RQ1 which generally represents the main contribution of this research.

Contributions

The main knowledge contributions that this dissertation makes are as follows:

- it theoretically and empirically demonstrates the significance of IoT edge based solutions in empowering the overall performance of IoT systems;

- it presents an IoT edge gateway design that combines the implementation of several AI technologies in order to facilitate smooth and faster sensory data processing and manipulation. The proposed designed approaches enhance the suitability of IoT gateway devices as a means to alleviate several burdens resulting from communicating most of the sensory data to the cloud;
- it introduces a new approach for semantic data annotation on IoT gateway devices. The presented approach is distinguished from the reviewed literature by presenting the filtration process before annotation. The semantic-based filtration model has been introduced to enable the creation of filtering rules. These rules can be configured to enable the selection of only the most significant data to be sent to the cloud;
- it presents and tests a modular approach that facilitates the detection and manipulation of events-of-interest. The event detection and classification modules are designed based on Complex events processing concepts. While, the manipulation module, which is divided into semantic filtering and annotations sub modules, is designed based on semantic web concepts;
- it outlines the significance of ML algorithms in promoting proactive behaviors of IoT edge-based solutions and performing real-time predictive tasks. This hypothesis has been evaluated in the context of an IoT-based air quality monitoring system. The proposed approach also presents the role that can be played by the fuzzy logic concept in supporting automatic decision-making tasks as well as in performing real-time monitoring of the performance of the used ML models.

Publications

- Mahmud Alosta, Ahmed Bali, and Abdelouahed Gherbi. "A lightweight semantic web-based approach for data annotation on IoT gateways." Procedia computer science 113 (2017): 186-193;

- Ahmed Bali, Mahmud Alosta, and Abdelouahed Gherbi. "An ontology-based approach for IoT data processing using semantic rules." International SDL Forum. Springer, Cham, 2017;

- Mahmud Alosta, Ahmed Bali, and Abdelouahed Gherbi. "Event driven and semantic based approach for data processing on IoT gateway devices." Journal of Ambient Intelligence and Humanized Computing 10.12 (2019): 4663-4678.

- Mahmud Alosta, Ahmed Bali, and Abdelouahed Gherbi. " Machine Learning and Fuzzy Logic Approach for Event Detection on IoT Edge Devices." Engineering Applications of Artificial Intelligence. Under Review.

- Mahmud Alosta, Ahmed Bali, and Abdelouahed Gherbi. "A Fuzzy Logic Based Approach for Efficient Machine Learning Applications on IoT Edge Devices." Concurrency and Computation: Practice and Experience. Under Review.

Thesis Organization

Since this work is article-based, we begin by discussing the general background and literature review. Thereafter, we provide details for each of our publications in dedicated chapters. Finally, we conclude the thesis and draw some future directions from the remaining open research questions.

CHAPTER 1

BACKGROUND AND LITERATURE REVIEW

This chapter reviews some background information related to the Internet of Things including (IoT) the evolution of the IoT definitions from different perspectives and its architecture, the thing concept, IoT deployment models, the application, and data characteristics. Additionally, some technologies that significantly promoted the realization of the internet of things are discussed.

1.1 Internet of Things Definition

The tendency towards embedding microprocessors in everyday objects, so they can communicate information has been researched extensively the last two decades, specifically when Mark Wiser presented his vision about ubiquitous computing Ye *et al.* (2015). The vision of ubiquitous computing implies constant connectivity and availability of different computing devices that expose several services. Typically, these services are accessible via a "technology-rich" environment, which makes them more likely to be influenced by their surroundings (i.e., context), more specifically, to the person, the time, and the place of their use. Certainly, this vision has constantly motivated the novelty behind IoT supported technologies such as connectivity, miniaturization, and intelligent information processing Ye *et al.* (2015).

Internet of Things (IoT) represents a diversified paradigm that involves the interaction between a diversity of hardware and software technologies to create new applications/services aiming at accomplishing common goals. This technological diversity has led to inconsistency with respect to a unified definition of the term Internet of Things that meets different perspectives and can be agreed upon by the world community of users Rose *et al.* (2015). In particular, IoT is a multi-perspective vision that has attracted many different groups including academicians, researchers, practitioners, innovators, and developers that have each defined the term differently. However, all these definitions commonly agreed upon the idea, that while the first version of the Internet is about data generated by people, the next wave is about data generated by things Rose *et al.* (2015).

The term Internet of Things was firstly coined by Kevin Ashton in 1999 to describe a tracking system that was designed by attaching RFID tags to some objects and connecting them to the Internet. Since then, several definitions have been proposed that reflect different perceptions of the IoT. Atzori *et al.* (2010), emphasize that any IoT definition should take into consideration three main perspectives. The first perspective represents the "Internet" part of IoT, which deals with various communication infrastructures between devices, systems, and users. The second dimension corresponds to the "Things" part of the IoT, which focuses on enabling interaction between the "smart" physical objects as well as with the users. This area has produced an identification technology such as RFID for integrating physical objects into IoT. Third, the engineering aspects of distributed computing are concerned with access to physical things and devices, maintaining a network of things, and retrieving useful information from massive and presumably inconsistent data generated by IoT. This field has produced middleware approaches, applying software-oriented architecture for IoT, using semantics to discover devices, as well as mining information from IoT data Atzori *et al.* (2010).

Another definition by the CASAGRAS project defines IoT as "A global network infrastructure, linking physical and virtual objects through the exploitation of data capture and communication capabilities. This infrastructure includes existing and evolving Internet and network developments. It will offer specific object-identification, sensor, and connection capability as the basis for the development of independent cooperative services and applications. These will be characterized by a high degree of autonomous data capture, event transfer, network connectivity, and interoperability." Casagras (2009). While in Gubbi *et al.* (2013), the definition is tended mostly toward the role of IoT in data gathering and sharing of information and is as follows: "IoT for smart environments is an interconnection of sensing and actuating devices, providing the ability to share information across platforms through a unified framework, developing a common operating picture for enabling innovative applications. This is achieved by seamless large-scale sensing, data analytics, and information representation using cutting edge ubiquitous sensing and cloud computing."

In summary, this diversity of IoT definition has brought thought to many researchers that the IoT is just a new term that refers to other fields of research such as ubiquitous computing, and distributed systems. Nevertheless, other researchers believe that IoT exploits these fields as implementation approaches aiming at achieving synergies between things on the scale of the Internet.

1.2 IoT Architecture

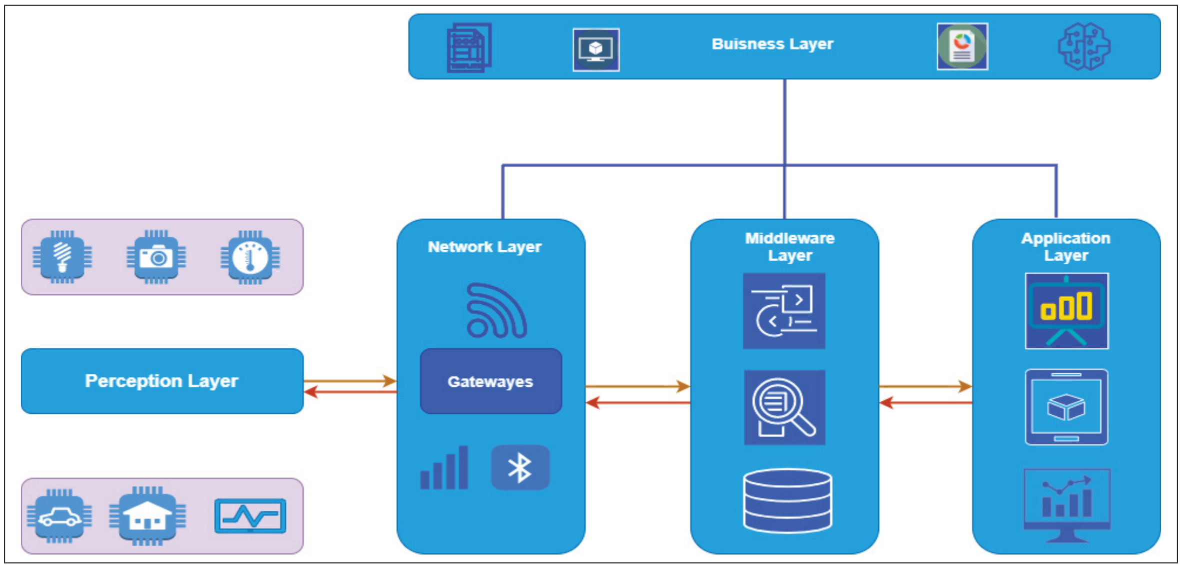


Figure 1.1 Internet of Things Layered Architecture.

The IoT as a domain of research is still maturing; therefore a unified and commonly agreed upon unique IoT architecture has not been projected yet. Also, several factors influence the IoT architecture design process, such as scalability, interoperability, data storage reliability, and Quality of Service (QoS) Wu *et al.* (2010); Gubbi *et al.* (2013). Consequently, several proposals in the literature have presented different IoT architectural views. While some of them rely on the basic three-layer architecture (Perception, Network, and Application layers), others tend to be more abstract and propose five-layer architecture by adding two more layers (Middleware and Business layers) Mashal *et al.* (2015). Figure 1.1 depicts a five-layered architecture model that mostly meets the application development requirements. We briefly discuss them in the following subsections Mashal *et al.* (2015); Gubbi *et al.* (2013); Mashal *et al.* (2015); Wu *et al.* (2010) :

- **perception Layer:** it is also called the device layer and it represents the wide spectrum of sensors and actuators that form IoT systems. They are mainly deployed to run several functionalities and to serve different needs such as acquiring data to explore the environment such as temperature and humidity or to explore location, weight, pressure, etc. The perception layer is mostly reliant on some kind of plug-and-play mechanism to configure a set of heterogeneous objects. The key technologies of this layer include RFID, sensors, nanotechnology, etc;

- **network Layer:** this layer is in charge of assigning a unique address to each object connected to the network. Also, it maintains a secure bidirectional transmission of the information between the perception layer and the application layer. Several short and long-distance communication protocols can be a part of this layer such as Bluetooth, ZigBee, WiFi, and Lora;

- **middleware Layer:** this layer can be conceived of as an abstraction layer that functions on top of the network layer. Its main objective is to enable smooth hiding of the complexity of the underlying hardware components of IoT systems. It is also known as service management that pairs a service request with its corresponding service provider based on a name or address. Moreover, the middleware layer carries out some tasks including processing received data, making decisions, and delivering services upon request over the network layer. Several application protocols are part of this layer such as MQTT, HTTP, REST, and COAP. In addition, this layer employs several computing technologies such as database frameworks, cloud computing, and intelligence processing techniques;

- **application Layer:** this layer is mainly intended to offer end services upon customer request. This can be observed in several distinct use cases such as indoor air temperature and humidity measurements delivery to users who are interested in this information. The significance of this layer lays in its ability to hide the complexity of the underlying layers and provide high-level smart services to respond to users' needs. Furthermore, it is likely to be directed to cover numerous vertical market sectors such as smart home, smart building, transportation, industrial automation, and smart healthcare;

- **business Layer:** this layer is in charge of managing the overall IoT system activities and services. More specifically, it uses data received from the application layer to construct conceptual business models, graphs, flowcharts, etc. In addition, this layer is also intended to support several tasks related to the development of IoT systems' components such as design, analysis, implementation, and evaluation. Furthermore, big data tools can be integrated with the business layer to support decision-making tasks. Finally, this layer performs the monitoring and management tasks of the aforementioned four layers. It compares the output of each layer of the IoT system to a predicted output to promote service enhancement and users' privacy.

1.3 The Thing within Internet of Things

The "thing" in the IoT vision is an umbrella term that refers to a wide spectrum of different physical objects, and also includes physical places and persons. Physical objects can be tracked or controlled remotely by easily attaching low cost sensors or RFID tags to them. Similarly, sensors can be used to monitor places like buildings or rooms, and things in the environment such as rivers or glaciers. Furthermore, wearable sensors can be attached to people, so they can monitor and collect data concerning human body activities. In Carrez (2009) the authors declared the term "entities of interest" to refer to things within the Internet of Things. This term can refer to any object that comprises a set of attributes, which can be used for describing it and its state relevant from a user or application perspective. While in Haller (2010), the author discussed the distinction between entities of interest and devices in IoT. He described the entity of interest as the object that has some value for the observer, while the device is a technical component required to observe or interact with the entity of interest.

In order to connect one or more entities of interest and make them reachable through the Internet, technical communication devices (e.g., sensors or actuators) are required. Attaching or embedding these devices into the entities forms what we call smart devices Meyer *et al.* (2015). Obvious example of such devices are sensors and actuators, RFID readers, mobile phones, and embedded computers. These devices are considered as a subset of all things connected to IoT. However, they are significantly different from any other things due to the computational resources that enable them to link entities of interest to the Internet. For instance, they offer information about the thing, as an identifier or sensed data, and they may provide actuation capabilities as well. To enable smooth access to resources from the application level, services are required Meyer *et al.* (2015). These services can reach resources in two ways, directly or indirectly. While the former is achieved through a service interface exposed directly by resources, the latter employs a network service called a proxy that establishes a connection with the actual resource. In addition, this model can provide an extra level of aggregation and abstraction Haller (2010). Two main service paradigms have been widely used to access resources, namely RESTful for resources direct access and SOAP for indirect access.

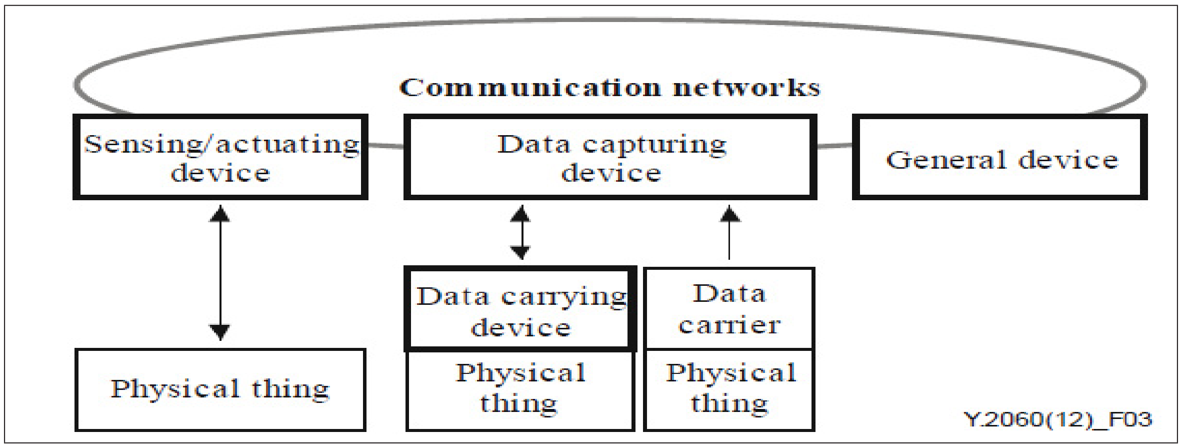


Figure 1.2 Types of IoT Devices and their relationship with physical things
Taken from SECTOR & ITU (2012)

Figure 1.2 shows IoT devices classification by SECTOR & ITU (2012), which depicts the relation between devices and physical things according to their relations to data. Also, ITU acknowledged that the support of communication capabilities is the minimum requirement of the devices in the IoT. In addition, ITU has identified four categories of devices as follows:

- **data-carrying device:** it is likely to be attached to the entity of interest and acts as a communication bridge that maintains the indirect connection between physical things and the communication networks;

- **data-capturing device:** it is featured by its read/write capabilities and it either directly or indirectly interacts with the physical thing. The direct interaction is performed using data carrier devices attached to the physical things, while indirect interaction is achieved by reading/writing information on a data-carrying device;

- **sensing and actuating device:** devices which fall under this group are likely to have two distinct capabilities. First, it can detect and measure observations in the surrounding environment. Second, it can convert these observations into digital electronic signals. Furthermore, detected digital signals can be converted to actions using this category of devices. Typically, sensors and actuators within the same network can communicate using either wired or wireless communication technologies, as well as using gateways to connect to the information networks;

- **general device:** this type of device can be envisioned as a device that originally comes with embedded processing and communication capabilities; so it is able to communicate with the communication networks via wired or wireless technologies. Home electrical appliances, industrial machines, smartphones, and PCs are examples of this category that can be seen in different IoT application domains.

1.4 Internet of Things Deployment Patterns

Due to the diversity of IoT applications, different communication models have been proposed by Tschofenig *et al.* (2015). These models characterize the ways that devices can be used to connect to the Internet and provide services to the user. The Internet Architecture Board describes four communication models: Device-to-Device, Device-to-Cloud, Device-to Gateway, and Back-End Data-Sharing.

1.4.1 Device to Device

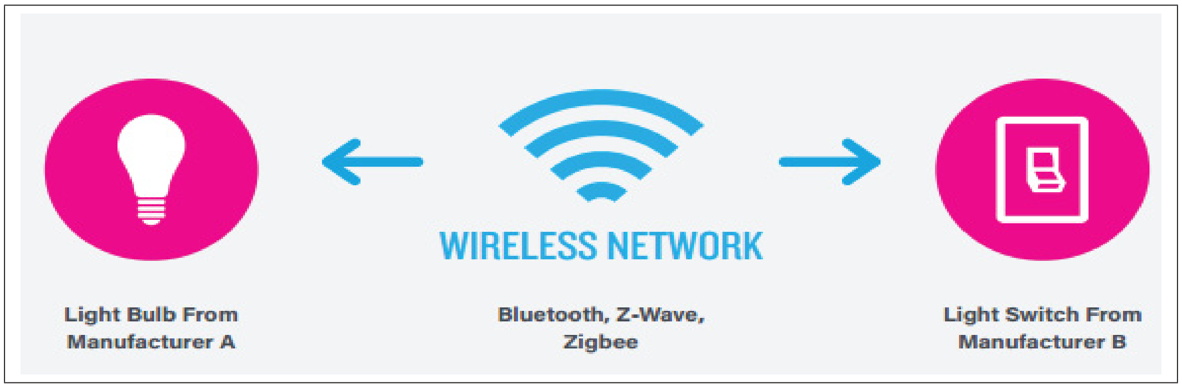

WIRELESS NETWORK
Light Bulb From Manufacturer A
Bluetooth, Z-Wave, Zigbee
Light Switch From Manufacturer B

Figure 1.3 IoT Device to Device Communication Model
Taken from Rose *et al.* (2015)

Device-to-Device network represents an approach to directly connect two or more devices using short-range communication protocols such as Bluetooth, Z-Wave, and ZigBee. Home automation systems usually utilize this model which enables devices that use the same protocol to exchange small data packets with each other to accomplish specific functions. Figure1.3 shows an example of using this model. The main drawback of the Device-to-Device model is that it limits user choice to devices within a particular protocol family Tschofenig *et al.* (2015); Rose *et al.* (2015); Yu *et al.* (2017).

1.4.2 Device to Cloud

Device-to-Cloud communication implies that the IoT device directly exposes its service to an Internet cloud service provider for data exchange and message traffic control. Usually, this type of communication establishes the connection between devices and the Internet, and afterward to the cloud service over traditional wired Ethernet or Wi-Fi connections as shown in Figure1.4. This type of communication enables end-users to remotely access their devices via a smartphone or web interface. In this scenario, devices and cloud services frequently belong to the same provider and use the same protocol to exchange data; thus interoperability challenges can arise

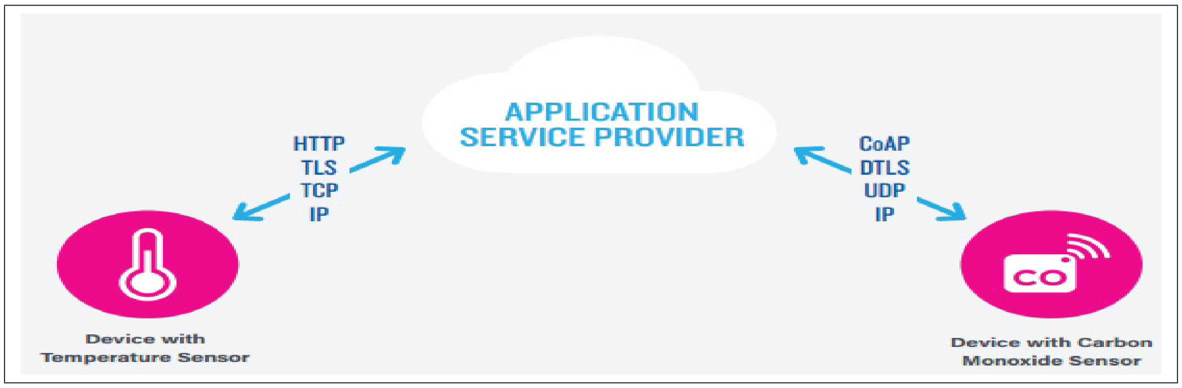

APPLICATION SERVICE PROVIDER
HTTP
TLS
TCP
IP
CoAP
DTLS
UDP
IP
CO
Device with Temperature Sensor
Device with Carbon Monoxide Sensor

Figure 1.4 Device to Cloud Communication Model
Taken from Rose *et al.* (2015)

when attempting to integrate devices made by different manufacturers Tschofenig *et al.* (2015); Rose *et al.* (2015); Yu *et al.* (2017).

1.4.3 Device to Gateway

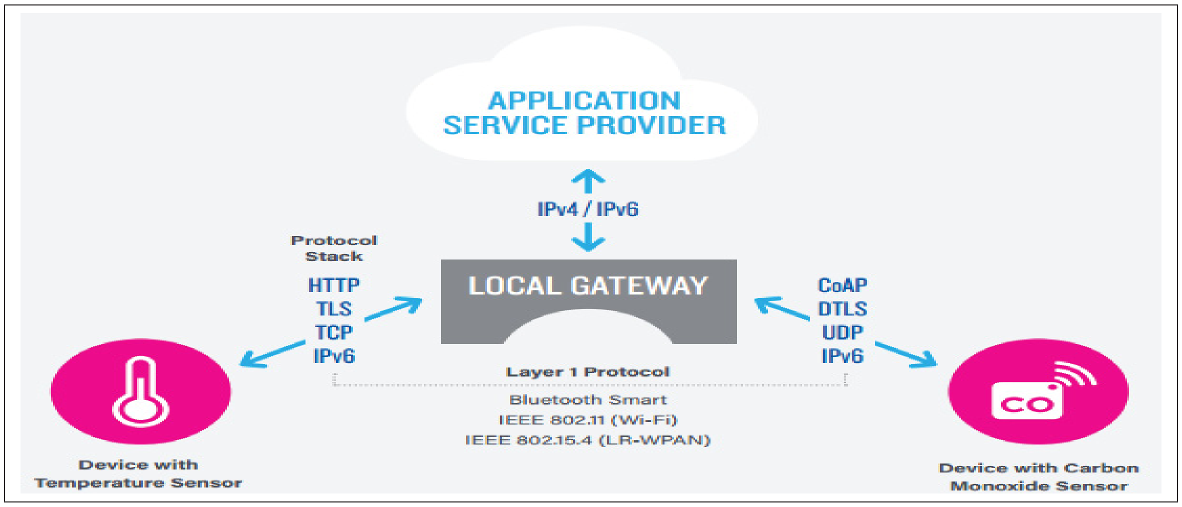


Figure 1.5 Device to Gateway Communication Model
Taken from Rose *et al.* (2015)

Unlike the previous model, the Device-to-Gateway model indirectly interacts with the Internet cloud service provider using an intermediary device called a gateway. This scenario often emphasizes the existence of an application software that runs on a local gateway device which acts as a medium of communication between an IoT device and a cloud service, and offers security. It can be employed to map between different data protocol thus mitigating the interoperability challenge. A smartphone can be used as a gateway; in this situation, application software usually takes the form of an app that pairs with the IoT device and communicates with a cloud service Tschofenig *et al.* (2015); Rose *et al.* (2015); Yu *et al.* (2017).

1.4.4 Back End Data Sharing

Back-End Data-Sharing can be envisioned as an extended form of a single Device-to-Cloud communication model. This model allows a third-party authorized user to reach sensor data stored in certain cloud services and combine it with data from other sources, and send it to another service for aggregation and analyses. This model partly overcomes the data-silos issue caused by uploading captured data to only one cloud service provider Tschofenig *et al.* (2015); Rose *et al.* (2015); Yu *et al.* (2017).

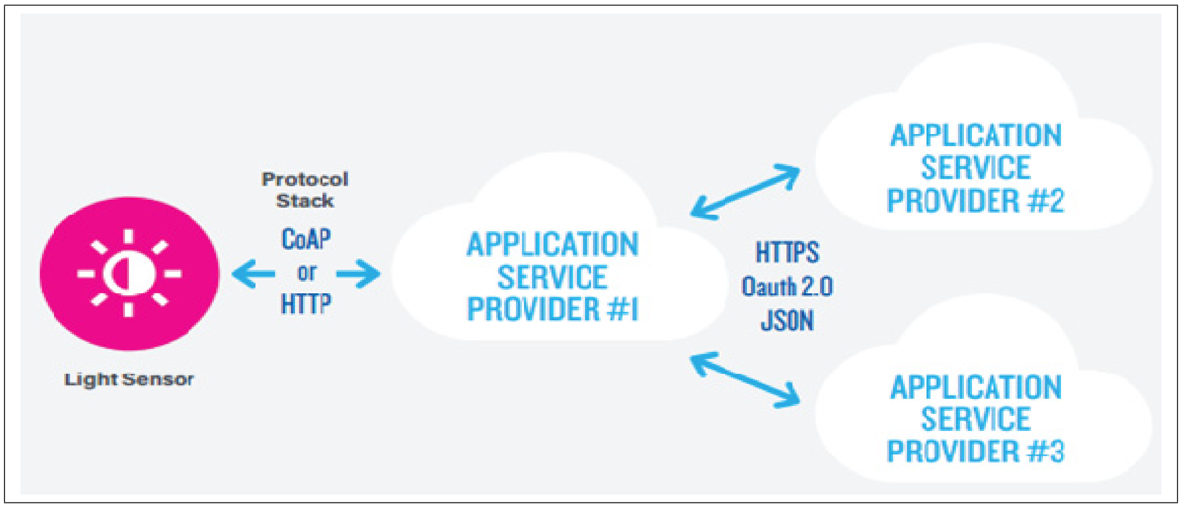

Protocol Stack
CoAP or HTTP
Light Sensor
APPLICATION SERVICE PROVIDER #1
HTTPS
Oauth 2.0
JSON
APPLICATION SERVICE PROVIDER #2
APPLICATION SERVICE PROVIDER #3

Figure 1.6 Back End Data Sharing Communication Model
Taken from Rose *et al.* (2015)

In summary, these models are conceived as the underlying design strategies intended to enable communication between IoT devices. In addition to some technical perspectives, the adoption of these communication patterns is strongly influenced by the nature of connected IoT devices.

To overcome restrictions imposed by using a proprietary device, the Device-to Gateway model has been widely implemented. This point emphasizes the vitality of maintaining device interoperability and shared open standards as a key element in the development of IoT systems Rose *et al.* (2015).

1.5 Internet of Things applications

The Internet of Things vision is forecasted to change the landscape of the ICT, since it has opened the way toward the development of a diverse spectrum of innovative applications in many domains, i.e., cities and homes, environment monitoring, health, energy, and business. In this section, we discuss the characteristics and development approaches of IoT applications.

1.5.1 Characteristics of Internet of Things applications

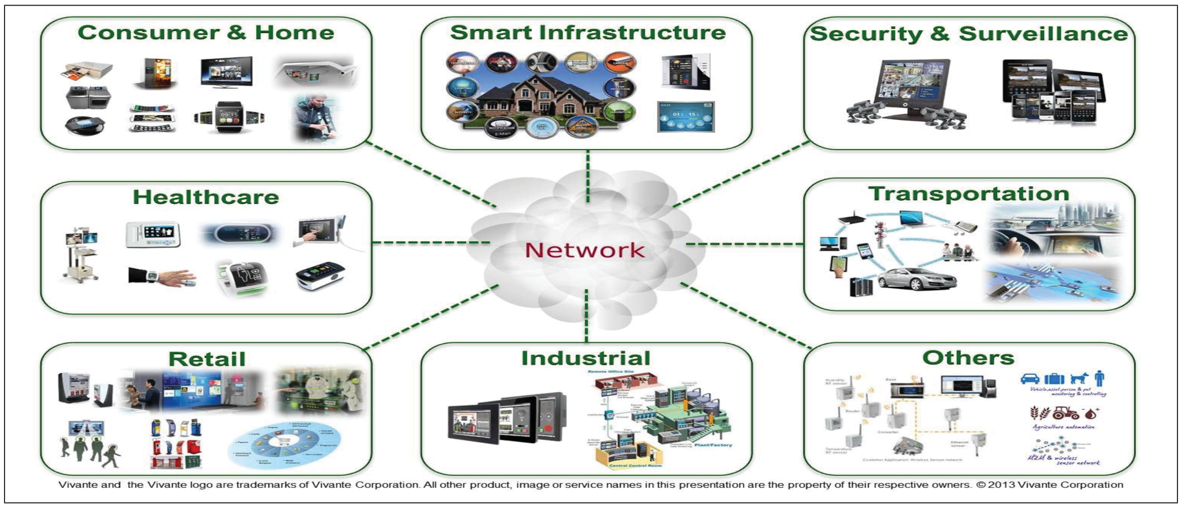

Consumer & Home
Smart Infrastructure
Security & Surveillance
Healthcare
Network
Transportation
Retail
Industrial
Others
Vivante and the Vivante logo are trademarks of Vivante Corporation. All other product, image or service names in this presentation are the property of their respective owners. ©2013 Vivante Corporation

Figure 1.7 Domains of IoT Applications
Taken from B.Tao (2013)

1) **Diversity:** recently, IoT networks have proliferated significantly, and are capable of exposing their services to a wide range of applications in numerous domains and environments. Authors in Patel *et al.* (2013) group these domains and environments into five domain categories as follows: 1) transportation and logistics; 2) healthcare; 3) smart environment

(home, office, and plant); 4) industrial; and 5) personal and social domain. Moreover, Figure 1.7 highlights some potential application domains for the IoT. It is likely that these different applications would demand different requirements, and different deployment architectures;

2) **Real time:** IoT applications can be generally categorized based on the used data as real time and non-real time. In applications like healthcare, transportation delivery of real-time data is vital, and any delay can negatively influence the application or service, which might cause a catastrophe in mission-critical applications B.Tao (2013); Patel *et al.* (2013);

3) **Topology:** two topology models categorize IoT application, static or dynamic. While in the former model, devices involved in the IoT application are fixed, so do not move once they are installed, devices in the latter model are autonomous, and are not tied to a certain location. An example of the dynamic topology is speed sensor and presence sensor that is installed on the highway and employed in road traffic monitoring and control application. In the static topology, applications may employ devices such as wearable devices (e.g., smartphone, badge) to monitor or track moving objects such as a person or animals B.Tao (2013); Patel *et al.* (2013);

4) **Scale:** refers to the number of devices involved in a certain task application. For instance, in smart office applications, calculating the average temperature of a room requires very few temperature sensors while calculating the daily temperature of a large building requires the use of a large number of devices with temperature sensors B.Tao (2013); Patel *et al.* (2013);

5) **Heterogeneity:** typically, IoT applications run on a network of connected devices, these devices are either homogeneous or heterogeneous in terms of hardware and software specifications. A homogeneous network consists of devices that are mostly identical from a hardware and software standpoint. On the other hand, a heterogeneous network consists of devices of different types of sensors and actuators. For instance, a fire management application consists of different kinds of sensors such as temperature sensor, smoke detector and different kinds of actuators such as light screen, alarm, etc. These devices may be

different in terms of their implementations. For instance, one temperature device produces data in Celsius and the other in Fahrenheit B.Tao (2013); Patel *et al.* (2013);

6) **Data Delivery Models:** IoT applications are classified based on models used to interact with devices to deliver sensed data. These models manage the generation of the application traffic. Generally, there are four models, as described below Patel *et al.* (2013):

   - **event-driven:** in this model, sensors record data only as a response to an event of interest;
   - **continuous:** data are constantly transferred by sensors to the sink at a pre-specified rate;
   - **query-driven:** in this model, occurrence of a certain event is a necessity to send data from sink node to sensor node;
   - **hybrid:** two or more data-delivery models function on the same network.

7) **Goal-Driven:** an IoT application either has a sense-only goal or a sense-compute-actuate goal:

   - **Sense-Only (SO):** an application with this objective collects data and stores it for later, off-line analysis. A smart office application is a typical example, where information about a building, such as average temperature, is stored and can be recorded for a later purpose;
   - **Sense-Compute-Actuate (SCA):** in this category, the application collects data, then processes it, and generates appropriate action. For instance, the fire management system that analyzes data captured by a smoke detector and temperature sensors alerts residences by triggering alarms. This class of applications is found in domains such as optimizing power consumption costs, or workplace safety, etc Patel *et al.* (2013).

8) **Entity type:** refers to the type of devices involved in a certain application. To elaborate, it implies whether these devices are sensors or actuators, virtual or physical, or diverse. For instance, in a smart office application, when a user enters or leaves a room, a storage service (i.e., virtual entity) is queried for the user's preference. Based on the user's preference, the heater (i.e., physical entity) is triggered Patel *et al.* (2013).

1.6 Internet of Things Data Characteristics

The Internet of Things paradigm is anticipated to unprecedentedly fuel the current business process with an enormous and diverse amount of real time data. These data can be converted to a variety of services and applications and used to facilitate our daily life activities. These data, generated by different resources, have several characteristics Aggarwal *et al.* (2013); Ding *et al.* (2011); Shemshadi *et al.* (2016); Ma *et al.* (2013)

1) **Heterogeneity:** an IoT system may contain various kinds of sensors whose sampling data have heterogeneous data structures. For instance, in the smart system scenario, the same IoT system may encompass several categories of sensors such as meteorological sensors, hydrological sensors, geological sensors, biomedical sensors, traffic sensors; furthermore, each category can be divided into different kinds of sensors. For example, traffic sensors can include GPS sensors, RFID readers, video based traffic-flow analysis sensors, traffic loop sensors, road condition sensors, and so forth. The sampling data from different sensors can have different semantics and data structures which greatly increases the difficulties in the data processing. Obviously, data from the different sources mentioned can be, but are not limited to numerical, text (JSON), and XML. The structure of data can be of structured data (such as standard records), semi-structured and non-structured data (such as video, audio, and other multimedia data);

2) **Massive scale:** as the number of devices connected to the Internet is constantly increasing, and these devices are contentiously gathering data from the monitored objects, a massive amount of heterogeneous real time data is expected to be generated. This phenomenon has raised the scalability issue since such an amount of data would demand large storage platforms and powerful processing mechanisms. Consider a retail store, where millions of items are available daily. If these articles need to be tracked per day, and each tracking process generates 100 bytes of data, the total of constantly produced data may reach 100 GB daily and 36.5 TB in a year to support object tracking and discovery. The scalability

issue demands employing intelligent techniques to classify data and accelerate the search and discovery process for data of interest from an IoT application perspective;

3) **Temporal-spatial:** data collected by IoT systems generally corresponds to a location and a time instant describing where and when the value is sampled. Based on the location attribute, objects in IoT can be classified into two categories mainly static or dynamic. The static attribute refers to sensors that are installed in a certain location to observe an entity of interest, for instance, a temperature sensor fixed in a certain room or office. On the other hand, the dynamic attribute implies that the sensor object can move over time, for instance, GPS sensors and video-based traffic flow sensors installed in buses;

4) **Implicit semantic:** row IoT data is low-level, lacks semantics, and is multidimensional. To better integrate such data and exploit it in higher-level applications, intelligent mechanisms are required to manipulate and abstract complex semantics. Essentially, to automate the process of information communications and interactions in IoT, it is a prerequisite to have precisely described data; so that machines and software agents can process and interpret them. As an example, when a physician justifies a symptom of a patient, several vital signs such as blood pressure and heart signal are usually monitored;

5) **Timeliness:** a thing's state in the IoT environment is mostly influenced by real-time changes of its surroundings. Thus, it is vital that IoT data is being sent to the server swiftly and regularly. Although several IoT applications employ historical data, it is definite that exploiting timely data can define the status of objects in real-time in an unambiguous manner. Therefore, in order to enable operating IoT systems with or without the humans in the loop, tight and coupled physical systems that encompass reliable and available database systems with a high level of integrated intelligence are needed.

In summary, IoT applications are diverse not only in the domain of usage, but also in terms of topology, protocols, goals, and data delivery models. Consequently, this diversity is reflected in the generated data, which in turn complicates the development of systems, where users can combine data from different applications and resources.

1.7 IoT System Components

A typical IoT system comprises three main components namely, sensor nodes, gateway devices, and back-end cloud. These components represent data sources, data communication networks, and data processing respectively.

1) **Sensor nodes:** sensor nodes are the lowest level and are composed of a set of very limited resources, sensors, and microcontrollers. Sensors can be deployed everywhere and to serve a variety of use cases, for instance, agriculture farm fields, hospitals, schools, human body, etc. Sensors are conceived as the digital backbone of IoT systems that have the capabilities to detect changes and events in the environment and report them to higher layers for further analysis. They can sense environmental parameters such as temperature, humidity, light, and the presence of chemicals. Their role is very significant to the overall IoT system performance because it is the point where data gathering tasks start; thus they are required to be very accurate Poongodi *et al.* (2020); Yu *et al.* (2017);

2) **Gateway devices:** devices at the gateway level have more computing resources compared to the devices at the sensor node level. They act as a bridge to establish communication between devices at the sensor node and the services on the cloud. This level works as a hub for collecting, aggregating, and partially processing sensory data, and forwarding them to the cloud. In addition, they might receive controlling instructions or perform some decision-making tasks on their own. The increasing demand to shift some computing tasks to the network edge has been promoted by the significant improvement gateway devices have seen recently. Typically, several data preprocessing tasks can be assigned to IoT gateway devices to eliminate redundancy and unnecessary overhead. Also, this tendency is designated to cope with some time-sensitive use cases to shorten latency and improve IoT systems responsive Yu *et al.* (2017); Al-Fuqaha *et al.* (2015);

3) **Cloud back-end:** cloud platforms gather data from a variety of deployed gateway nodes and sensors, and they provide event-based services to end-users customized as a notification service, applications, or as a graphical interface. Cloud services are featured by the

availability of plenty of resources that can satisfy the variety of requirements of IoT use cases. In addition, they are capable of handling management tasks for big data analysis, which facilitates the processing of data and the extraction of valuable knowledge Yu *et al.* (2017); Al-Fuqaha *et al.* (2015).

1.8 Edge Computing

The rapid increase in the number of devices connected to IoT networks and the amount of data generated by them has posed new requirements to meet the QoS of several IoT applications. More specifically, cloud computing based solutions have shown some deficiencies in meeting the delay-sensitive and context-aware service requirements of IoT applications. This has pushed toward the adoption of new approaches to cope with the aforementioned requirements by extending some cloud capabilities to the network edge closer to data sources. In this regard, Edge computing has been widely accepted as a key component of IoT solutions especially to handle time-sensitive applications and to reduce data volumes to be transmitted to the cloud Khan *et al.* (2019).

Edge computing as defined by Gartner (2013) in their glossary is “part of a distributed computing topology where information processing is located close to the edge where things and people produce or consume that information”. According to Gartner’s report, by 2022 75% of collected data will be processed outside of traditional cloud platforms. These figures are mainly promoted by the large-scale and distributed deployment of IoT-connected devices Boguda (2020).

The distributed nature of edge computing refers to reducing the reliance on central and remote cloud servers for carrying out the entire data processing and storage tasks. Instead, these tasks can be assigned to edge devices, which located are a short distance from data sources. This proximity is significant to efficiently run timely and context-aware services. Moreover, the edge computing paradigm contributes to the reduction of expenses related to processing data on the cloud Boguda (2020). This reduction can be gained from saving the bandwidth needed to transmit data to the cloud. And the cost of using additional resources provided by cloud services. Furthermore, a considerable portion of edge devices is equipped with acceptable computational capacity, which empowers them to handle the demands of IoT. Thus, serving several IoT application requirements that are not compatible with the delay expected using conventional cloud services. In the following section, a more detailed discussion about the distinct characteristics of edge computing in the context of IoT is provided Badidi *et al.* (2020).

1.8.1 Edge computing characteristics

Regardless of some similarities between edge and cloud computing paradigms, several characteristics that have made the former unique are as follows:

- **proximity:** unlike the cloud, edge nodes can be distributed geographically; so they can provide numerous computational resources close to end-users and devices, thus supporting large-scale real time data collection and analysis. This in turn would likely improve users' experience. In addition, local availability of resources allows users to leverage the network context information for making offloading decisions and service usage decisions Khan *et al.* (2019);

- **low latency:** generally, in IoT two main factors contribute to the latency of an application namely: computing latency and transmission latency. The former results from the time needed to process data packets, which heavily depends on the computing capacity of the system. More specifically, some IoT devices such as sensors are resource-limited, so they communicate data to a higher level device that has sufficient resources to accomplish the job. However, this data transmission forward and back between end-devices and cloud servers will likely result in a significant increase in latency. Bringing the computation resources and services to the edge closer to devices and users will considerably reduce the latency in accessing the service. For instance, conveying data analysis tasks to the edge eliminates latency which will be reflected in faster response time. This also makes the generated data more useful and actionable Yu *et al.* (2017);

- **scalability:** edge nodes enable IoT systems to be more scalable to meet the increased number of end-users and the workload. In addition, edge nodes large scale deployment is mainly promoted by their relatively reasonable cost compared to cloud data-centers Al-Fuqaha *et al.* (2015);

- **heterogeneity:** typically, IoT edge devices run under an extremely heterogeneous environment in terms of hardware components and their counterpart software modules and APIs. Existing variances result in the rise of interoperability issues, which pose the main obstacle in the successful deployment of IoT Edge-based solutions. In addition, network heterogeneity refers to a diverse communication technology stack that impacts the delivery of edge-based services Khan *et al.* (2019).

Table 1.1 Edge vs Cloud computing

Characteristic	Edge	Cloud
Deployment	Distributed	Centralized
Components	Physical edge nodes	Virtual resources
Resources	Limited	Unlimited
Response Time	Fast	Relatively slow
Cost	Relatively low	High

1.9 Semantic Web Technologies and their role in IoT application development

Semantic web technologies represent a diverse technology stack intended to facilitate deriving meaning from information. They are used to tag data with meaningful, and sharable concepts that can be understood by different software agents. The merging between IoT and Semantic Web technologies has introduced a novel framework called Semantic Web of Things (SWoT), which envisions the adoption of the Semantic Web as a key enabler of data integration in IoT applications. The creativity of the Semantic web revolves around its ability to allow software agents to seamlessly reuse, share, and combine information available in the World Wide Web. This information is frequently represented in a form of web resources and described in a way that enables deducing new information. Knowledge-based systems can benefit from the SWoT vision that extends their autonomic capabilities to empower them to perform several tasks including information storage, management, and discovery in addition to facilitating stable access to information sources Ruta *et al.* (2012).

In addition, the movement toward this trend has been stimulated by the idea of reducing as much human intervention as possible within IoT applications. In detail, due to their capabilities of annotating data so they can be machine interpretable SW technologies have the potential to enable software agents to automatically discover and interact with IoT devices based on their capabilities and facilitate the management of large-scale IoT resourcesJara *et al.* (2014). In this context, several semantic web technologies have been recently used to improve the quality of IoT applications; we discuss some of them in the following section.

1.9.1 Ontologies

Ontologies as defined by Dillon *et al.* (2012) “are formal, explicit specifications of a shared semantic conceptualization that are machine-understandable and abstract models of consensual knowledge”. The concept of ontology was originally conveyed from the philosophy field to the computer science field to facilitate the knowledge representation process by describing certain domains as a set of concepts and the relations among them. An ontology consists of four main components: classes, relations, attributes, and individuals. Classes are the main component, which can be composed of one or some subclasses, used to define more specific concepts. Classes and subclasses have attributes that represent their properties and characteristics. Individuals are instances of classes or their properties. Finally, relations in the ontology are used to connect all the aforementioned elements Munir (2016).

To facilitate interoperability and data exchange between IoT resources, recent activities to design ontologies to be used for several purposes including the description of sensor and sensor networks, IoT resources and services, smart “Things”, etc. We briefly discuss some of them below.

- **CSIRO Sensor Ontology:** this ontology was designed for describing and reasoning regarding sensors, observations, and scientific models. More specifically, CSIRO ontology describes some key concepts such as grounding (location, mobility, and range of sensing), Operation Model, and Process Munir (2016);

- **SWAMO:** an ontology was developed to support dynamic, compostable interoperability of sensor web products and/or to extend services. It describes autonomous agents for system-wide resource sharing, distributed decision-making, and autonomic operations. This ontology focuses on the sensor domain, and particularly on processes to take control over them Munir (2016);

- **SSN:** the semantic sensor network ontology was developed by W3C, and it revolves around three major concepts: systems, processes, and observations. The ontology can describe sensors, their accuracy, and capabilities, observations, and methods used for sensing. In addition, SSN ontology comprises concepts to express several sensor specifications such as operating and survival ranges, along with its performance within those ranges. Also, a structure for field deployments is included to describe the deployment lifetime and sensing purpose of the deployed macro instrument Compton *et al.* (2012). The SNN ontology has been used by several projects as a compatible component with other ontologies, so it has proved a level of interoperability;

- **O& M-OWL (SemSOS):** the O& M-OWL ontology was created to infer "high-level" concepts from "low-level" phenomena by reasoning over sensor data. By incorporating these concepts into a Sensor Observation Service (SOS), users would be able to query for events without requiring expert knowledge of how events and phenomena are related. Mostly, this ontology targets sensors combining description, and it is consisted of four main concepts: observation, process, feature, and phenomenon Munir (2016).

In its essence, ontologies are knowledge representation techniques; thus they must be augmented with mechanisms for (i) Representation and (ii) Inference. In this perspective, several Semantic-Web ontology representation formalisms have been presented - namely RDF and OWL Aggarwal *et al.* (2013).

1.10 Machine Learning For IoT

The pervasive connectivity, cost reduction, and relatively small foot-print of IoT connected objects have contributed to the development of novel systems that vary in their objectives and requirements. Data gathered from such systems are a key element in the efficiency improvement in several IoT based services that people rely on to perform their daily activities. A significant challenge for these services concerns the treatment of data; it is important to ensure efficient and reliable data analysis. This means that any misinterpretation of data may lead to erroneous decisions, which can lead to fatal consequences in some cases. ML techniques have been widely applied as a key enabler of data analysis in several computer science fields including pattern recognition, image processing, and predictive applications. They are featured by their capability in performing data classification, prediction, and clustering tasks. Recently, they have attracted both IoT academic and industrial communities to apply such intelligent algorithms to improve the outcome of IoT system functionalities. Essentially, ML techniques are intended to promote IoT system efficiency to handle data using statistical models. These models are trained using data samples that are identified by measurable characteristics called features. ML models seek to uncover the correlation between these features and the target output values called labels. The labels obtained during the training phase are exploited to support a ML models' parameters, which enable them to make inferences on new data samples and generate scores. These scores can be used to identify patterns or make decisions. The underlying principle of using ML in performing decision-making tasks is that a score can be an indicator of an event of interest that might require attention. Generally, ML algorithms are characterized based on their usage into four categories as follows:

1) **Classification:** this category of ML algorithms falls under the supervised learning style techniques, which is concerned with the learn function (f) to map between input variable (x) and output variable (y) Jagannath *et al.* (2019). Typically, in classification tasks, the data set is composed of, a features set (input) and known labels (output) or labels classes with categorical or district valuesSamie *et al.* (2019). The training step is applied to reveal the relationship between them through a scoring function. During the testing step, the scoring

function is applied to the testing dataset to generate predictions that will be compared to the actual values to assess the model performance. Classification tasks are categorized based on the number of label classes into binary and multi-class classification. The former category involves two label classes such as true/false or 0/1 as in the occupancy detection use case Samie *et al.* (2019). The latter category involves more than two label classes as in the human activity recognition use case. Several classification algorithms are widely used, including support vector machine (SVM), decision tree (DT), naive Bayes, k-nearest neighbors (KNN), artificial neural networks (ANNs), and linear discriminant analysis (LDA). Classification algorithms have been applied in different IoT based application including smart transportation Bansal *et al.* (2020), indoor localization Turgut *et al.* (2019), fall detection Al-Rakhami *et al.* (2018), waste management Dubey *et al.* (2020), as well as in IoT health care application such as early detection of dementia Ahamed *et al.* (2020);

2) **Regression:** likewise, regression algorithms also belong to the supervised learning style. Regression exhibits the relationship between input variables and output to provide a predictive model. The output of regression is likely to hold a numerical value or to be a continuous variable. For instance, a regression model can be used in a smart grid application to predict energy usage patterns based on historical observations Samie *et al.* (2019); Zantalis *et al.* (2019). Linear regression (LR) is one of the most accepted and simplest models for regression. In addition, Bayesian networks (BNs), decision trees (DTs), support vector regression (SVR), and K nearest neighborhood (KNN) are used for regression tasks. Regression models have been widely adopted in IoT based prediction solutions including demand prediction for a bike-sharing system Xu *et al.* (2020), early heart disease detection Kumar & Gandhi (2018) and real time power consumption monitoring as proposed by Arce & Macabebe (2019);

3) **Clustering:** clustering falls into the unsupervised learning style that attempts to learn useful properties of the training data rather than learning to map inputs to specific outputs Samie *et al.* (2019). More specifically, clustering is employed to extract the structure and hidden patterns from data. It partitions data points under analysis into groups that

share similar features and structure. Different clustering algorithms use different metrics to cluster data points into groups including distance to other data points, the density of the surrounding data points, or fit to a probability distribution, among others. Similarly, these metrics are used to compute the categorization process when the trained clustering algorithm is applied to data samples in order to place them into one of the existing clusters Jagannath *et al.* (2019). K-means is one of the most adopted clustering algorithms, which applies the distance metric to calculate the minimum distance between data points, and the maximum difference between clusters. In addition, several other clustering algorithms are proposed including fuzzy clustering, and density-based spatial clustering that relies on different clustering metrics in their operation. Clustering algorithms have been exploited in the context of IoT to provide efficient security mechanism for intrusion detection in some research works Mliki *et al.* (2019); Shukla (2017); de Lima Pinto *et al.* (2018); Yao *et al.* (2018);

4) **Reinforcement (RL):** mainly this category of learning algorithms is intended to promote application where decision making tasks are essential. It is based on learning a decision function that iteratively takes the previous decision output as an input, evaluates the decision and updates its parameters until the optimal output is obtained. To achieve this, reinforcement algorithms apply the reward or penalty concept Samie *et al.* (2019). One of their significant attractive features, especially for IoT, is the capability to learn with no prior knowledge about the relation between the input and actual output (e.g., decision). In the reinforcement learning style, agents learn the model from experiences and trial-and-error. In order to improve the efficiency of the decision making process, reinforcement algorithms perform actions, observe the output and adjust the state of the environment. Q-Learning is one of the most applied reinforcement algorithms due to its simplicity of use, as well as it achieves faster convergence to the optimal solution Jagannath *et al.* (2019). In addition, the combination of reinforcement learning and deep neural networks (DNN) has led to the emergent of algorithms such as deep reinforcement learning (DRL), and e deep Q-network (DQN). RL has been applied in several IoT use cases but more work has been dedicated to the resource management and tasks offloading especially on the edge devices taking

into consideration their limited resources Lu *et al.* (2020); Deng *et al.* (2020); Wang *et al.* (2020b).

Incorporating ML techniques into the IoT realm has been investigated on multiple levels of the system, varying from the lowest level represented by the sensor nodes to the top-level represented by interactive end services. This has unleashed the potential that the deployment of ML techniques may introduce to the IoT infrastructure and applications. ML can be applied to serve several purposes including network optimization, congestion avoidance, and resource allocation at the network level, as well as real-time or offline data analyzing and decision making at the application level.

1.11 Literature Review

This section investigates related work and focuses on works that are closely related to the main contributions presented in this thesis. More specifically, more concentration has been placed on the exploration of semantic and ML based approaches for IoT data processing.While some approaches are keen to investigate the applicability of either, we have highlighted some related work that applied both in a combined manner, but not on the edge.

1.11.1 Edge and Semantic Based Approaches

Semantic Web technologies have been extensively applied to ensure better quality of data processing mechanisms in the IoT domain. This is due to their capabilities in alleviating the heterogeneity issue that span over the different levels of the IoT network, including the technological heterogeneity and data format heterogeneity.

Several studies that have been reported in the literature discussed and evaluated the feasibility of SW techniques at the network edge of IoT systems. Su *et al.* (2017) carried out several evaluation experiments to assess the applicability of transferring OWL2 ontologies to the edge. They have used several comparison criteria including the syntax length and time needed to build ontologies under different syntaxes. The authors also investigated the RDF ontology format and its applicability for edge reasoning under different evaluation criteria including scalability and latency Su *et al.* (2018). In both cases, Android phones with limited resources have been used as edge nodes. Two implementation scenarios were applied. In the first (basic) the ontology is stored on the edge node, while in the second, the edge nodes fetch the ontology from a remote ontology repository. In both cases, the edge node is in charge of running reasoning tasks and sending the result to the cloud.

Recent work by Le-Tuan *et al.* (2020) has studied the impact of the RDF engines on the IoT edge resource-limited devices. They conducted an empirical study in a simulated scenario of a weather data management system. Several experiments were conducted using Raspberry pi to assess the impact of the RDF engine on its computing, resources namely CPU and RAM. These experiments have shown that the RDF engine needed to be optimized in order to be applied on IoT edge devices. Thus, the authors proposed “RDF4Led” a lightweight and optimized version oriented for IoT edge devices. It follows similar architecture as traditional RDF engines with special optimization of Physical RDF Storage, the Buffer Manager, and the Query Executor to meet the resource limitation aspect of IoT edge devices. These optimization operations have been applied to mainly minimize RAM and CPU utilization, and to achieve faster processing time.

Gyrard (2013) proposed an architecture to facilitate the semantic annotation process of heterogeneous sensory data. The architecture is decomposed to three gateway levels including: sensor gateways to capture sensory data, aggregation gateways to annotate heterogeneous sensory data with shared semantics, and semantic-based applications to reason over annotated data. The author adopts the linked data concept for reasoning. To automate the semantic annotation process, a sensor measurement ontology (SenMESO) was developed and applied. It takes the role of middleware to map heterogeneous measurements to their domain ontologies. This work emphasizes the integration of different ontologies and protocols based on semantic reasoning to map heterogeneous ontologies, which improves the interoperability among different domains.

A semantic annotation approach targeting virtualized wireless sensor networks was proposed by Khan *et al.* (2015). It is designed based on the concept of overlays and intended to function under both resource-constrained and resource-full environments. The proposed architecture is composed of two overlays namely: data annotation and ontology storing. The former receives a request from a high-level application to annotate the sensor, where each sensor involved in the process has its own annotation agent. The annotation agent downloads the required ontology from the ontology overlay. These steps are performed concurrently during the implementation of the application, which might not be suitable for resource-constrained environments. The authors proposed a base ontology that extends from the SSN ontology, in addition to a fire detection domain ontology. This architecture does not provide means for data analysis and storage.

Seydoux *et al.* (2018) have proposed an Emergent Distributed Reasoning (EDR) mechanism, for scalable semantic data processing in the Fog. EDR is designed based on the assumption of the hierarchical network topology including the cloud and the fog layers. But, rather than transmitting data to the cloud for processing, fog nodes apply semantic rules to sensory data. On the other hand, the cloud is conceived as the root node and provides an interface layer that receives rules from applications running, for instance, on smart devices, and forwards it to a fog-enabled node. The proposed approach accordingly follows an adaptable rule and data deployment strategies. Moreover, used semantic rules are expressed in SHACL (SHapes And Constraints Language). The intention is to realize data processing closer to their sources and mitigate the network bottleneck issues, as well as, reduce the response time and ensure data privacy.

Desai *et al.* (2015) introduced a conceptual design of Semantic Gateway as Service (SGS) to promote interoperability of IoT applications at the messaging protocol level. Mainly, the proposed architecture targets enabling the translation between messaging protocols such as XMPP, CoAP, and MQTT by means of a multi-protocol proxy architecture. The SGS is composed of three main components: multi-protocol proxy, semantic annotation service, and gateway service interface. In addition, the proposed SGS supports message store and topics router, which enables the multi-protocol proxy and the gateway service interface translating between messaging protocols. The SGS uses OGC standards and the SSN ontology for the annotation process. First, received raw sensory data are annotated using the OCG; then each message is tagged with sensor description using the SSN ontology. Although the authors have explained the proposed architecture in detail, it lacks a concrete proof of concept implementation.

El Kaed *et al.* (2017) proposed a Semantic Rules Engine (SRE) for edge-based industrial IoT applications. SRE is composed of two parts namely: a rules engine and a semantic engine. The semantic engine represents an abstraction layer that is intended to hide the complexity and heterogeneity of the underlying infrastructure including devices, protocols, data, and any topological changes. This is realized through leveraging device metadata, which enables the retrieval of contextual information using semantic queries. On the other hand, the rules engine is proposed to enable efficient decision-making tasks. It provides an effective way to deploy a control mechanism (as rules) on industrial gateway devices. These rules are expressed in a simple scripting language and can be modified and uploaded at run time without disrupting the operation of the gateways.

Kotis & Katasonov (2012) proposed a design of a semantic smart gateway framework (SSGF) to address the interoperability challenge at the semantic level. More specifically, this work aims at facilitating the integration of domain ontologies used by different parties. SSGF acts as a middleware that provides tools needed to apply ontology alignment, and registration of smart objects on the network. It supports thr semi-automated description of smart objects that are not pre-equipped with ontological definitions by applying an on-the-fly ontology learning mechanism. Further, to implement the proposed SSGF, existing approaches in the domain of ontology mapping (e.g. Ontology Alignment API) were applied to facilitate interoperability between smart objects. However, this approach relies on some heavyweight processes, which might not be feasible in some IoT resource-constrained environments.

Christophe (2012) proposed a distributed framework composed of geographically distributed nodes managing a pool of semantically described IoT resources. It enables scalable search and management for the IoT resources and intended to promote knowledge sharing between nodes within the geographical boundaries. Thus, the proposed framework incorporates the location as a key parameter for searching the IoT resources, where nodes publish their location. Then a managing node based on neighboring nodes creates federations, where each node can reason over received data and the produced knowledge is stored locally, avoiding a single and centralized database (or Triple Store) to be flooded by queries (and data). Also, a set of semantic web rules are defined to enable sharing knowledge discovered by one node with a subset of selected nodes.

To facilitate the semantic description, discovery, and integration of loT objects, authors in Chun *et al.* (2015) proposed an IoT directory named IoT-DS. loT-DS encompasses four main modules: loT discovery, SPARQL engine, Triple Store, loT component model. To discover heterogeneous loT entities, loT-DS allows a client to semantically search the required loT objects. For an efficient store of semantic data as RDF triples, authors employ a triple store for storage and retrieval of RDF triples. In addition to the IoT directory, authors developed an IoT registration Web portal, which is a graphical user interface, enables users to register or discover loT objects as well as visualize the locations of IoT objects on a map.

In summary, from the reviewed studies several conclusions can be drawn. First, the use of SW techniques has shown its feasibility to promote interoperability among different IoT systems and applications. Second, SW techniques can be used either to annotate contextual data related to IoT devices including location, device and meta-data. These can be required and used to facilitate service discovery by semantic-based IoT applications. Third, SW techniques can also be used to annotate sensor observations to enable smooth integration between data coming from different sources. Finally, in order to enable edge-based semantic IoT solutions, a considerable effort is needed to optimize the deployment of SW models and engines to meet resource limitations. Hence, a key issue that is highlighted in the literature is the lack of data preparation and filtering mechanism. This implies that the edge devices are likely to handle the burden of processing and annotating all the data regardless of its importance in the context of the running application, which increases resource consumption and network traffic between the cloud and the gateway.

1.11.2 Edge and ML based approaches

There is a growing tendency toward incorporating artificial intelligence algorithms into the network edge. This attraction has been pushed by the significant increase in the development of IoT applications and the diversity of its use cases. Merenda *et al.* (2020) have presented a detailed review of models, architecture, and requirements of IoT edge-based solutions to implement machine learning in their work. They carried out several compression experimentations deploying ML models on IoT edge devices with limited resources.

Murshed *et al.* (2019) discussed several machine learning and neural network models, and several IoT edge-based use cases such as healthcare and smart home monitoring. This study has shed some light on common frameworks used to deploy ML models on edge devices, and several resources-constrained hardware platforms capable to run ML models at the network edge. Other interesting studies Cui *et al.* (2018); Samie *et al.* (2019) have surveyed the importance of embedding ML into IoT applications, and in summary, they emphasize the great potential of ML as a key enabler of endowing IoT devices with the capabilities of information inference, data processing, and intelligence.

On the other hand, several big names such as Google[1], Amazon[2], Microsoft[3] and IBM[4] have thrown their attention toward providing ML services either to analysis IoT data on their cloud platforms, or enable the creation of ML models and later deploying them on IoT edge devices. However, as previously mentioned in the introduction section this raises two concerns mainly service cost and privacy.

Collaborative frameworks have been investigated widely to gain the best of both paradigms: the cloud and the edge Mora *et al.* (2018). A cloud-edge collaborative framework for IoT data analytics was proposed by Moon *et al.* (2019) . The cloud is in charge of creating ML models

[1] https://cloud.google.com/edge-tpu.

[2] https://aws.amazon.com/machine-learning/accelerate-amazon-sagemaker/.

[3] https://azure.microsoft.com/en-gb/services/iot-edge/.

[4] https://www.ibm.com/cloud/blog/models-deployed-at-the-edge.

and selecting the most appropriate one to be deployed on the edge to predict future PM10 and PM2.5 concentrations in a specific location. The selection process is done based on the correlation between the data used to train the model and the data gathered at the location.

Rahman & Rahmani (2018) investigated the advantages of using ML over ordinary rules to control IoT systems through a distributed intelligent system. Their hypothesis revolves around providing low-level intelligence to process 'small data' at the edge, while providing high-level intelligence to process 'big data' at the cloud. Belief-network was chosen to evaluate the solution in a simulated scenario. The approach presented in our work differs from both works as it offers controlling mechanisms to empower decision-making tasks by the system. The decision can be performed by the user or automatically by edge devices in case of emergency.

Taneja *et al.* (2019) proposed an approach to distribute and decompose the data analytics workload in the edge layer. They emphasize the importance of knowledge gathered from local areas to get a better view of the parameters under consideration. They adopt a tree-like topology considering the cloud as a root, edge as an intermediate node, and IoT devices as leaves. Edge nodes perform the data analysis task using a multivariate linear regression model and send the result to the cloud. They concluded that the distributed approach has considerably lowered resource consumption at the cloud and significantly reduced the amount of data sent to the cloud. We share a similar interest in this work to provide data analytic capabilities at the edge of IoT networks. Likewise, Rajith *et al.* (2018) presented a collaborative framework for real-time HVAC control systems based on an IoT cloud-edge framework. They used the edge as a data collector, aggregator, and forwarder, while they placed the prediction engine on the cloud since they use the Weka software, which relatively requires resources to analyze sensor data. Our approach provides additional support to the IoT edge-solutions by implementing some event detection mechanisms that further promote proactive decision-making functionalities.

Syafrudin *et al.* (2019) demonstrated the affordability of edge-based solution for early fault detection and warning in industrial applications. Raspberry pi was used as an edge device. It holds an inference engine running a Random Forest model supported by DBSCAN to improve its accuracy. The edge is directly sent notifications to a web interface for visualization; thus no cloud communication is required. This work fall within efforts to improve IoT based systems in performing proactively based on predictive models; however, it lacks actuation mechanisms. In addition, it only focus on the prediction of one parameter, unlike the proposed approach, which considers edge-based reasoning over multiple parameters.

Due to the diversity of IoT data sources, and its distinctive temporal and spatial characteristics, it should be considered differently from other homogeneous inputs, such as image and audio during preprocessing steps. Several initiatives in the literature attempt to propose approaches to facilitate the creation of ML models to serve different IoT use cases.

Alves *et al.* (2019) proposed a Machine Learning Framework for IoT data (ML4IoT). It mainly concentrates on facilitating the creation of ML workflows on large amounts of different IoT time-series datasets using different ML algorithms. Two types of workflows were proposed namely Batch and Online learning. The batch workflow comprises steps followed for data pre-processing, model selection, parameter configuration, training, and testing, while the online workflow considers the model evaluation using new data chosen from sources similar to the ones used for model training.

Additionally, the literature highlights several works that manifest the feasibility of integrating ML and FL techniques in the IoT. Within this context, Shukla *et al.* (2019) proposed a hybrid approach of fuzzy logic and reinforcement learning for sensory data processing. This approach is based on 3-Tire IoT architecture, where data are first gathered at the first layer and classified using a fuzzy-based inference engine. Then the systems forward the data to the second layer (fog) that runs a reinforcement learning model to select the most time-sensitive data to directly send to the end-user, while data with less importance are sent to the cloud for storage and future use.

Iram *et al.* (2019) proposed a real-time traffic controlling system using machine learning and fuzzy logic. The system was deployed on several edge devices to perform context-aware traffic signals controller based on the occupancy information recorded in a specific location. Occupancy information is gathered on a large-scale, analyzed and clustered using the K-Means algorithm. Then fuzzy logic is applied on the clusters, thus a longer duration is assigned to edges with a higher density of traffic.

A similar combination of ML and Fuzzy logic techniques are proposed in the presented work. In addition to using fuzzy logic in deducing over predicted sensor data to detect and classify events, it is also used to perform the decision making based on some kind of credibility measure induced from the ML model. Also, our approach assumes that both ML and FL run on the edge which is likely to ensure faster response time.

On the other hand, a combination of an ML and a Complex event processing (CEP) based approach is proposed by Akbar *et al.* (2017). The authors proposed an adaptive moving average approach to train ML models. A flexible prediction window mechanism is used to perform real-time predictions. This as revealed in the experiments has lowered the propagation of error-rate. They used MAPE as an accuracy measure and ordinary If rules to control the adjustment of the prediction window.

Adeleke *et al.* (2017) investigated the applicability of integrating ML models into the sensor semantic web to promote proactive IoT systems. They evaluated their proposition in the context of an indoor air quality monitoring scenario. Several ML models were assessed including MLP, BN, DT, and RF to classify the PM2.5 observations according to predefined classes and within predefined sliding windows. Then the result generated from the predictive component is forwarded to the stream reasoning framework in the form of Resource Description Framework (RDF) triples. A C-SPARQL engine equipped with queries is used to perform reasoning and to facilitate the monitoring and decision-making tasks. When a situation of interest is detected, a proper notification or an actuation instruction is initiated to avoid undesirable consequences.

Janjua *et al.* (2019) proposed a framework for IoT edge-based rare event detection. In order to avoid the labeled data, they used an unsupervised learning approach that combines BIRCH and Agglomerative Clustering techniques. While the former is intended to shape the steamed raw sensory data as microclusters, the latter is applied to form macro clusters. This is done by merging the micro-clusters resulted from the first step based on the distance between the cluster centroids. This approach considers audio data as their target use case and it was realized on Raspberry pi.

Finally, a recent work by Guillén-Navarro *et al.* (2021) proposed an edge and ML based solution to alleviate the impact of frost in stone fruit fields. The proposed system encompasses an ML based intelligent module to perform predictions, which used to activate or deactivate the anti-frost technique. For proof of concept, the authors created their own testbed consisting of several wind, temperature, and humidity sensor nodes. To improve their system predictions, first, the authors developed two outliers detection techniques based on k-nearest neighbors and k-means clustering. Although obtained results have shown the efficiency of the proposed approach, implementing the k-means clustering technique is computationally expensive for a considerable portion of IoT edge devices.

CHAPTER 2

EVENT DRIVEN AND SEMANTIC BASED APPROACH FOR DATA PROCESSING ON IOT GATEWAY DEVICES

Mahmud Alosta[1], Ahmed Bali[1], Abdelouahed Gherbi[1]

[1] Department of Software Engineering and IT, École de Technologie Supérieure
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3

Paper published in the Journal of Ambient Intelligence and Humanized Computing on May 2018.

ABSTRACT Internet of Things (IoT) applications rely on networks composed of a set of heterogeneous sensors and smart devices, which have the capability to constantly monitor the surroundings and gather data. This heterogeneity is reflected in raw data collected by such type of systems. Additionally, these data are continuously streaming; thus leading to huge volumes of heterogeneous data, which are further transferred to centralized platforms for processing. Consequently, two main challenges have arisen. First, the heterogeneity aspect of IoT data makes high-level IoT applications' task of interpreting such data and detecting events in the real world more complex. Second, sending sensory data to a centralized platform leads to some issues, such as extensive consumption of IoT devices' limited resources, network traffic overloading, and latency, which might negatively impact the response time especially in systems that were designed to handle critical situations. In this paper, we propose a decentralized approach for IoT data processing, by delegating this task to distributed edge devices (Gateways) taking into consideration their limited resources and network bandwidth. To accomplish this, we proposed a two-layer data processing approach that employs a hybrid model encompassed of Complex event processing (CEP) and Semantic web (SW) techniques. While the first is proposed for performing aggregation and classification tasks, we use the latter for performing semantic filtering and annotation tasks. We have evaluated the feasibility of our approach to process sensory data in the context of Air Quality Monitoring scenario using an experimentation involving established ontologies. Several benchmarks are considered such as overall run-time, data size, and response time.

Keywords: Semantic Web (SW), Complex Event Processing(CEP), Internet of Things (IoT), Ontology, Data Heterogeneity, Data Annotation

2.1 Introduction

IoT is a novel paradigm that is foreseen to change the landscape of Information and Communication Technology (ICT). This conversion is promoted by the unprecedented proliferation of Internet-connected devices in our contemporary life.A recent study by Cisco anticipates that by 2020, 50 billion devices will be connected to the Internet Evans (2011). These devices are expected to be deployed in different fields of applications to continually observe the environment. This deployment is primarily targeting collecting situational data, which can be further used to develop applications that humans can use to monitor events in the real world and react accordingly. Thus, that would generate a massive amount of heterogeneous data, which further can be transferred through gateway devices to the Internet cloud platforms. These platforms can further process and analyze the collected data and harness them to develop innovative IoT applications that can potentially contribute in improving several areas of our daily lives such as health care, transport logistics, and traffic monitoring Evans (2011); Manyika (2015).

Typically, IoT systems are composed of a varied set of sensors and devices that monitor different aspects of the environment and collect data. This characteristic of IoT system components' heterogeneity is reflected in the generated data, which in turn hinders the task of IoT application to interpret this data and efficiently utilize it. Moreover, data integration process is hardly maintained in such environment, which leads to the lack of interoperability among different IoT systems, thus, limiting the development of applications that can benefit from data generated from diverse domains of IoT Ding *et al.* (2011); Borgia (2014). In a nutshell, deducing knowledge from gathered raw sensory data is a prerequisite to developing interoperable IoT applications that can detect events in the real world and respond accordingly.

In addition to the heterogeneity aspect of IoT data, these data are continuously streaming. Consequently, huge amounts of data are regularly generated and sent to cloud platforms for further processing and analyzing. Although these platforms have the capabilities to manipulate such amount of data, this process consumes a considerable amount of resources, where in some cases users would have to expand their storage facility to meet the increasing amount of data. Furthermore, blindly sending data back and forth between the cloud and IoT gateways has led to network traffic overloading and latency issues that might influence time-sensitive services Borgia (2014).Also, applying semantic annotation algorithms on such amounts of data at the gateway device level will result in extensively consuming its resources, which is likely to impact the performance.

Semantic Web (SW) technologies have been extensively utilized to interpret and integrate data coming from a diversity of resources on the Web. Recently, they have been extended to the IoT domain to enhance the quality of data and to promote interoperability Barnaghi *et al.* (2012). This is achieved by modeling IoT data based on shared vocabularies that can be interpreted by different software agents. This process is called semantic annotation, which implicates employing several SW standards such as OWL, RDFs, and RDF to build conceptual models (i.e., Ontology) that describe application domain concepts and the relationships that exist between them. Moreover, SW technology stack provides SPARQL protocol and RDF query language, which can be employed to query and reason over RDF databases to infer new knowledge from them Aggarwal *et al.* (2013).

To alleviate the burden of transmitting data to the cloud through IoT networks, and to minimize processing cost at the cloud level, a recent trend of employing IoT gateway devices, such as a hub for data aggregation and processing has been widely adopted. Recently, these devices have seen a significant improvement in terms of computing resources, which empower them to implement filtering and annotation algorithms. Moreover, these devices could achieve faster response times since they work closely with sensor nodes. This way can be helpful in reducing the amount of data transferred to the cloud, which lightens the network traffic and lowers latency.

In this work, we propose an IoT data processing approach targeting its implementation at the gateway devices. The proposed approach is based on the composition of the semantic web and event-processing techniques. It is designed to enable the platform independent implementation, where regardless of the gateway and the connected sensors specifications, developers can customize their event and semantic rules required for data processing. Moreover, we take into consideration the minimization of data to be processed at the gateway and further to be sent to the cloud. By doing this, we are targeting to accomplish two objectives. First, reducing, as much as possible, resources required for data processing on IoT gateway devices. Second, reducing the amount of data to be transferred through IoT networks to the cloud. To do this, our approach employs two levels of processing to manipulate sensory data. The first level is intended to identify this data as events and to classify them based on predefined event classification rules defined by developers according to the targeted application. Also, this step is intended to perform some essential aggregation and filtering tasks, and most importantly, to specify the priority of events transformation to the following steps for processing based on their vitality in the context of the application, for instance, If the detected event is classified as an alert event it would be given a higher priority than a normal event . While at the second level, data events generated from the first level are transformed, and two types of processing are applied. The first type is targeting filtering incoming data using a set of semantic rules; basically, to filter out redundant and insignificant data. While at the second type of processing, the semantic annotation process takes place at this level of processing. It aims at tagging the raw sensory data by a shared set of vocabularies; thus, it can be smoothly consumed by different high-level IoT applications and services.

The remainder of this paper is organized as follows: Our motivating scenario is described in Section 2.2, followed by a description of the background research and related work in Section 2.3. In Sections 2.4, 2.5, and 2.6 a detailed description of the overall architecture and the functional components of the proposed approach are presented. A prototype implementation and evaluation are presented in Section 2.7 while Section 2.8 concludes the paper and highlights some points as future perspectives.

2.2 Motivating Scenario

Consider an indoor air quality monitoring applications, where a user is interested in acquiring data about the air measurements in a specific property (i.e., Home or office). In our scenario, we undertake home as the observed property, where several sensor nodes are installed in different locations (rooms, living room, and the kitchen). Each node is encompassed of temperature, humidity, Carbon monoxide (CO), and Carbon dioxide (CO2) sensors. These sensors are constantly observing their surroundings, gathering data, and communicating them to the home gateway. These data are further transferred to the cloud for processing and storing. Apart from using them in an indoor air quality monitoring application, these data might be of interest for other types of applications and services. For instance, a healthcare application could use humidity sensor data to issue warnings about the risk of getting infected by some viruses as a result of low levels of humidity. In this case, the application should be able to interpret sensor humidity readings and integrate them with data coming from external resources, such as the web. Another application could be interested in calculating the average temperature in a building, where several IoT gateways connected to sensor nodes and expose data to the Cloud. A major issue would be integrating data from different IoT gateways, where in some cases temperature might be defined as "Temp," while in another gateway it might be defined as ’T.’

Both aforementioned use cases emphasizes on the need to define data collected by different IoT gateways and sensor nodes using a unified form that can be interpreted and shared by various IoT domain applications to gain the utmost benefits from the deployed underlying IoT networks. In this regard, semantic annotation has become a prerequisite for developing interoperable IoT applications that can take full advantage of such infrastructure. Semantic annotation refers to the process of defining raw sensory data using a set of concepts that are either defined by domain ontology or standard ontologies. By doing this, high-level IoT applications and services will be able to consume sensory data regardless of their sources or format. Also, it will be possible to integrate them with other sources of semantic data on the web.

However, the annotation process relies on a heavyweight stack of semantic technologies; it is therefore likely to consume a considerable amount of computing resources. Even though cloud platforms are featured by their substantial and extensible computing resources, more resources lead to higher processing cost. Moreover, sensor nodes and gateways constantly send data to the cloud through IoT networks; this is likely to lead to network bottlenecks, which negatively impact the system reaction especially in urgent situations due to the latency issue. Let's take a sudden CO gas leak as an example from our scenario. CO is an odorless, tasteless and highly toxic gas Goldstein (2008). Thus, it is of extreme significance for systems to react on time in such situations, where seconds or even a fraction of a second delay might lead to hazardous consequences. For more details on this issue Sheltami *et al.* (2016) surveys recent trends of using WSN for gas monitoring and detection.

To handle such aforementioned situation, a recent trend is to process raw sensory data on IoT gateways devices. These devices have been endowed with improved computing resources that enable them to some extent to carry out data processing tasks; also, several benefits of this paradigm were mentioned in Section 2.1. In our motivating scenario, since we propose to semantically annotate sensor data, to performing this task at the gateway level, we need to take into consideration two significant factors namely gateway limited computing resources and the network bandwidth. While the former factor refers to the potential impact of the annotation process in terms of required time and resources, the latter refers to the increased network traffic since the annotated file will have a bigger size than an ordinary raw sensory data file. Correlating this to our scenario, and considering the situation stated in the previous paragraph as an example, the annotation process at the gateway level is likely to be more time consuming since all sensory data flows are supposed to be treated through this step. Consequently, time-sensitive applications might be critically impacted and inadequately supported by such workflows.

Hence, based on the discussion and the scenario described above, we set the following requirement that the proposed solution should maintain:

- **RQ1:** on the fly processing of raw sensory data. It implies data should be manipulated before sending it to the cloud;

- **RQ2:** raw sensory data should be sending to the cloud on the form of semantic data; so it can be smoothly integrated with other data sources on the web or from other IoT networks;

- **RQ3:** the semantic annotation process should be performed using standard ontologies, so it can be shared between IoT based semantic applications and services;

- **RQ4:** a filtering mechanism should be employed prior to the annotation process step. It targets the minimization of data to be processed at the annotation step and further sent to the cloud;

- **RQ5:** handling critical and warning situations on time by giving the priority of processing to the sampled data that is likely to have higher impact on the situation of the observed property.

2.3 Background and Related Work

In this section, we introduce the main two technologies used in our proposed solution namely SW and CEP. The related work is also presented in this section,wrt, to our proposed approach.

2.3.1 Semantic Web

Due to their capabilities of annotating data, so that they can be machine interpretable, several semantic web technologies have been recently adopted to promote data integration and interoperability in the IoT domain.

RDF is a standard language for representing information about Web resources as XML format. It provides a unified framework for exchanging information between applications without loss of meaning. Data in RDF are stored in the form of triples; each triple is consisted of (subject, property, and object). In typical IoT application consisting of a set of devices generating a set of data, devices are semantically represented by the subject; while the property represents the measured quantity, and the object represents the measured value. RDF Schema (RDFs) is an extension of RDF vocabulary, which enables more detailed description taxonomies of classes and properties. In other words, RDFs can be perceived as an expressive meta-model used to describe the vocabulary used in an RDF document.OWL represents a more expressive way to model data on the semantic web. It was essentially developed to overcome some RDF and RDFs limitations such as the lack of a clear way for domain or range constraints' description, and the lack of ability to represent closure, inverse or transitive properties Aggarwal *et al.* (2013). RDF, RDFs, and OWL can be perceived as meta-meta models that encompass the set of vocabularies used to define new domain specific schemas and ontologies.

Ontologies, as defined by Dillon *et al.* (2012), "are formal, explicit specifications of a shared semantic conceptualization that are machine-understandable." The ontologies promote sharing of a unified understanding of domain-specific structured information among software agents, thus, enabling systems to consume data based on predefined concepts and relations in the ontology. Additionally, ontologies can be used as a way of defining rules and constraints over the data, which can be used to implement automated inference and reasoning Keskisärkkä (2017); Bali *et al.* (2017).

To facilitate interoperability and data exchange between IoT resources, recent activities took place to design ontologies to be used for several purposes, including the description of sensor and sensor networks, IOT resources, and services, smart Things. SSN, The semantic sensor network ontology, was developed by W3C, and it revolves around three major concepts: systems, processes, and observations. The ontology can describe sensors, their accuracy, capabilities, observations, and methods used for sensing. Additionally, other sensor specifications, such as measurement range and sensitivity are included in the SSN ontology Compton *et al.* (2012). SNN ontology has been used by several projects as a compatible component with other ontologies, so it has proved a level of interoperability. The SPITFIRE Ontology (spt) is based on the alignment among Dolce+DnS Ultralite(dul), the SSN ontology and the Event Model-F ontology (event). It is used to describe sensors, observations, and related concepts. For more information about the current ontologies in the IoT domain and the adoption of SW techniques, we refer the reader to Szilagyi & Wira (2016) and Hachem *et al.* (2011).

2.3.2 Complex Event Processing

Complex event processing (CEP) is a recent trend that has been widely accepted in the domain of IoT. This is due to its ability to detect events in real use cases by correlating information coming from different resources to form a complex event to satisfy a user request that can't be achieved by using single information source. For instance, in our motivating example, Section 2, CEP could be used for early detection of a fire event by creating a pattern to identify this event by correlating data coming from temperature, humidity, and CO sensors. Also, CEP could be used to perform classification and aggregation tasks. For instance, using CEP, it is possible to calculate the average temperature reading as long as it is within a predefined range. CEP concepts are implemented in the form of software engines. Their logic is defined in the form of queries or rules, and it is modeled using high-level declarative language that is likely to support temporal constructs Giordani & Archetti (2016). Furthermore, CEP engines can be distributed in the network, where it can be employed to perform some tasks that are likely to contribute to balancing the workload of processing incoming data flows. This leads to reducing the amount of traffic between the cloud and the IoT networks Chen *et al.* (2014).

2.3.3 Related Work

The adoption of Semantic Web technologies as a key component to improve the quality of IoT applications has gained the interest of many researches recently.

To address the interoperability challenge, Kotis & Katasonov (2012), designed a semantic smart gateway framework (SSGF). SSGF acts as a middleware that provides the tools to implement ontology alignment, and registration of smart objects on the network. To implement the proposed SSGF, existing approaches in the domain of ontology mapping (e.g., Ontology Alignment API) were employed targeting to promote interoperability between IoT devices. However, this approach relies on some heavyweight processes, which might not exist in some IoT environments. Also, there is no concrete implementation of this approach.

In Gyrard (2013), the author proposed an architecture to annotate heterogeneous data captured by sensor nodes with semantics by using aggregation gateways and reasoning on them using semantic-based applications. Aggregation gateways convert sensed data into semantic measurements using semantic web technologies (RDF, RDFS, OWL and domain ontologies). Afterward, semantic-based applications link this data to Linked Open Data to perform reasoning. A sensor measurement ontology (SenMESO) was designed to automatically convert heterogeneous sensor measurement to semantic data. This work emphasizes on the integration of different ontologies and protocols based on semantic reasoning to map heterogeneous ontologies.

The authors in Negash *et al.* (2016) present LISA2.0 protocol, is a lightweight in network IoT service bus architecture targeting achieving interoperability between different IoT systems. LISA provides two types of messages namely setup message and user message to support some tasks such as discover and registration. LISA employs an ontology to route service requests across protocol boundaries. However, it does not support semantic interoperability at the data level, and it lacks data filtering mechanisms to support node-centric processing.

To enable scalable search and management for the IoT resources, Christophe (2012) proposed a distributed framework composed of geographically distributed nodes managing a pool of semantically described IoT resources. The design of this framework relies on the location as a key parameter when searching the IoT resources, where nodes publish their location, and then a managing node based on neighboring nodes creates federations. By doing this, the author aims at managing data produced by nodes locally, where each node can make reasoning over received data and the produced knowledge stored locally.

To facilitate the semantic description, discovery, and integration of loT objects, authors in Chun *et al.* (2015) propose an IoT directory named IoT-DS. loT-DS encompasses four main modules: loT discovery, SPARQL engine, triple store, loT component model. To discover heterogeneous loT entities, loT-DS allows a client to semantically search the required loT objects. For an efficient store of semantic data as RDF triples, authors employ a triple store for storage and retrieval of RDF triples. This work focuses on the semantic description and integration of devices, not on generated data.

In Khan *et al.* (2015), the authors propose an annotation architecture mainly targeting virtualized wireless sensor networks. This approach was designed based on the concept of overlays, where it consists of two overlays namely: Data annotation and ontology storing. The former receives a request from a high-level application to annotate sensor data, where each sensor involved in the process has its annotation agent. The annotation agent downloads the required ontology from the ontology overlay. These steps are performed concurrently during the implementation of the application, which might not be suitable for resource-constrained environments and likely to increase network traffic.

A gateway and semantic web enabled IoT architecture was proposed in Desai *et al.* (2015), to achieve interoperability on the level of messaging protocols such as MQTT and CoAP. It comprises a semantic-based multi-protocol module to translate different messages in a unified form, by which, it promotes the generation of meaningful information. This work is mostly concentrated on the interoperability between protocols, not on data. Also, it lacks a concrete implementation.

Overall, the reviewed studies have some limitations, such as consecrating on the interoperability between devices rather than data, and the use of SW to facilitate sensor discovery. Another pivotal issue that is ignored is the lack of data preparation and filtering mechanism, which implies that the gateway has to process and annotate all the data regardless of its importance to the context of an application, thus increasing resources consumption and network traffic between the cloud and the gateway.

2.4 Overall Architecture

As shown in figure 2.1, IoT system components are mainly categorized into three main elements: Sensor node, Gateway, and cloud platforms. Typically, sensor nodes are the lowest level and are composed of a set of very limited resources, sensors and microcontrollers, their main task being only to sense the environment, and collect data and send them to Gateways. Devices at the gateway level have more computing resources compared to the sensor at the node level. As such, this level works as a hub for collecting, aggregating and processing sensory data, as well as, bridging connection between sink nodes and IoT cloud services. These platforms gather data from a variety of deployed gateway nodes and provide event-based services to end users customized as notification service, applications, or graphical interface.

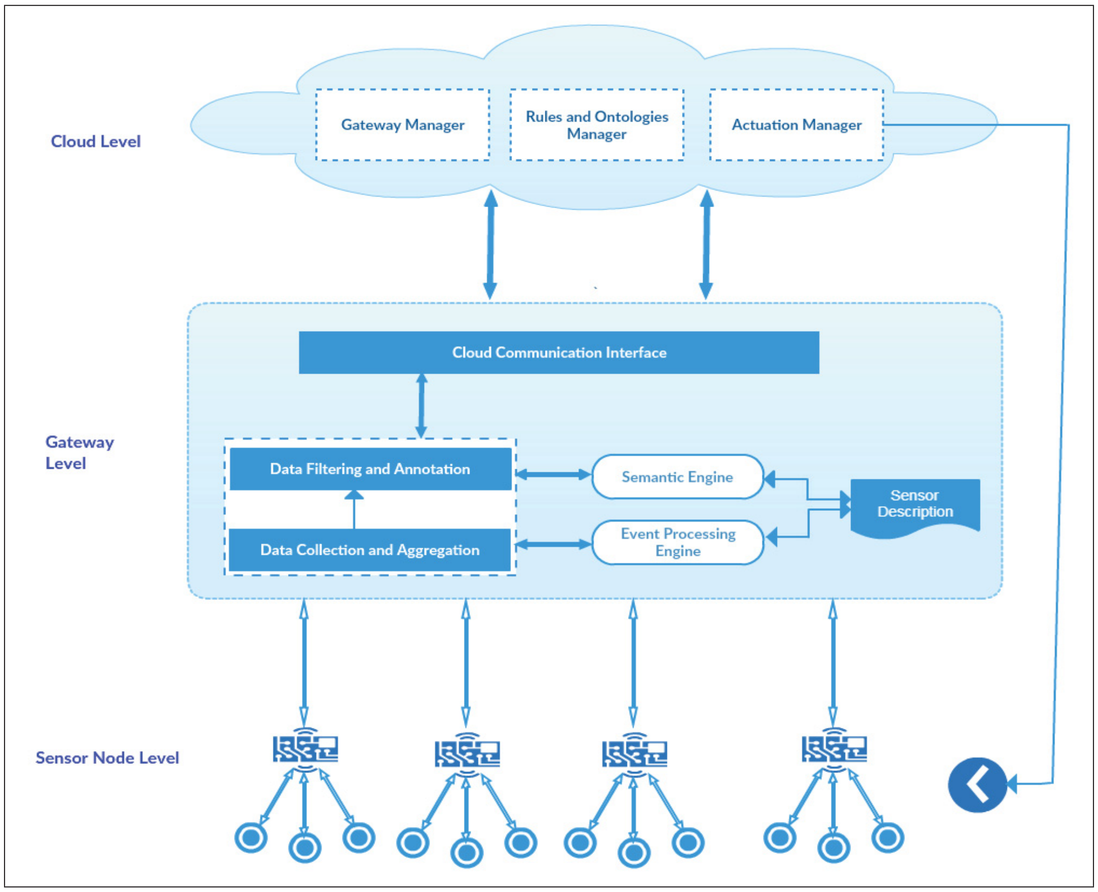


Figure 2.1 The Overall Architecture

In this work, we propose to delegate a considerable portion of the data processing task to the gateway devices. Three levels of data processing will be carried out consecutively at the edge level. First, data aggregation and classification, which has twofold purposes namely: Performing some basic filtering task based on the measured value, and classifying the data as a set of events to specify which one of them will have the processing priority at the following steps. At this step, we seek to support decision support applications by providing them with data that is of utmost importance for their context; also, we seek to slightly alleviate the task of data processing at the following components. Second, data filtering: At this level, firstly, a partial annotation is

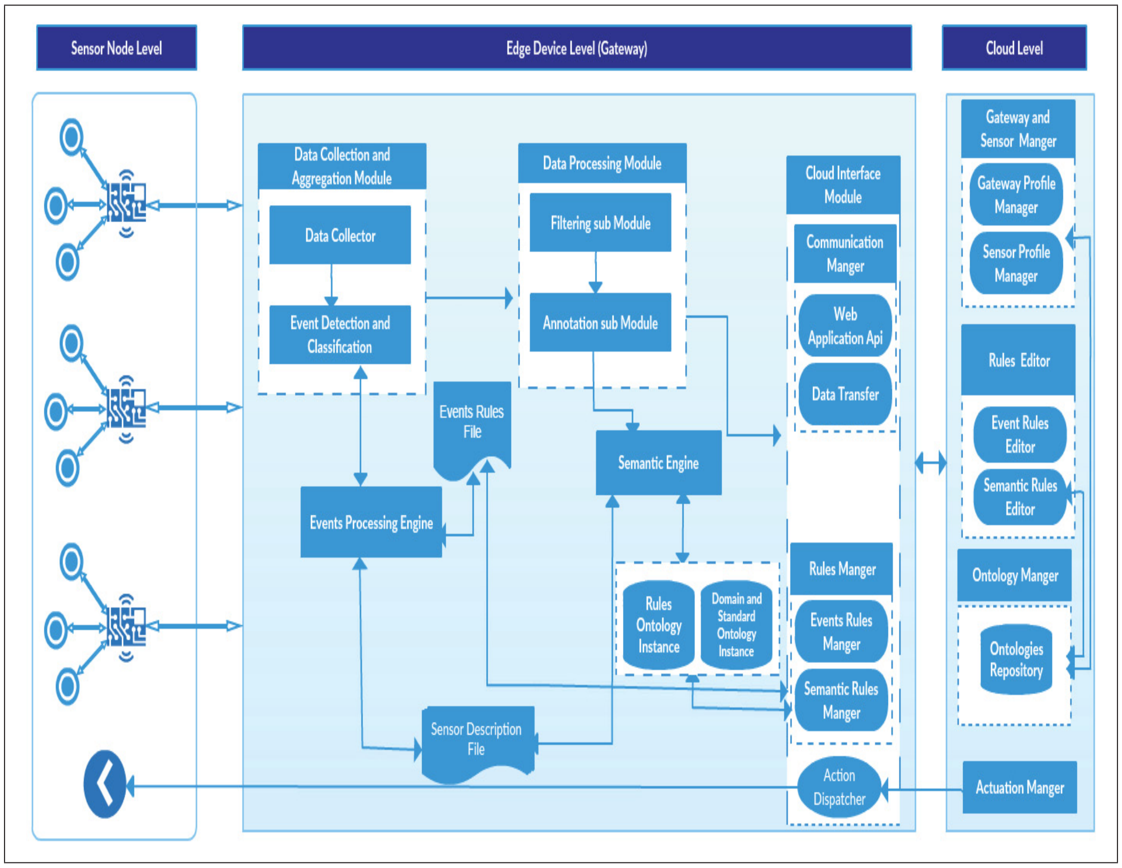


Figure 2.2 The Functional components of the Proposed Approach

performed on the data coming from the previous level. Thus, it becomes feasible to apply some semantic web-based filtering rules to remove redundant and unnecessary data. By doing so, our approach contributes to reduce the amount of data to be processed at the annotation step, thus, saving more computing resources, also reducing the amount of data to be later sent to the cloud, saving bandwidth and lightening the network traffic. Finally, data annotation level which aims at tagging incoming data by a set of shared vocabularies, so it can be easily consumed by high-level IoT applications and services. As illustrated in figure 2.1, the proposed approach is composed of three levels organized from down to top as follows: Sensor level, Edge device level, and cloud level. The following subsections describe the functional components of the proposed approach in more details as shown in figure 2.2.

2.5 The Functional Components at the Cloud Level

This level is proposed to provide flexibility to developers in modifying rules and queries used at the gateway level, by which promoting the scalability and independence of the system. This level is composed of two modules namely, Gateway and sensors manager, and rules and ontology manager.

2.5.1 Gateway and sensors manager

This module is responsible for registering the gateway at the cloud and performing the required modification on a sensor description file deployed on a given gateway. Thus, it enables our approach to be scalable in terms of the number of the connected gateways and sensors, which might be involved in a specific application or scenario. As depicted in figure 2.2, this module is composed of two sub-modules:

2.5.1.1 Gateway Profile Manager

Each Gateway that sends data to the cloud platform are required to be registered within this module in advance. This module formulates a document that contains description details intended to facilitate the identification of data sources, such as GatewayID, and location. The GatewayID is being referred to in the sensor description file for two purposes:

- to accelerate gateway and sensors accesses and detection of events location;
- to enable rules manager to identify event and semantic rules that should be deployed on a certain gateway, based on the types of connected sensors.

Table 2.1 provides a detailed description of the Gateway profile document.

Table 2.1 Gateway Profile Description

Field	Description
GatewayID	Each gateway has a unique identification.
Location	Longitude, Latitude
Gateway Type	For instance: Microprocessor or router.

2.5.1.2 Sensor Profile Manager (SPM)

This sub-module is intended to provide description details about sensors connected to a certain Gateway. It plays a significant role in our approach since it will be used by both Events processing and Semantic engine units. SPM is linked to the ontology manager to formulate and update sensor description file which will be deployed on the gateway. It can be conceived as an RDF file composed of a set of concepts to describe the sensor's descriptive properties. These concepts are imported by the ontology manager that handles the process of adding new terminologies, if needed, from different ontologies located in the cloud. In our approach, we mainly rely on the SSN ontology for sensor's capabilities and observations description. SSN provides unified vocabularies to facilitate sensors description in terms of capabilities, measurement processes, observations, and deployments. While domain concepts such as time, location, etc can be imported from other ontologies such as DUL ontology. Furthermore, the elasticity of our approach enables developers to employ other ontologies to fulfill the description requirements of any new class of sensors that they might need to add to their application. The ontology manager allows the initiation of a query to construct the sensor description document by mapping between the real sensor specifications and their corresponding concepts imported from different ontologies. The sensorId and type are two major descriptive concepts. They will be used to define the event type at the event processing step, and to define what rules should be applied at the filtering level. Also, this file contains a more detailed description of the location, Where, by using the GatewayID, it is possible to define, for instance, the location of a building where the gateway is deployed. Additionally, SensorLocation is used to locate at which floor and room the sensor is installed taking into consideration our motivating scenario.

A detailed description of the sensor description file is presented in Table 2.2.

Table 2.2 Sensor Description

Field	Description
SensorID	Each sensor has a unique identification.
GatewayID	To enable the detection of sensor location.
SensorLocation	FloorID ,RoomID
Sensor Type	To aid the process of rules and queries selection.
Model	Each sensor has a model for more specification description
Unite of measurement	Unite used to identify the measured value.
Measurement range	To identify weather a measured value is within the acceptable range to be processed or not.

2.5.2 Rules Editor

This module has been proposed to enable developers to configure their event and semantic rules that are required for data processing at the gateway level. This makes our approach versatile and gives developer the flexibility to set their own rules independently regardless of the used gateway and connected sensors. This module encompasses two sub-modules namely:

2.5.2.1 Event Rules editor

It is in charge of handling the modifications required to be applied to the rules used for event detection, aggregation, and classification. This component encompasses a file that contains a set of rules. These rules can be formatted as queries using Event Processing Language (EPL) Purich (2011). Each query has a unique name in addition to the text that describes the query itself. The query name is used to abstract the function of the query during the execution. This file will be sent to the gateway at runtime. It will be received by the Event Rule Manager component, which handles rules update process and redeploys the file on the gateway.

2.5.2.2 Semantic Rules Editor

This sub-module has been proposed to enable developers to define their semantic rules for data filtering during the execution of the annotation process. Here, we adopt our previous work for rule definition based on an ontology of rules Bali *et al.* (2017). But, unlike our previous approach, where rules and concepts filtering process is performed at the gateway level, we propose to assign this task to the cloud level and send to the gateway only rules and concepts relative to the gateway and connected sensors types. Thus, an instance of rules and concepts are generated at this stage based on the correlation between the Gatewayid and sensor description, where each gateway will have a set of generated rules deployed on it based on the type of sensors connected to it. By doing so, we avoid sending the rule set to the gateway and save resources required to perform the rule filtering task. Rules at this level are semantic, and they follow "If Then" syntactic.

2.5.3 Ontology Manager

This module has been designed to provide developers with the independence to either develop their domain ontologies or to extend other standard ontologies to their applications. Also, the ontology of the semantic rules is stored at this level. Users can leverage this module to select concepts from different ontologies to be deployed on the gateway level to perform both the filtering and the annotation processes. Thus, we seek to facilitate the integration between heterogeneous data gathered from different sources, as well as, to enable the integration between raw sensory data and data from other sources, such as the web. This, in turn, will enhance the level of interoperability between different IoT applications; thus, enabling the development of cross-domain IoT applications.

2.5.4 Actuation Manager

This component is intended to enable handling different situations by executing set of actions as a response to events taking place at an observed property. It can be configured to automatically react in situations like the gas leak case mentioned above, or semi automatically, by involving the user, in situations such as controlling the AC system upon certain temperature values.

2.6 The Functional Components at the Gateway Level.

In our approach as depicted in figure 2.1, the main role of the Gateway is collecting data from sensors and transferring them to the cloud after processing (filtering and annotation) for reducing its size and facilitating its interpretation by end applications. The gateway level consists of three modules namely, data collection and aggregation module, data filtering and annotation module, and cloud interface module.

These modules are shown in figure 2.2 for more details. The Data Collection and Aggregation module collects the raw data from the sensors, performs some aggregation and classification processes, then, it passes them to the Data Filtering and Annotation module, which in turn filters and annotates them, then transfers them to the cloud via the Cloud Interface module. In the following sub-sections, we explain these modules and their sub-modules in more details.

2.6.1 Data Collection and Aggregation Module

In our approach, we assume that data are sent to the gateway as a set of events coming from events generators (sensors).The main objective of this module is to collect data sent by sensor nodes and formulate them as a set of events. Each event must be composed of SensorID, Measured value, and Timestamp. This module is composed of two sub-modules, as shown in figure 2.2.

2.6.1.1 Data Collector

This sub-module receives data-streams from sensor nodes and stores them in a temporary file for analysis. Raw sensory data is received by this module as a set of pairs composed of SensorIDs and the measured values. Then, a tagging function stamps these data packets by timestamp to form events. These events are gathered in a single data-queue and further transferred to the following sub-module for event detection, aggregation, and classification.

2.6.1.2 Event Detection and Classification

This sub-module is an essential component of our approach. As shown in figure2.3, the main objective of this sub-module is to analyze coming events for detecting the most critical ones and assigning them the processing priority at the following steps. By doing so, we contribute to addressing RQ5. Also, it performs some basic filtering and aggregation tasks (such as "SUM, MIN, MAX, AVG"). Thus, the volume of sampled data can be relatively reduced. This, in turn, leads to saving resources required for semantic filtering and annotation. This, to some extent, contributes to addressing RQ4. The following steps describe the workflow of this sub-module:

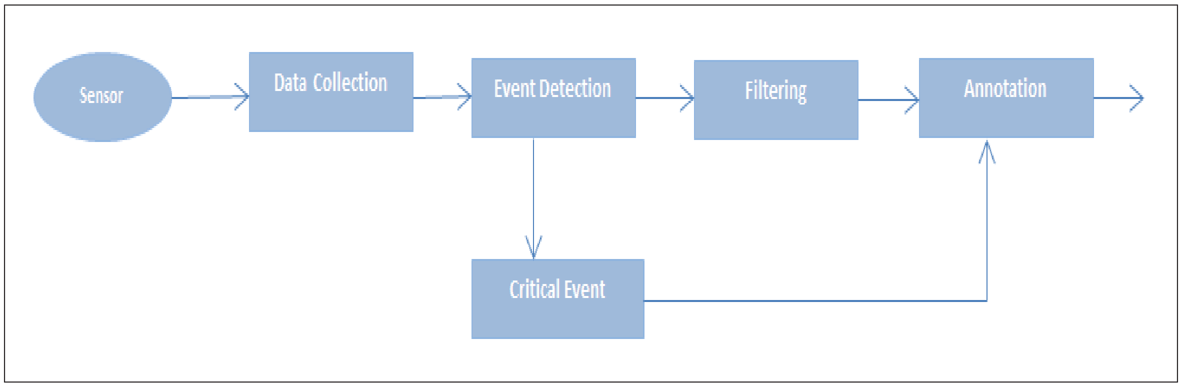


Figure 2.3 Event Processing Work flow

The first step is correlating raw sensory data packets that are composed of (SensorID, Measured value, and Timestamp) to the sensor type, and location stored in the sensor description file deployed on the gateway by the developer as mentioned previously. The intention behind this step is facilitating the detection of event type and location; thus, the system can specify what rules should be applied to data that matches specific event type.

Based on their types, events, as shown in figure 2.4, will be assigned to event adapters that are extended from Base event class. Then, an event handler will be called to perform the aggregation and classification tasks by applying a set of pre-configured rules that are stored in the event rules file. The objective of this step is to classify events based on the application requirements, thus, identifying the priority of transferring incoming events to the following steps for process.Based on the scenario described in Section 2.2, three types of event classifications are proposed in this approach namely: Normal, Warning, and Critical. Each one of them represents a certain action corresponding to the occurrence of a certain event.

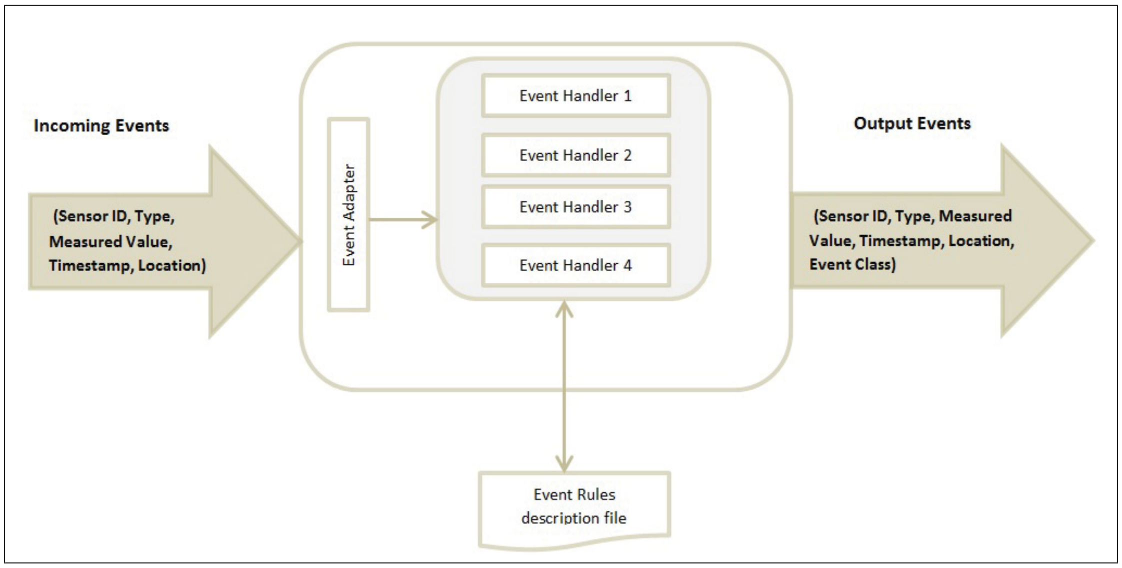

Incoming Events
(Sensor ID, Type, Measured Value, Timestamp, Location)
Event Adapter
Event Handler 1
Event Handler 2
Event Handler 3
Event Handler 4
Output Events
(Sensor ID, Type, Measured Value, Timestamp, Location, Event Class)
Event Rules description file

Figure 2.4   Event Processing Engine Components

- **normal event class:** used to report sensor readings that fill into the acceptable range and do not exceed a predefined threshold. In this case, the user has the flexibility to either proceed all the readings to the following steps or to only report the average value during a certain interval of time and transfer it to the following processing steps. This decision is mostly influenced by the application type and user requirements. Typically, this class of event representation is generated for regular monitoring and information purposes;

- **warning event class:** this type of event class is generated when a measured value is slightly above a certain threshold, and rises with time. Users can configure their queries to detect this pattern of events either by examining two or more consecutive values sent by the same source and exceed a predefined threshold. There are two probabilities in this case. First, if the reported values kept the same for a period, the system will aggregate these events, calculate the average, send it to the annotation step and issue a warning notification. Second, if the measured consecutive values match warnings class conditions with a slight constant increase in value, which implies that the status of a particular monitored object could become critical if the value keeps increasing. In this case, the measured values will directly be sent to the annotation process, without the need to pass the aggregation and filtering steps;

- **alert event class:** this type of event patterns is detected with 4 or more consecutive events, with the first one above a certain threshold significantly, each subsequent one greater than the last – and the last one being 'x' times greater than the first. In this case, an alert message will be triggered and the event will be given a maximum priority of processing and will be directly communicated to the annotation step.

Table 2.3 lists some examples of rules used in this process based on the sensors used in the proposed scenario. Event detection and classification rules can be employed to classify events

Table 2.3 Examples of Event Classification Rules

Sensor	Normal	Warning	Alert
Humidity	(Value>=30& <=65)	(Value>20&<30) OR (Value >65&<75)	(Value>0&<=20) OR (Value>=75&<=100)
Temperature	(Value>10&<=30)	(Value>30&Value<=40)	(Value<10 OR >40)
Gas (CO2)	(Value>=350&<=1000)	(Value>1000&<=2000)	(Value>2000)
Gas(CO)	(Value>=0&<=30)	(Value>=30&<=400)	(Value>400)

to several classes based on the requirements of the targeted application. For other types of unexpected events, our approach enables the developer to set rules to classify these events as an anomaly events, which are events that do not match any conditions included within the applied rules. Furthermore, we plan to investigate machine learning (ML) algorithms to develop prediction models for the detection of events that are not feasible using ordinary rules.

2.6.2 Data Filtering and Annotation Module

This module receives data packets as a set of classified events, and it performs the semantic-based filtering and annotation tasks. It plays a pivotal role in our approach since it considerably contributes in reducing the amount of data to be sent to the cloud, as well as converting raw sensory data to a semantic data that can be consumed by several IoT applications.

2.6.2.1 Data Filtering

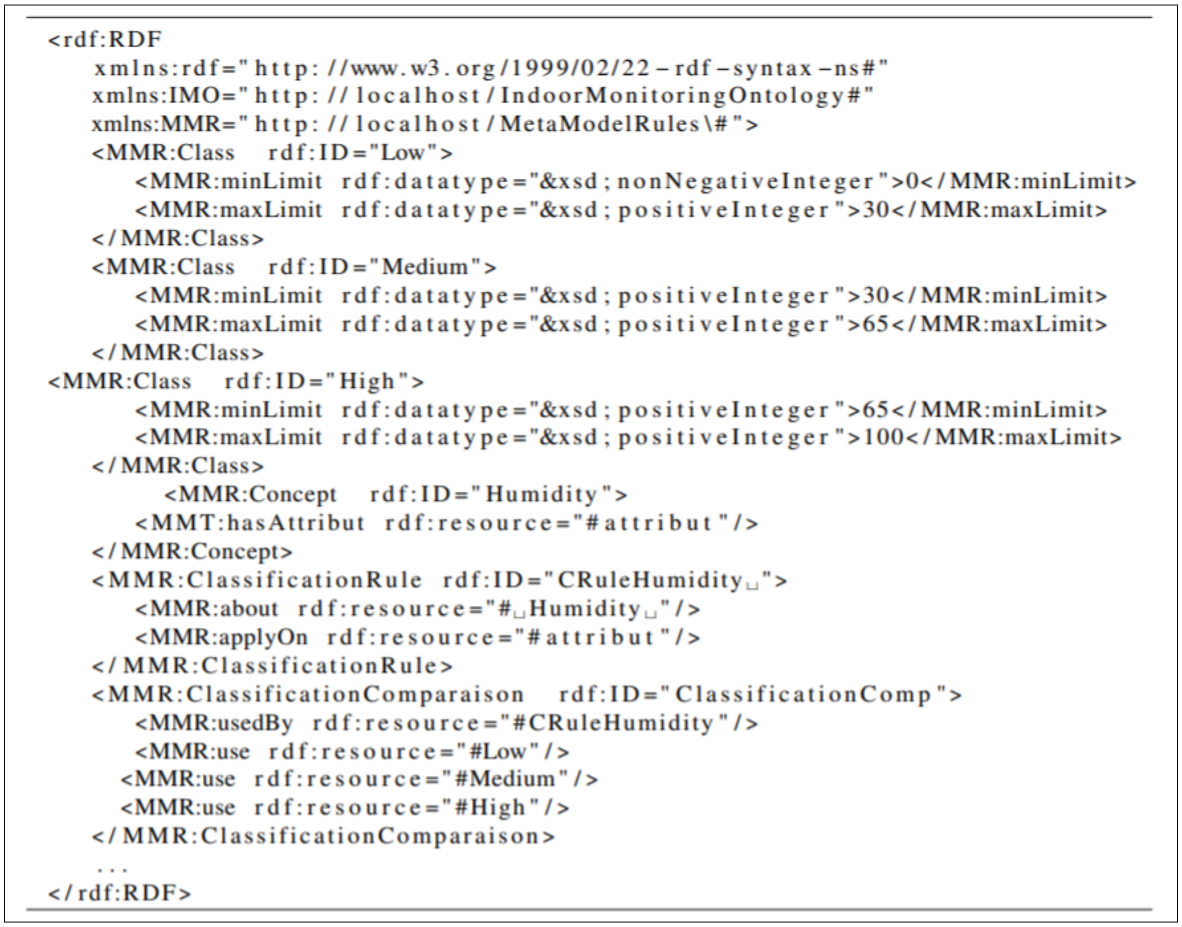
This component has a significant role in our approach, as it is intended to contribute to addressing RQ4. Data filtering process relies on semantic engine component to perform semantic reasoning over the received data. The semantic engine acts as an interface between the data filtering component and a knowledge base that intended to provide ontologies and concepts required for both filtering and annotation processes. At this step, the rules ontology is a gateway customized derived instance from the ontology of rules located at the cloud level. We propose to use the language of rules proposed in our previous work Bali *et al.* (2017). Three categories of semantic rules are defined in this language namely:

- rules based on the classification of data values;
- rules based on the old data value;
- rules based on a direct data value.

In this work, since a portion of the filtering process is delegated to the event detection component, we only use the first category of rules. This choice is justified by the fact that the selected type of rules imposes more restrictions on the processed data; hence, it is more likely to better perform in terms of filtering and minimization of data. Moreover, this implies that our approach could relatively be consuming less computing resources; since part of the filtering will be performed at the event processing layer, which relies on technologies that are lighter than the semantic web technologies.

Figure 2.5 presents an example of a semantic rule formatted as an rdf document. In addition to concepts used for rule description, other concepts are imported from other domain ontologies to define the domain elements that are associated to the rule. In our case, domain concepts are defined by Indoor Monitoring domain ontology referenced by the IMO namespace.

Table 2.4 lists some examples of rules based on the sensors used in the proposed scenario, and that can be modeled as an rdf file.



```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:IMO="http://localhost/IndoorMonitoringOntology#"
    xmlns:MMR="http://localhost/MetaModelRules\#">
    <MMR:Class  rdf:ID="Low">
        <MMR:minLimit rdf:datatype="&xsd;nonNegativeInteger">0</MMR:minLimit>
        <MMR:maxLimit rdf:datatype="&xsd;positiveInteger">30</MMR:maxLimit>
    </MMR:Class>
    <MMR:Class  rdf:ID="Medium">
        <MMR:minLimit rdf:datatype="&xsd;positiveInteger">30</MMR:minLimit>
        <MMR:maxLimit rdf:datatype="&xsd;positiveInteger">65</MMR:maxLimit>
    </MMR:Class>
<MMR:Class  rdf:ID="High">
        <MMR:minLimit rdf:datatype="&xsd;positiveInteger">65</MMR:minLimit>
        <MMR:maxLimit rdf:datatype="&xsd;positiveInteger">100</MMR:maxLimit>
    </MMR:Class>
          <MMR:Concept  rdf:ID="Humidity">
        <MMT:hasAttribut rdf:resource="#attribut"/>
    </MMR:Concept>
    <MMR:ClassificationRule rdf:ID="CRuleHumidity␣">
        <MMR:about rdf:resource="#␣Humidity␣"/>
        <MMR:applyOn rdf:resource="#attribut"/>
    </MMR:ClassificationRule>
    <MMR:ClassificationComparaison  rdf:ID="ClassificationComp">
        <MMR:usedBy rdf:resource="#CRuleHumidity"/>
        <MMR:use rdf:resource="#Low"/>
      <MMR:use rdf:resource="#Medium"/>
      <MMR:use rdf:resource="#High"/>
    </MMR:ClassificationComparaison>
    ...
</rdf:RDF>
```

Figure 2.5 Example of semantic rule

Table 2.4 Semantic Level Classification Rules

Sensor	Low	Medium	High		
Humidity	(Value<30)	(Value>=30)& (Value<=65)	(Value>65&&,Value<=100)		
Temperature	Very Cold	Cold	Warm	Hot	Very Hot
	(Value>-10 &<=10)	(Value>10 &<=15)	(Value>15 & <=25)	(Value>25 & TV<=40)	(Value>40)
Gas(CO2)	Safe		Unsafe	Dangerous	
	(Value>=350&<=1000)		(Value>1000 &<2000)	(Value>2000)	
Gas(CO)	(Value>=0&<=30)		(Value>=30 &<=400)	(Value>400)	

2.6.2.2 Data Annotation

This module receives data packets from the filtering sub-module. Each packet is composed of the SensorID, Type, location, measured value, and time stamp. Firstly, a mapping procedure is implemented to correlate data packets to the sensor description file using the sensorID. This step aims at importing extra data, such as Model, gatewayID, and unite of measurement, to enrich sensor description at the cloud level. For instance, sensors model property can be used to obtain extra information (i.e., accuracy, and latency) about the sensor from other resources, such as a database located on the cloud, or an RDF file that contains concepts for a more detailed description of the sensor. These data have been avoided at the previous processing steps for two reasons. First, they are rarely to change; thus, they do not affect the results of the preceding steps. Second, adding them at early stages would lead to enlarge the size of the processed data packets; which would lead to more consumption of computing resources. These extra data can be added to the annotated document at further steps, at the cloud level, to enable seamless integration between the collected data, and the data used by IoT services, as well as, to facilitate deeper reasoning processes to infer the relation between these data. This kind of process is likely to require more computing resources, and it is usually to be delegated to the cloud side.

In this work, we adopt two established ontologies namely: SSN[1] and DUL[2] as reference ontologies, which satisfy RQ3. While it is the choice of the developer to build domain ontology to represent main concepts of his/her application, in our case, to make our annotation general, we have built Sensor domain ontology (Sdo) and extended it by concepts from both aforementioned ontologies as depicted in figure 2.6. The main concepts of the developed ontology are: Sensor and SensorOutput, which are subclasses of ssn:SensingDevice and ssn:SensorOutput respectively. In the following subsection we describe some of the concepts introduced by our Sdo ontology:

- **SensorID:** it is linked to Sensor class through hasID property, and is intended to facilitate the search and accesses to a certain sensor;

[1] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn.

[2] http://www.loa-cnr.it/ontologies/DUL.owl.

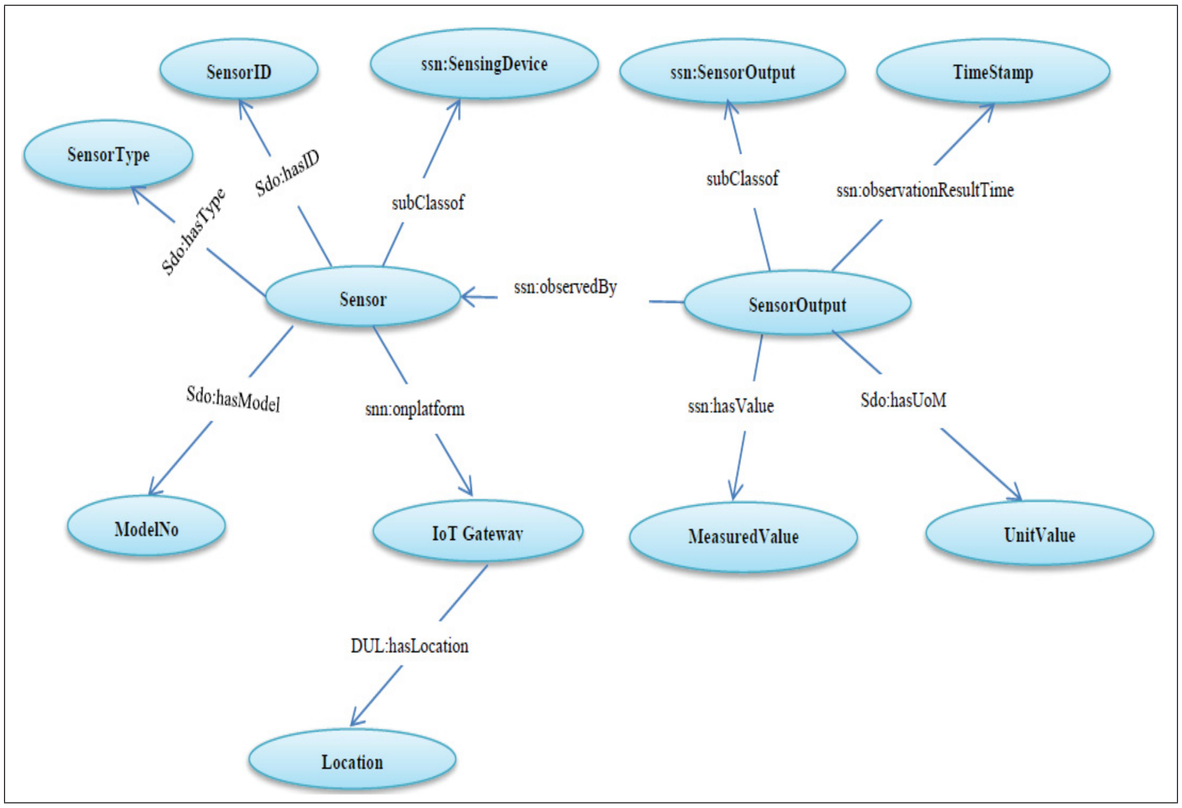


Figure 2.6 Fragment of our Sdo Ontology

- **SensorType:** this concept is presented to classify sensors based on their categories and linked to the sensor class by hasType property;
- **SensorModel:** this concept is presented to classify sensors based on their functionality, by which users can specify which filtering rules could applied. It is linked to the sensor class by hasModel property;
- **UnitValue:** this concept is presented to describe the unit of measurement of sensor readings and it is linked to sensoroutput class through hasUoM property. It can be extended to UoM ontology to present more details about the measured value.

To enable task priority management and to avoid conflict during the annotation phase, we adopt the concepts of priority preemptive scheduling scheme Burns (1993) in our approach. This mechanism is featured by its ability to handle tasks according to their priorities. Preemptive scheduling could interrupt the execution of tasks with low priority (i.e., normal events) upon the arrival of a higher priority task (i.e., critical events) to the ready-task queue. To do this, we assemble data resulting from the event detection phase in three data queues for processing namely, normal, warning, and critical. Afterward, three tasks for data annotation will run concurrently. The task that handles the data queue that encompasses the data representing critical events is given the highest priority. Thus, accelerating the manipulation of such types of events, this contributes to addressing RQ5.

The output of this step is an RDF file; which further transmitted to the cloud interface module. The annotation process enables software agents to smoothly consume data in an understandable format, which alleviate the burden of developers to integrate data from different sources, which contributes to addressing RQ2.

2.6.3 Cloud Interface Module

This module is designed to bridge the connection between the cloud and the proposed gateway. It is targeting three main tasks. First, providing functional independence between the physical level and the cloud services level. Second, transmitting data to the cloud in the form of RDF files. Third, receiving requests from high-level applications for sensor discovery and query of real-time data, or performing some actuating tasks. For reliable data transformation, we propose to employ a REST server. REST is a lightweight and, easy to use and deploy mechanism that facilities the integration between the web and IoT gateways. Also, this module holds an event rule manager, and a semantic rules manager to perform the task of updating the event rules file, and the ontology of rules.

2.7 Implementation and Evaluation of the Proposed Approach

In this section, we present our prototype implementation of the proposed approach and discuss the results. Our implementation takes into consideration the motivation scenario described in Section 2.3. The proposed approach has been evaluated by carrying out a set of experimentations using a simulated testbed that consists of three parts: sensor nodes, middleware, and the gateway. We have created a set of virtual sensors to simulate CO, and CO2 sensors for gas detection, as well as, to simulate the temperature and humidity sensors. Data collector component at the middleware side is configured to read sensors periodically (2000ms).

The middleware is intended to carry out real time data processing to contribute to realizing RQ1. We have used Java JDK 1.8.0 2 for implementing the prototype. For the event processing component, we have created a customized rule engine to implement event rules described in Table 2.3. For the semantic engine, semantic filtering, and the annotation process, the parsing of semantic documents has been handled using Jena[3] 2.12.1. The prototype is tested under Raspbian a light-weight Linux-based OS deployed on Raspberry[4]pi3 model B which has a Quad-core 1.2 GHz Cortex-A53 CPU, 1 GB RAM, and 16 GB SD card. We performed our experimentations to process sensor data in four cases namely: Without Event and Semantic Rules, With Semantic Rules and Without Event Rules, With Event Rules and Without Semantic Rules, and With Event and Semantic Rules. Several compression benchmarks are considered as follows:

Data Size: This category of evaluation criteria aims at measuring the overall average size of data generated in all aforementioned cases. The goal of this test is to identify how much data have been reduced using our Event and semantic-based approach compared to other cases. Reducing the data size denotes that our approach contributes to reducing the network traffic between the gateway and the cloud. This, in turn, would enhance the response time of the system using our gateway at the cloud level. As depicted in figure 2.7, With Event and Semantic Rules option performs better than other options. In some cases, it seems that With Event and Without Semantic Rules option performs relatively higher or equal to the With Event and Semantic Rules; but, better than the other two options, however, and as depicted in figure 2.8, the latter option takes significantly more concerning the overall runtime. On the other hand, Without Event and Semantic rules options always expose the highest average of data size, which emphasizes the

[3] https://jena.apache.org.

[4] https://www.raspberrypi.org.

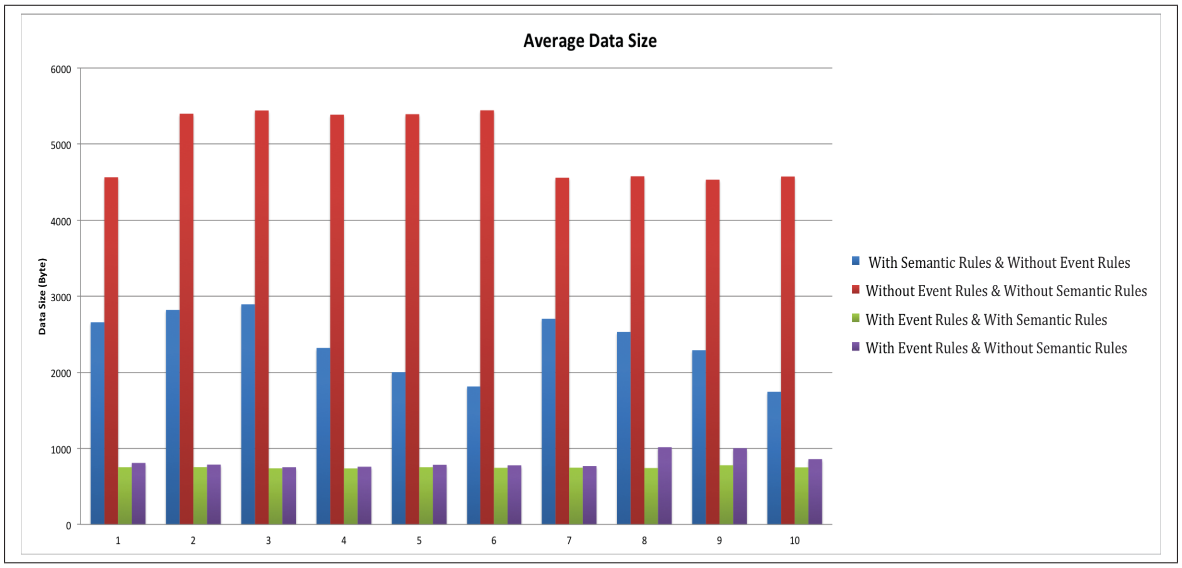
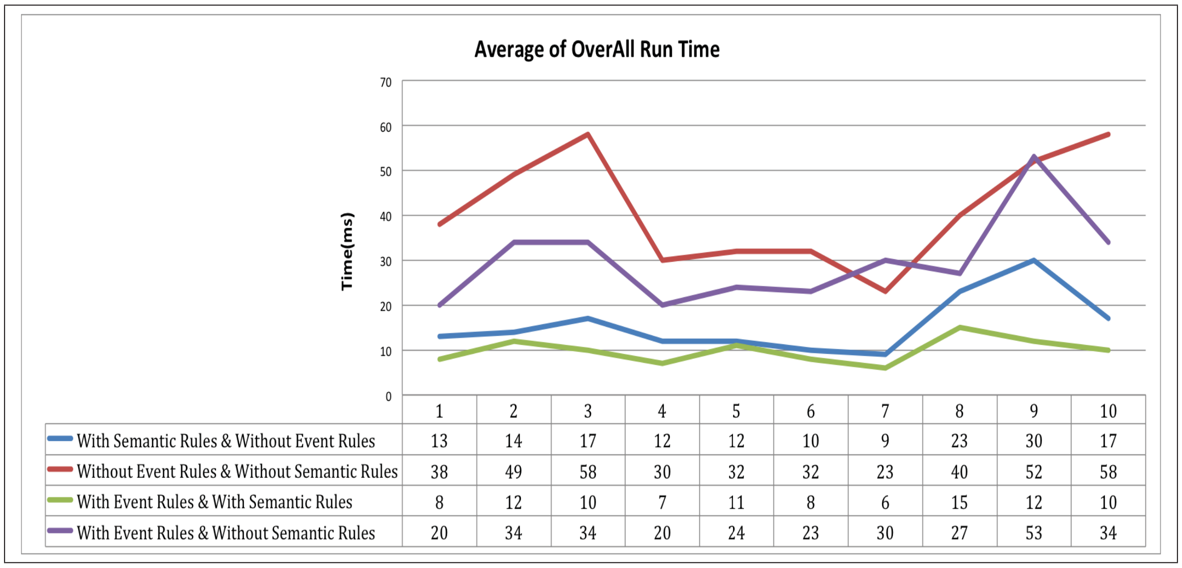


Figure 2.7 Result of Average Data Size Evaluation

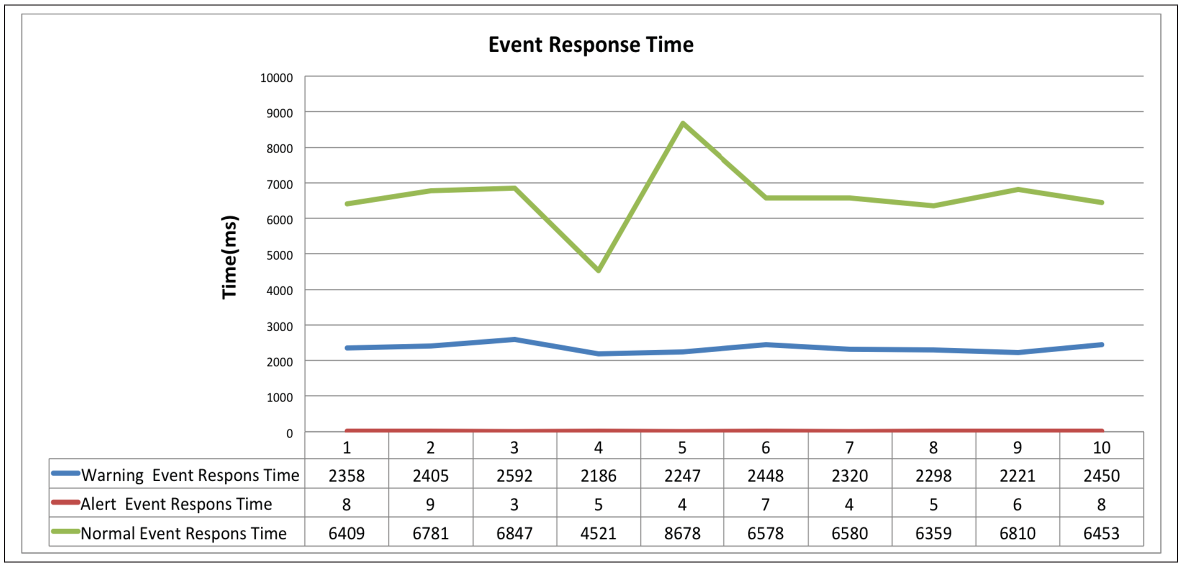
significance of applying filtering mechanisms to avoid redundant and invaluable data. Also, the figures in this type of evaluation show that the option With Event Rules and Without Semantic Rules filters better than the option With Semantic rules and Without Event rules. This can be justified by the type of rules used in each case.



	1	2	3	4	5	6	7	8	9	10
With Semantic Rules & Without Event Rules	13	14	17	12	12	10	9	23	30	17
Without Event Rules & Without Semantic Rules	38	49	58	30	32	32	23	40	52	58
With Event Rules & With Semantic Rules	8	12	10	7	11	8	6	15	12	10
With Event Rules & Without Semantic Rules	20	34	34	20	24	23	30	27	53	34

Figure 2.8 Result of OverAll Run Time Consumption Evaluation

*Overall RunTime:*In this category of evaluation, we seek to identify for which of the four aforementioned cases, the time consumed to get the annotated file ready to be sent to the cloud is the lowest. Figure 2.8 shows that With Event and Semantic rules have an overall time lower than the other options. This can be justified by the role played by this two processing layer architecture in terms of reducing the amount of data, which has resulted in lowering overall processing time. On the other hand, Without Event and Semantic rules takes a longer time than the other two cases. Figure 2.8 shows that unlike the former type of evaluation criteria, in this type With Event rules exposes higher value in terms of overall runtime than With Semantic rules. This is likely to be due to the number of rules applied at the event level being higher than the number of rules implemented at the semantic level.



	1	2	3	4	5	6	7	8	9	10
Warning Event Respons Time	2358	2405	2592	2186	2247	2448	2320	2298	2221	2450
Alert Event Respons Time	8	9	3	5	4	7	4	5	6	8
Normal Event Respons Time	6409	6781	6847	4521	8678	6578	6580	6359	6810	6453

Figure 2.9 Result of Event Response Time Evaluation

Event Response time: This category of tests is presented to assess the performance of our proposed approach at the different event classes that we have defined for our use case. More specifically, we measure the average response time, starting from receiving the raw data from sensor nodes till receiving a response from the gateway that the rdf document is ready to be sent in the three event classes: Normal, Warning, and Alert. As depicted in figure 2.9, Alert event type has been processed in a time that is significantly less than the other two event types. This is due to the priority of processing mechanism implemented by the proposed approach, which indicates the feasibility of the proposed approach of handling critical situations, and overcoming latency issue. On the other hand, Warning event type has been given a higher processing priority compared to the figures of the Normal event type.

Although conducted experimentations have shown promising results, more research and extensive experimentations are needed, especially using actual infrastructure to assess the efficiency of the proposed approach in terms of handling unpredictable and unobserved conditions.

2.8 Conclusion and Future Work

In order to facilitate the task of IoT systems in detecting events and handling them on time, we have presented in this paper our approach for sensory data processing on IoT gateways. The proposed approach has been designed taking into consideration the vitality of events that have the utmost impact on the observed situation, as well as, limited resources on IoT gateways. It is encompassed by two layers for real-time data processing. The first layer is based on the notion of an event-driven data processing. It is intended to handle the event detection process, by which the collected data are aggregated and classified as events based on predefined classification rules and concepts. This classification enables the system to accelerate the manipulation of events that expose unusual patterns that could harmfully affect the situation. The second layer is proposed to filter and annotate sensory data using SW technologies. This layer supports the task of high-level IoT applications in terms of interpreting and consuming raw sensory data, which enable them to react to events in real life scenarios on time. Furthermore, this layer contributes to mitigating the network bottleneck issue, since it reduces the size of data to be sent to the cloud. We have evaluated the feasibility of the proposed approach in the context of air quality monitoring system based on simulated environment. The results of the implementation have shown the potential of our approach in reacting to alert and warning class events within an acceptable response time compared to other cases. Also, the approach has proven its feasibility in terms of reducing data sizes. As a future perspective, we plan to conduct more experimentations, especially, in real life use cases taking into consideration adding more compression parameters, such as network traffic and resource utilization. Moreover, we will consider studying the load balancing algorithms to manage the distribution of the edge computing tasks based on resources available at the gateway level. Furthermore, to facilitate the detection of anomaly events, an investigation study is required to evaluate the applicability of applying Machine Learning techniques (ML) on limited resources devices, and to estimate the amount of resources needed to run ML techniques, in addition to, using SW and CEP techniques simultaneously on the same gateway device.

CHAPTER 3

MACHINE LEARNING AND FUZZY LOGIC APPROACH FOR EVENT DETECTION ON IOT EDGE DEVICES

Mahmud Alosta[1], Ahmed Bali[1], Abdelouahed Gherbi[1]

[1] Department of Software Engineering and IT, École de Technologie Supérieure
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
Paper submitted to the journal of Engineering Applications of Artificial Intelligence on MARCH 2021.

ABSTRACT In the era of IoT, the globe has witnessed a deluging of smart and tiny devices that are pervasively engaged in all aspects of our contemporary life, including homes, cities, vehicles, and factories. These devices have the capability to monitor the surroundings, capture observations of interest, and fulfilling actuation instructions. Typically, the captured observations are streamed across networks to the cloud for processing, analysis, and storage. However, cloud-based solutions for IoT data processing have exposed a relative inefficiency in terms of latency, network bottleneck, and bandwidth and service cost. In addition, in several IoT applications performing real-time data analysis, detecting events, and supporting real-time decision-making tasks are of utmost priority. Hence, any delay caused by transferring data over long distances and congested networks might not be tolerated. This has pushed toward a wide investigation and adoption of Edge-based solutions. Edge devices have been recently under ongoing efforts to improve their computing and communication capabilities. This has empowered them to carry out some processing and decision-making tasks closer to data so urces. Concurrently, several intelligent data processing techniques, such as Machine learning and fuzzy logic, have been incorporated into the development process of IoT systems. Machine learning models can be applied to reveal the ambiguity behind sensors' data behavior, and to realize accurate predictions that can improve systems' ability to proactively making decisions using the fuzzy logic rules. In the light of the aforementioned, this research work proposes an IoT edge oriented approach based on the integration of machine learning and fuzzy logic techniques to perform real-time data analysis, event predictions, and modeling, and support proactive decision-making tasks. The proposed approach performance has been evaluated within an air quality case study and several benchmark criteria have been evaluated including ML and FL efficiency, the response time, and resource utilization.

Keywords: Internet of Things (IoT), Edge Computing (EC), Machine Learning (ML), Fuzzy Logic, Edge analytic, Proactive Monitoring

3.1 Introduction

According to recent Cisco estimates, the number of Internet-connected devices will raise up to 29.3 billion by 2023, achieving a substantial increase compared to 18.4 billion devices connected in 2018. Interestingly, the share of M2M connections is expected to rise from 33 % in 2018 to nearly 50 % by 2023, which will result in 14.7 billion connected IoT devices CISCO (2020). This has increasingly pushed toward the adoption of the Internet of things (IoT) and its applications in all aspects of our daily lives. To name a few smart buildings, smart cars, industry, and health care sectors. This rapid expansion of IoT devices and applications, together with data floods they generate is expected to increase the network traffic.Where it is projected that the global Internet traffic will reach 4.8 ZB annually by 2022, compared to 1.5 ZB in 2017 CISCO (2020).

Recently, it has been significantly observed the large-scale deployment of sensors that are typically used to monitor the environment and to capture observations of interest on time. These observations are usually captured in the form of raw data that requires additional manipulation in order to decode the meaning and reap the value. Thus, it is directly or through a middle layer, these data are transmitted to the cloud for further processing and analysis. In some scenarios, services on the cloud are configured to send back some kind of controlling signals to the sensor node layer, for instance, to adjust the HAVC system, based on the conclusion derived from the received data. Taking into consideration the ever-increasing volumes of data generated by the tremendous amount of sensors, this would likely lead to place more pressure on the network and impact its performance. Additionally, some IoT use cases demand a quick response time, which cannot be guaranteed following the aforementioned topology.

To overcome the aforementioned challenges, a recent tendency of delegating some cloud processing functionalities to devices at the network edge, close to data sources, is gaining considerable interest from both the academic and industrial communities Shi *et al.* (2016). Several factors have motivated this diversion, which can be categorized into two main sets. The first set of factors is strongly associated with the category and the requirements of the IoT application under development. For instance, in some use cases such as health care, surveillance, and smart building control systems, privacy is of utmost priority. Thus sending all data and controlling such systems from the cloud might be of great concern due to security vulnerabilities that the cloud is highly exposed to Hawedi *et al.* (2018).

Additionally, in some time-sensitive IoT use cases, response time is of significant importance, thus any delay induced by the round trip of sending data to the cloud and waiting for analysis and getting back the response is not tolerable Yu *et al.* (2017). Such scenarios can be more obvious in IoT based fire detection and indoor air quality systems. Where detecting events of interest on time is a priority Al-Osta *et al.* (2019), which imposes the need for real-time data processing mechanisms with the least time possible. The second category of factors is mostly relevant to the cost of blindly transmitting data to the cloud. Although most cloud services provide a limitless amount of resources, they follow the pay-on-demand model. As well as, more data to be transferred leads to more network traffic, which eventually leads to more bandwidth consumption, in addition to, network congestion issues.

Due to the vast amount of data generated by IoT devices and new application requirements, real-time analysis of these data is essential. The temporal and spatial aspects of the collected data induce the on-the-fly processing to decode the meaning and to disclose behavioral patterns, which can be used as competent knowledge to serve several decision-making based applications and services. Thus, incorporating intelligent techniques, such as machine learning (ML) and fuzzy logic (FL), into the IoT data processing pipeline is likely to boost the performance of its applications.

Essentially, the capability of ML algorithms of making accurate predictions on time-series IoT data would raise the efficiency of IoT applications to plan and act proactively to predictable upcoming events. This can be performed by analyzing historical data to reveal the coloration between different sensors recorded readings, as well as between data sequences coming from the same sensor. Thus, the generated model can be used to forecast the unseen data and enable a smooth and proactive decision-making process. However, historical data might not sufficient to reflect the dynamic changes in the environment. Simultaneously, some IoT systems have higher requirements for real-time performance. Hence, to improve the responsiveness of IoT systems, ML algorithms have been extended to the edge layer, thus, enabling intelligent edge devices endowed with the capability of real-time data analysis. Nevertheless, the role of the cloud can't be avoided and ML models can't be fully trained and inferred at the edge due to resource limitations challenge. But it would be of significant benefit to performing in a cloud-edge-device coordination manner via data offloading strategies Carvalho *et al.* (2020). In addition, the cloud is more efficient for the persistent storage of large data sets and to perform large-scale data analysis tasks.

The benefits of extending intelligent ML algorithms to the edge devices and closer to data sources have been investigated in the literature Abdulkareem *et al.* (2019); Merenda *et al.* (2020); Murshed *et al.* (2019). Several propositions have been highlighted in the literature that the core of their efficiency is the integration of ML and statics techniques that have the potential to harness the IoT data floods and create predictive models such as predicting air pollution Zhang & Woo (2020), fall detection Sarabia-Jácome *et al.* (2019), indoor temperature Paul *et al.* (2018), and industrial maintenance Ruiz-Sarmiento *et al.* (2020). These approaches can be envisioned as the basis of proactive solutions for IoT applications, but they are likely to function insufficiently when it is perceived from event identification and decision-making perspectives that are likely to be better handled by the fuzzy logic.

Fuzzy logic (FL) is multivalued logic used to derive a meaningful conclusion based on the aggregation and inference over multiple parameters. These parameters can be envisioned as sensory data packets that might be noisy, imprecise, vague, and ambiguous. More specifically, FL addresses the partial truth concept where the range of the truth value may span from completely true to completely false Mittal *et al.* (2020). This is a significant feature that can be used to improve IoT systems responsiveness, especially in handling and modeling uncertainty events. In addition, several operational factors have motivated applying of the FL concept to IoT solutions, including its ability to imitate human reasoning, tolerate unreliable and imprecise sensor readings, improve decision-making tasks, and reduce resource consumption. Several works in the literature demonstrate the agility of the FL approach in identifying events of interest Kapitanova *et al.* (2012); Maksimović *et al.* (2014). Furthermore, several other works applied the FL concept to micro-controllers and gateway devices such as Arduino and raspberry pi, which demonstrates its small foot-print and its suitability to function under resource-limited environments Chiesa *et al.* (2020); Palevi *et al.* (2019); Gozuoglu *et al.* (2019).

In our previous work Al-Osta *et al.* (2019), we handled the event detection concern by using event complex rules and semantic web techniques. The approach has proven its feasibility in detecting events of interest, and assigning priority of processing to critical conditions, as well as, behave ***’reactively’*** to detected events.

In this work, we proceed within the same research direction, nevertheless, more emphasis has been placed on improving edge-based IoT solutions to perform ***’proactively’***. This is achieved by embedding the ML analyzer and the FL inference modules into our proposed approach. The former module is intended to improve IoT Edge-based solutions on predicting future data using fine-tuned ML models. While the fuzzy logic inference engine coordinates the decision-making process through two levels of reasoning. The first level is intended to decode the meaning from the received data points predicted by the former model and consolidate them in the form of an event of interest. This event will be in a further step used as an input to FL based decision-making to recognize the necessary precaution that can be planned beforehand. Since our approach is time-series oriented and dependent on predicted data to identify events, we propose the concept of ***’model credibility’***. This concept is used as an indicator of the system’s reliability and as an additional parameter to promote the decision-making process. In the following points, we summarize the main contributions of the presented work:

- introducing a conceptual design of a cloud-edge based framework to facilitate the creation and evaluation of intelligent edge-based solutions for IoT data processing;
- the investigation of best practices for efficient ML and FL-based solutions deployment in resources-limited environments such as IoT edge devices;
- the evaluation of multi-step recursive prediction strategy using several ML models simultaneously and its suitability to IoT edge devices;
- the design and implementation of FL rule-base model that decomposes the event detection process, so it empowers a lightweight mechanism for complex event detection in resource-restricted environments;
- introducing the ***'model credibility'*** concept, and incorporating it in the decision-making process, so more system reliability is ensured;
- an in-depth analysis of an indoor air quality use case revealing the significance of early event detection mechanisms;
- the realization of the proposed approach in the context of an indoor air quality use case by using a real dataset, and considering different implementation scenarios.

The remainder of the paper is organized as follows: Section 3.2 presents the related work in the light of the approach proposed in this paper. Then, the use case followed in this work is presented in Section 3.3. The conceptual design of the components forming our proposal is presented in Section 3.4. This followed by a deeper explanation of the modules constituting the proposed approach in Section 3.5 and Section 3.6. While Section 3.7 presents an exhaustive exploration of the used dataset in addition to the experimental setup and result discussion. Finally, Section 3.8 concludes the paper and highlights perspective directions for future work.

3.2 Related Work

Collaborative frameworks have been proposed to gain the best of both paradigms: the cloud and the edge Mora *et al.* (2018). Moon *et al.* (2019) proposed a cloud-edge collaboration framework for IoT data analytic. The cloud generates ML models and selects the most appropriate one to be deployed on the edge side to predict future PM10 and PM2.5 concentrations in a specific location. The selection process is done based on the correlation between the data used to train the model and the data gathered at the location. The work by Rahman & Rahmani (2018) investigated the advantages of using ML over ordinary rules to control IoT systems through a distributed intelligent system. Their hypothesis revolves around providing low-level intelligence to process 'small data' at the edge while providing high-level intelligence to process 'big data' at the cloud. Belief-network was chosen to evaluate the solution in a simulated scenario. The presented approach in this work differs from both works where it offers controlling mechanisms to empower decision-making tasks by the system. The decision can be performed by the user or automatically by edge devices in case of an emergency.

Likewise, Taneja *et al.* (2019) proposed an approach to distribute and decompose the data analytics workload in the edge layer. They emphasize the importance of knowledge gathered from local areas to get a better view of the parameters under consideration. They adopt a tree-like topology considering the cloud as a root, edge as an intermediate node, and IoT devices as leaves. Edge nodes perform the data analysis task using a multivariate linear regression model and send the result to the cloud. They concluded that the distributed approach has considerably lowered resource consumption at the cloud and significantly reduced the amount of data sent to the cloud. We share a similar interest in this work to provide data analytic capabilities at the edge of IoT networks.

Another collaborative framework presented by Rajith *et al.* (2018) for real-time HVAC control system based on IoT cloud-edge framework. They used the edge as a data collector, aggregator, and forwarder, while they placed the prediction engine on the cloud since they use the Weka software, which relatively requires resources, to analyze sensor data. Besides, our approach provides additional support to the IoT edge-solutions by implementing some event detection mechanisms that further promote proactive decision-making functionalities.

In addition, several approaches have been proposed in the literature demonstrating the feasibility of effective edge-based analytic workflows. A recent work by Da Rosa Righi *et al.* (2020) proposed an IoT-based notification system that collects and analysis dairy farm data. Their architecture is resilient to meet edge and cloud implementation scenarios. They used the Auto-Regressive Integrated Moving Average model (ARIMA) based prediction engine to facilitate in advance notifications of the nutritional behavior of cows and its expected impact on milk production, so better nutritional plans can be adapted. While, Syafrudin *et al.* (2019) demonstrates the affordability edge-based solution for early fault detection and warning in industrial applications. Raspberry pi was used as an edge device. It holds an inference engine running a Random Forest model supported by DBSCAN to improve its accuracy. The edge is directly sent notifications to a web interface for visualization; thus no cloud communication is required. Both works are fallen within efforts to improve IoT based systems in performing proactively based on predictive models; however, they lack actuation mechanisms.In addition, they only focus on the prediction of one parameter, unlike the proposed approach, which considers edge-based reasoning over multiple parameters.

Besides Janjua *et al.* (2019) proposed a framework for IoT edge-based rare event detection. In order to avoid the labeled data, they used an unsupervised learning approach that combines BIRCH and Agglomerative Clustering techniques. While the former is intended to shape the steamed raw sensory data as microclusters. The latter is applied to form macro clusters. This is done by merging the micro-clusters resulted from the first step based on the distance between the cluster centroids. This approach considers audio data as their target use case and it was realized on raspberry pi. In our work, we are also seeking to improve IoT edge-based systems efficiency in detecting events of interest. In addition, the FL concept has been adopted to convey meaningful descriptions of the event, which is obtained from fusing raw data coming from different sources.

Additionally, the literature highlights several works that manifest the feasibility of integrating ML and FL techniques in the IoT. Within this context, Shukla *et al.* (2019) proposed a hybrid approach of fuzzy logic and reinforcement learning for sensory data processing. This approach is based on 3-tire IoT architecture, where data are first gathered at the first layer and classified using a fuzzy-based inference engine. Then the systems forward the data to the second layer (fog) that runs a reinforcement learning model to select the most time-sensitive data to directly sent to the end-user, while data with less importance are sent to the cloud for storage and future use. While Iram *et al.* (2019) proposed a real-time traffic controlling system using machine learning and fuzzy logic. The system was deployed on several edge devices to perform context-aware traffic signals controller based on the occupancy information recorded in a specific location. Occupancy information is gathered on a large-scale; analyzed and clustered using the K-Means algorithm. Then fuzzy logic applied on the clusters, thus a longer duration is assigned to edges with a higher density of traffic. A Similar combination of ML and Fuzzy logic techniques are proposed in the presented work; however, in addition to using fuzzy logic in deducing over predicted sensor data to detect and classify events, it is also used to perform the decision making based on some kind of credibility measure induced from the ML model. Also, our approach assumes that both ML and FL run on the edge which is likely to ensure faster response time.

Adeleke *et al.* (2017)investigated the applicability of integrating ML models into the sensor semantic web to promote proactive IoT systems. They evaluated their proposition in the context of an indoor air quality monitoring scenario. Several ML models were assessed including MLP, BN, DT, and RF to classify the PM2.5 observations according to predefined classes and within predefined sliding windows. Then the result generated from the predictive component is forwarded to the stream reasoning framework in the form of Resource Description Framework (RDF) triples. A C-SPARQL engine equipped with queries is used to perform reasoning and to facilitate the monitoring and decision-making tasks. When a situation of interest is detected a proper notification or an actuation instruction is initiated to avoid the undesirable consequences. While we share a similar interest in improving the IoT system to behave proactively, the mentioned work is intended to perform classification and their data were labeled manually into two classes, and based on that the decision is made. While in this work the FL concept is applied to forecasted time series data (Not a label class) to detect events and classify them in a more resilient manner that is reflected in the decision making procedure.

Also, de Mattos Neto *et al.* (2014) proposed an intelligent system for air quality forecasting, emphasized on the time-series recorded observations of PM2.5 and PM10 concentrations. They employed an optimized MLP as their prediction model. The optimization process is intended to adjust the number of input nodes and hidden layers of the MLP. The MLP has been also investigated in our work and it has been compared to other ML techniques to guide an objective trade-off selection that considers model efficiency in addition to its consistency with real-time implementation on IoT edge devices.

3.3 Use Case

Due to the remarkable reduction of the installation cost, IoT systems have been widely adopted for real-time indoor monitoring tasks. Typically these systems are configured to report their recorded observations to the cloud to facilitate remote monitoring and control. However, in some time-sensitive scenarios, the round-trip delay induced by the cloud can't be tolerated.

This work considers monitoring indoor air quality (IAQ) parameters as a motivating use case scenario. Since the majority of people spend a considerable portion of their time indoors, environmental indoor parameters have become a major influence on human health Marques & Pitarma (2018). This influence is likely to be more tangible, especially with the current lockdown regulations imposed by the vast spread of ***COVID-19***. The abiding exposure to poor air quality could lead to short and long-term serious health issues, such as asthma, respiratory tract infections, and allergic reactions. Moreover, even short-time exposure to high levels of some indoor air pollutants, such as CO2, can cause dwellers discomfort, drowsiness, and lethargy. Thus, relevant research activities have been reported that their main objective is to develop cost-efficient systems for indoor air quality monitoring Li *et al.* (2019); Molinara *et al.* (2020); Moon *et al.* (2019).

The presence of particular pollutants, such as CO2, PM2.5, and VOC, can be detected using IoT sensors. These sensors can be installed on different locations to monitor the environment and periodically report their observation to an intermediate device (i,e. edge) which in turn forward it to the cloud. Cloud platforms have the ability to store a big amount of historical data. The deep analysis of that can be of great benefit for assisting healthcare providers to support the medical diagnosis. On the other hand, short term detection of indoor air pollutants is vital to accurately identify the current status of the ambient environment. Where in some cases, especially with people who suffer from some respiratory-related health issues, such as ***COVID-19***, exceeding certain limits of pollutants exposure might lead to hazardous consequences Mehmood *et al.* (2020).

Table 3.1 The Short and Long Term Effects of Indoor air pollution

Taken from Sharma *et al.* (2020); World Health Organization (2013); Davis (2019); EPA (2017); Mad Saad *et al.* (2017)

Pollutant	Short-term Effects	Long-term Effects
CO2	Headaches, nausea, dizziness, and sleepiness	Stress, kidney calcification, bone demineralisation
PM2.5	Increased hospital admissions for heart or lung causes	Respiratory diseases (e,g.asthma),mortality hazard
PM10	Eye irritation, worsening of respiratory diseases	Lung cancer
VoC	Eye, nose and throat irritation Headaches	Damage to liver, kidney and central nervous system

As shown in Table 3.1, short and long- term exposure to indoor pollutants is the potential to lead to severe medical conditions that in some cases would require intensive health care. Moreover, both ambient temperature and humidity are associated with several health issues that are relatively varying in intensity. For instance, low humidity levels are linked to dry eyes and itchy skin, as well as, increased vulnerability to colds and respiratory illness. While high humidity levels could cause body dehydration, discomfort, and fatigue. Similarly, improper temperature levels have been associated with several health conditions such as skin pain and arrhythmia Mad Saad *et al.* (2017).

Ventilation systems are used to regulate ambient air parameters such as temperature and humidity, as well as pollutants concentration. Thus, maintaining thermal comfort and acceptable IAQ levels. IAQ monitoring systems are a valuable source of stream data that can be used to regulate the HVAC systems and enhance the occupant's well-being, concentration, and productivity in indoor spaces. This requires real-time supervision of IAQ configured based on predefined intervals. These intervals might be presented in the form of notification messages to the user; so She/He can take control of the situation. Or it may be initiated in the form of controlling instructions based on some kind of predictive engine results. Thus, it becomes feasible to take appropriate decisions on time and avoid any undesirable consequences. To cope with the aforementioned requirements, the approach presented in this work is intended to push such kind of intelligence into the network edge closer to data sources, so faster data processing and analysis, as well as, decision-making tasks are ensured. In more detail, this work can be envisioned as a part of the monitoring system that is based on a predictive analysis by estimating peak levels of indoor air pollutants. Thus it becomes feasible to warn dwellers about pollution and reduce the exposure of sensitive people.

In addition, it is of great benefit to the IAQ data to be transferred to the cloud in a format that can be shared and integrated with other sources of data either from other locations or data from the web. Thus a large scale deep analysis is possible to be performed, and long term patterns and effects can be unleashed. This process requires the data to be semantically annotated using universal and domain-specific concepts Al-Osta *et al.* (2017). These concepts are usually grouped and shared in the form of ontologies. Ontologies have been widely used to integrate data from different sources on the web. Recently they have been extended to the IoT domain to alleviate the data heterogeneity issues resulting from the diversity of the underlying technology stack and infrastructure Hachem *et al.* (2011).

3.4 The Overall Design of the Proposed Approach

As shown in figure 3.1, a three-layer architecture is envisioned in this work, namely sensor nodes, edge, and cloud layers. At the lowest layer (Layer 1), sensor nodes are deployed, which is composed of a set of devices characterized by its limited-resources. These devices (i.e., sensors and actuators) are intended either to collect data and forward them to the upper layer (Layer 2) or to receive and fulfill controlling instructions. They are likely to be grouped into clusters located within limited physical boundaries and connected to one or more edge devices. Edge devices on Layer 2 offer more capabilities compared to devices at Layer 1; thus they handle the burden of locally aggregating and analyzing data in a timely manner. As such, a quick decision can be performed and avoid the round-trip delay. The output of edge nodes has two directions either to be sent forward to the upper layer (cloud) as a result of analysis or to be sent back to the lower layer (sensors) as a controlling instruction. Furthermore, the edge layer bridges the communication between sensor nodes and the cloud; thus it becomes feasible to monitor and control devices remotely.

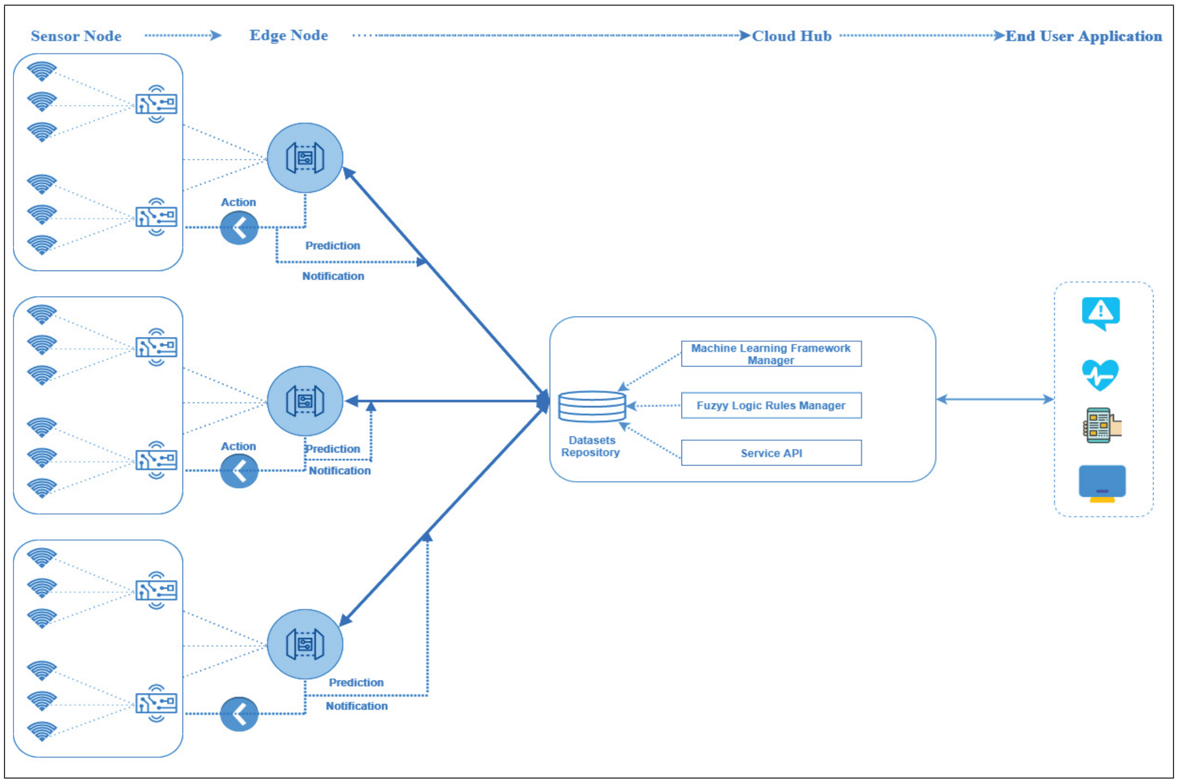


Figure 3.1 The Overall Design of the Proposed Approach - The Three-Tire Architecture

On the other hand, Cloud platforms (Layer 3) offer a large scale monitor and control over communities of deployed edge nodes. Moreover, these platforms are featured by the availability of plenty of resources on demand; this endows them with capabilities to carry out resource-intensive tasks such as ML model training, and data analysis and visualization on large stored datasets. Thus it becomes feasible to extract insights, patterns, and correlations from diverse data generated by different IoT networks and edge devices connected to the cloud. In our approach, the cloud also guides the selection of the ML model that fits the need of specific task and data characteristics, and deploy it on the targeted edge platform. In this work, a machine learning framework manager is presented to be in charge of creating, testing, and selecting machine learning models to be later deployed on the edge.

3.5 Cloud Layer Components

In this section, a detailed description is provided to modules that proposed to initialize ML models and FL rules intended to be deployed and run in later stages on the edge side.

3.5.1 Machine Learning Framework Manger (MLFM)

This module is located on resource-rich platforms, either on the cloud or on the fog layer. It represents the logical sequential steps followed to build ML models. The execution of this workflow produces ML models that will be deployed and used in online predictions at the edge layer. In this framework, one of two main tasks is performed based on the received request as described below:

- **model creation:** this step represents the primary stage of the model life cycle. It starts with the creation of the data set to be used to train a model. This data set might be generated from single or multiple data sources given the task of the service to be equipped with the trained model;
- **model retraining:** this task is initiated by receiving the retraining request from a certain edge device or it might be configured periodically. The objective of this step is to generate a new model after updating the data set with new data instances and compare the accuracy of the two models.

3.5.1.1 Pre-processing and Feature Engineering

This step has a significant impact on the performance of the created model. It specifies the preprocessing steps to be applied to the dataset defined in the previous step. These tasks are defined for each dataset along with their parameters. For example, data aggregation and transformation is a common preprocessing task used for IoT data and can be specified in this step. Moreover, Feature extraction is performed to define or create new features from the set of basic features. For instance, from the time stamp feature, it is possible to define the day feature and identify trends based on it. The following feature engineering tasks have been implemented in this work:

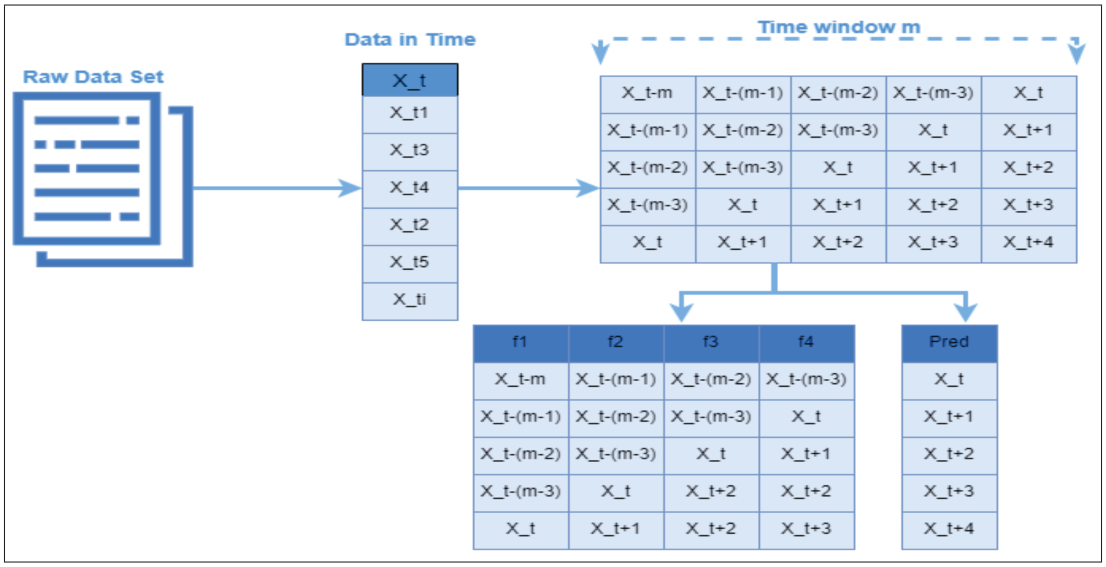
- **feature scaling:** in machine learning, scaling, which also called normalization, refers to the process that conveys equality of scale to values range of independent variables or features of data. To elaborate more, it is likely that the feature set used to train a model to be of a different value range. For instance, in our use case, the value of temperature ranges from 0 to 50, while the CO2 values range is from 0 to 5000. So, if an ML model were trained on these features without scaling, the model would assign higher weights to higher values and lower weights to lower values, disregarding the units of values.Thus, model performance would be more biased and impacted by parameters with higher weights. By using min-max scaling, data are being transformed such that the features' values are determined within a specific range e.g. [0, 1], and assigned equal weights. A Min-Max scaling is typically performed using the equation listed below:

  $$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

  where x is the original value of feature f, x̀ is the normalized value.

- **Exponential Moving Average (EMA):** EMA is a smoothing method derived from the weighted average method. It assumes that the importance of the data decreases non-linearly with time. It can eliminate the unexpected change in the time-series and learn the trend. This step has been mostly used to reduce the noisy time-series data. It's also called "smoothing" the data. It is achieved by essentially weighing the number of observations and using their average. This is known as the Moving Average(MA). EMA can be categorized as an MA that emphasizes more significant on the most recent data points by assigning them higher weights Montgomery *et al.* (2015);

- **lag features:** this step is essential to transform time series prediction tasks into some kind of classification task. This typically requires transporting time series sequence data into an ***'attribute-value table'*** that is used as input to machine learning prediction models Parmezan *et al.* (2019). This is achieved by using values of a specific factor at previous time steps as input variables and the next time step as the output variable. Simply, given a sequence of numbers for a time series dataset, it is possible to predict the value at the next time (t+1) given the value at the previous time (t-1). First, a time window of size (m) is specified, which represents a range of sequential sensor readings. Each of them is considered as a feature value 'f' in time 't'. The combination of these features consolidates the vector of features v(f) in the time range t=1 to n as shown in figure. 3.2.



X_t
X_t1
X_t3
X_t4
X_t2
X_t5
X_ti

X_t-m	X_t-(m-1)	X_t-(m-2)	X_t-(m-3)	X_t
X_t-(m-1)	X_t-(m-2)	X_t-(m-3)	X_t	X_t+1
X_t-(m-2)	X_t-(m-3)	X_t	X_t+1	X_t+2
X_t-(m-3)	X_t	X_t+1	X_t+2	X_t+3
X_t	X_t+1	X_t+2	X_t+3	X_t+4

f1	f2	f3	f4
X_t-m	X_t-(m-1)	X_t-(m-2)	X_t-(m-3)
X_t-(m-1)	X_t-(m-2)	X_t-(m-3)	X_t
X_t-(m-2)	X_t-(m-3)	X_t	X_t+1
X_t-(m-3)	X_t	X_t+2	X_t+2
X_t	X_t+1	X_t+2	X_t+3

Pred
X_t
X_t+1
X_t+2
X_t+3
X_t+4

Figure 3.2 Time Series reframing

3.5.1.2 Model Selection and Training

It defines the ML algorithms along with their parameters to be used to train ML models. For example, the type of algorithm (e.g.BN, MLP, SVR, KNN), specific algorithm configurations, and how the datasets are split between training and testing are some of the parameters defined in this step. In the following subsections, we briefly present four state-of-the-art machine learning algorithms that we consider in our experimental evaluation.

- **K-Nearest Neighbors (KNN):** KNN algorithm is widely used for both classification and regression use cases. It is a non-parametric method that does not require any prior knowledge about the distribution of data used for training. Thus stimulating the training phase and revealing a reliable performance especially for non-linear series. KNN's core idea lays in measuring the similarity among samples in the features space. Each sample combines both, a vector of features and its associated label class (for classification) or numeric target value (for prediction). In the case of a new sample with an unknown target value, KNN tracks its nearest and most similar K neighbors and aggregates some metrics such as the mean or the medium to predict the unknown target value. Also, some metrics such as the Euclidean, Manhattan, and Hamming distance can be used to locate the K of a data point Goyal *et al.* (2014). KNN models are likely to be easy to implement and to quickly fit; however, they are relatively slow while performing predictions. This is because it requires to search all data points in its training asset to find the nearest ones;

- **Multilayer Perceptron (MLP):** MLP is an artificial neural network (ANN) that imitates the human brain in performing information processing. ANN is structured as a set of neurons (nodes) that are distributed in layers and linked to each other. Essentially, neural networks are capable of identifying unseen patterns and achieve generalization by learning from both data patterns fed to them and the errors they observe while training. MLP is well-known for its ability to represent any smooth measurable functional relationship between the inputs (predictors) and the output (predictands). A typical MLP structure is composed of at least three layers of neurons namely: Input and output layers, as well as, a set of intermediate layers called hidden layers. The latter layers consist of units that are intended to map between the input and output unit characteristics and they do not directly interact with the environment. An efficient mapping process between the input and output units is considerably dependent on the existence of a sufficient connection between the input units and a substantial set of the hidden units. Determining the number of the hidden layers and specifying the number of neurons in each layer has been recognized as a significant factor in assessing the efficiency of the MLP model Taud & Mas (2018);

- **Bayesian network (BN):** BNs are designed leveraging the combination of both graph and probability theories. The former is used to graphically represent the structural model of BN; while the latter is used to compute uncertainties based on the probability concept. The structured model is composed of nodes and arcs that are integrated into the form of a directed acyclic graph (DAG) Maimon & Rokach (2005). Each node in the graph is used as a representative of a discrete or continuous variable. While arcs are annotated by probability distributions that represent the interactive dependency relationship between such variables. Thus the conditional and joint probability distributions over all the nodes (variables) in Bayesian networks are specified graphically. Typically, the learning process of a Bayesian network from data is carried out through the induction of its two different components. Where the graphical structure of conditional dependencies represents the model selection step. While the probability distributions and quantifying the dependency structure represents the parameter estimation step. Generally, Bayesian analysis computes the probability of a parameter in a model likewise revealing information learned about the parameter from data. This is accomplished using the "prior belief" concept which denotes performing a prediction about the parameter prior to revealing the data. Additionally, upon the availability of new data, the "prior belief " can be updated which likely to lead to model improvement Sebastiani *et al.* (2009);

- **Support Vector regression (SVR):** SVR is a machine learning algorithm based on statistical learning theory and structural risk minimization principle. It is a version of the Support Vector Machine (SVM) used for regression purposes. SVM is a learning pattern that uses a high dimensional feature space. The prediction function yielded by SVM is expandable based on a subset of support vectors (SV).In regression, SV represents input data points that realize the maximal margin Steinwart & Christmann (2008). SVR transforms input data points into a lower-dimensional feature space using a nonlinear kernel function to generate a linear regression function with a reduced number of estimated parameters. This likely leads to a higher generalization performance resulted from using support vectors for forecasting compared to using multiple parameters. In addition to the kernel, SVR includes penalty coefficient and non-sensitive coefficient parameters as key points to optimize its prediction efficiency. To obtain optimum parameters, a grid-search algorithm is applied. However, it is likely to be time-consuming especially with large data sets Iram *et al.* (2019).

In addition, several ML validation criteria such as RMSE, MSE, and R2score are widely applied to measure the accuracy of the predicted scores. These criteria can be used to guide the selection of a certain model over another. Afterward, a versioning function will be used to tag the model with a unique ID. The ID will be used later to facilitate access to the model for deployment or comparison purposes. In the following, we introduce the R^2 score which is used to assess the model performance in our work.

3.5.1.3 R-squared (R^2)

R^2 is a statistical measure that assesses the closeness of the predicted data to the fitted regression line. More specifically, R^2 can be accepted as an adjustment indicator that demonstrates the degree to which the variance in the values of the independent variables is reflected in the variance in the dependent variable value.It is also known as the coefficient of determination and its value typically ranges from 0 to 1. However, the values of R^2, in some cases, might expose negative values. This is likely to result when the predicted values are being compared to real values that have not been extracted from a model-fitting procedure using those data. R^2 value can be computed using the following formula:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\Sigma i\left(y_i - \hat{y}_i\right)^2}{\Sigma i\left(y_i - \bar{y}_i\right)^2} \tag{3.2}$$

Two parameters are being computed as explained by the R^2 formula namely: the residual sum of the squared errors of a regression model ***(SSres)***, and the total sum of squared errors ***(SStot)***.

The latter is represented by a flat line that assumes that every value of y is the mean value of y. R^2 is being verified to confirm that the fitted regression line will predict better than the mean.

3.5.2 Fuzzy Logic Framework Manager (FLFM)

In this module, the concept of FL has been proposed to enable the detection of meaningful events from raw predicted sensory data. The steps followed for detecting event of interest is as depicted in figure 3.3. Mainly, FLS is consisted of fuzzification, fuzzy rules base, fuzzy inference system and defuzzification.

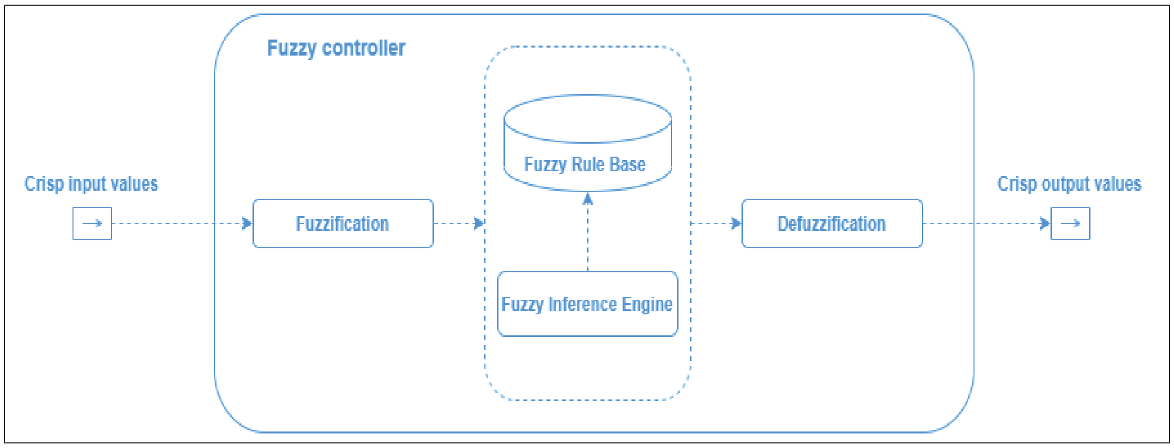


Figure 3.3 The process of fuzzy logic

- **Fuzzification:** this step is intended to convert each crisp value input into a fuzzy value represented by a membership degree specified by a corresponding membership function. In our case, crisp inputs are in the form of sensor readings including, CO2, PM25, temperature, or humidity captured observations. For instance, a room temperature = 20 C, can be converted using the membership function to a fuzzy linguistic value (eg. the temperature is warm). The membership function might be represented graphically using different shapes. Some of the most commonly used shapes include triangular, trapezoidal, and Gaussian shapes;

- **Fuzzy Inference Engine:** it is also called the fuzzy inference process. It is combined of If-Then rules, logical operators, and membership functions. The rules are grouped in the fuzzy rule base, which is consisted of a set of linguistic statements that describe how the fuzzy inference system should identify an input or control output. The rule-base is likely to be organized taking into consideration all possible value combinations connected by logical operators(e.g. AND, OR, NOT). The presented approach considers two dependent fuzzy outputs namely: Event and recommended action. To detect the event, we consider 4 input variables, taking the above-mentioned use case as a reference. Each one of these variables is linked to 4 member functions resulting in a total of $4^4 = 256$ rules that can be applied. This is likely to increase the size of the rule-base exponentially to cover the number of linguistic variables. In order to reduce the number of rules to meet the resource limitations of IoT edge devices, we have applied two steps as proposed by Kapitanova *et al.* (2012). First, the separation of the rule-base; in which, we have divided the inputs of the event detection process into two inputs rather than four. Each input is derived by another sub-event that takes two parameters as an input and generates one output. Second, the combination of rules, where this step involves combining rules with similar outcomes. Both steps are further explained with more details in Sub-Sec 3.6.3;

- **Defuzzification:** the implementation of rules located in the rule-base results in the obtaining several shapes that form the modified membership function. For instance, let's assume that a set of rules is applied to identify the possibility of having an unsuitable level of air quality as a result of the sudden increase of CO2 or PM25 levels. In this case, the output of the fuzzy inference engine might be as follows: Low(60 %), Medium (22%), and High(18%). The role of the defuzzification is to transform this set of percentages into a single crisp value that precisely describes the event of interest.

3.6 Edge Layer Components

In this section, a detailed description is presented of the functional modules forming the edge layer of our proposed approach. As shown in figure 3.4 these modules are configured to run in a collaborative way to accomplish real-time processing of sensory data.Further, they have been proposed taking into consideration the resources-restriction of IoT edge devices. For instance, since the training ML models are likely to be a resource greedy process, it has been by default assigned to the cloud, while the analysis module at the edge is running as a server that encompasses the pre-configured ML models and generates predictions in real-time. Also, several strategies have been followed to minimize the complexity of the FL rules processing.

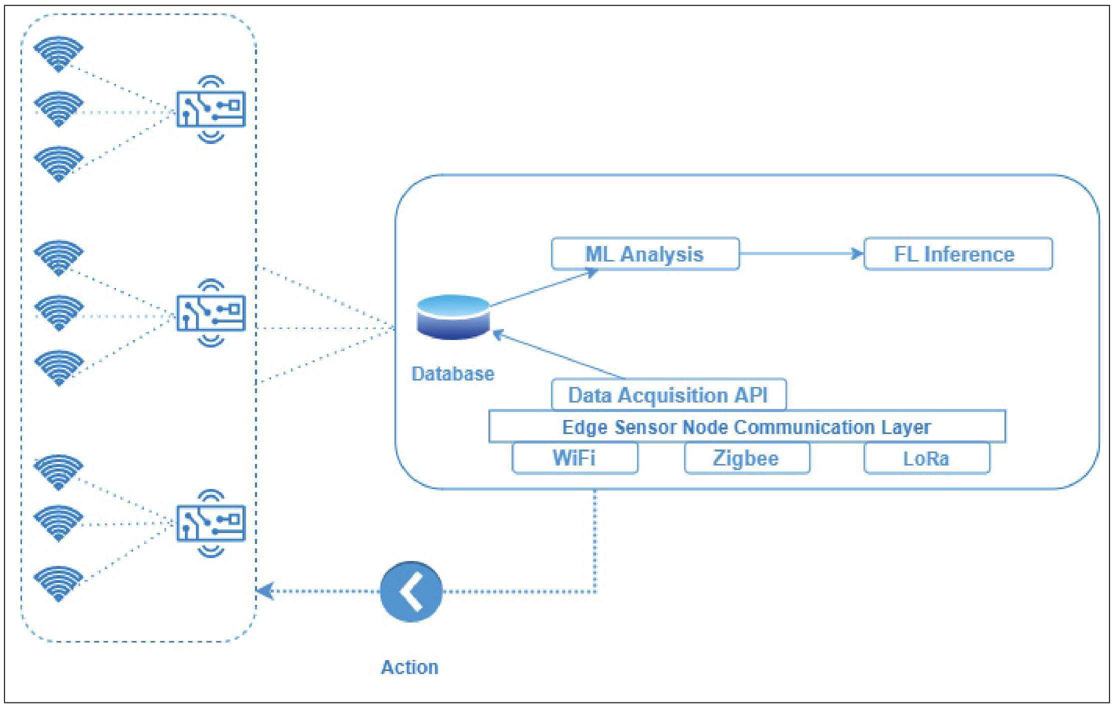


Figure 3.4 Edge Based Inference and Controlling System

In addition, several requirements have been considered to enable efficient on-the-fly data processing and handling of events on IoT edge devices. First, avoiding resource intensive tasks and enable only tasks that are compatible with the resources available on the edge device. Second, the ability to detect and identify abnormal events in advance. Third, the edge should be able to make decisions independently , and as quickly as possible using processing results. Finally, the edge device should be configured to send as little data as possible to the cloud. In order to meet these requirements, our edge proposal encompasses three main modules namely communication and and acquisition, ML based data analyzer, and FL based event detection. Further explanation is presented in the following subsections.

3.6.1 Communication Layer and Data Acquisition

In IoT applications, data are first collected through means of data acquisition APIs. These APIs are intended as an abstraction layer of the underlying infrastructure and to enable bidirectional communication channels between sensor nodes and the edge devices. As shown in figure 3.4, network protocols such as Zigbee, WiFi, and LoRA is conceivable to be implemented to bridge wireless communication Martínez *et al.* (2020). These protocols have been designed to perform under different environments and to enable communication within different ranges to serve a variety of IoT use cases. On the other hand, data acquisition is conceivable by several application protocols such as MQTT, and CoAP Ansari *et al.* (2018). In this work, an MQTT broker is used to send data and controlling instructions between our proposed edge solution and sensor nodes. MQTT is a lightweight protocol, and it has been widely adopted in different IoT applications.

Data are streamed in a time series sequence and locally stored on the edge for aggregation and analysis in further steps. Several time series databases (TSDB) such as Redis [1], and InfluxDB [2] can be used. The local storage is temporary and data will be transformed in later steps to the analysis for prediction. Also, data will be used to evaluate the model performance. Finally, to protect data privacy, they will be only transferred to the cloud upon user consent. This process is important to update the dataset stored in the cloud to be further used to create new models or retrain the current ones.

3.6.2 The Edge ML based Analysis and Prediction Module

Real-time analysis of IoT data is critical because the detection of patterns and anomalies in real time can have a constructive impact on event outcomes. In order to cope with this, this module is intended to serve the deployment process of ML models at the edge and close to data sources. In addition, this module features several sub-modules to evaluate and pre-process the incoming data, as well as, to monitor the performance of the implemented model.

[1] https://redis.io/

[2] https://www.influxdata.com/.

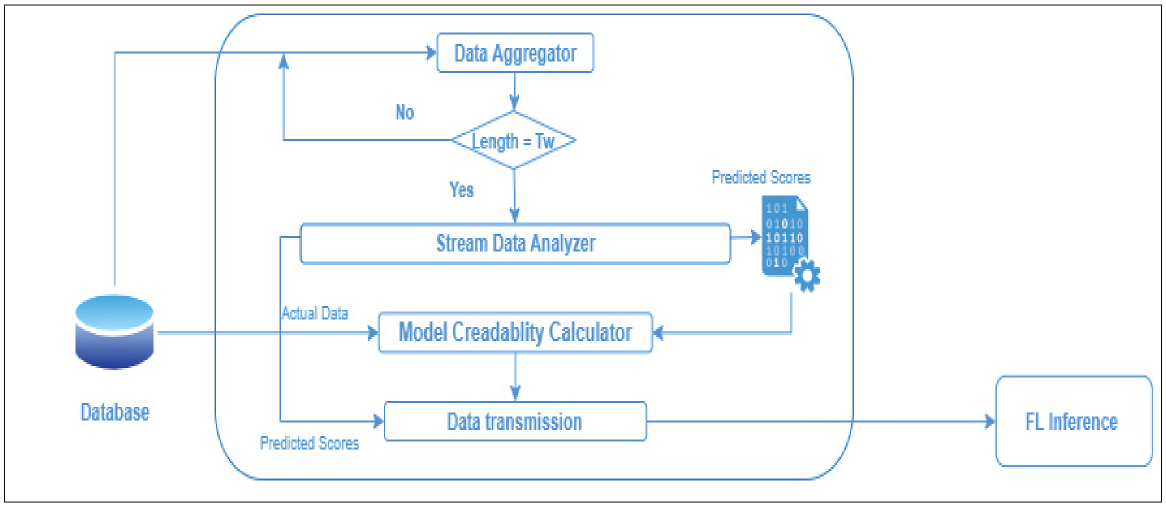


Figure 3.5 The Edge ML based Analysis and Prediction Module

3.6.2.1 Data Aggregator

This sub-module is intended to transform retrieved data packets to a format suitable to the used prediction model. Several feature engineering steps might be followed in this step. First, new features will be extracted from incoming data. For instance, from the time stamp it is possible to derive the day, hour, minutes, and weekend or not. Also, Exponential Moving Average (EMA) will be computed to generate the weighted average of the previous records. Higher weights will be assigned to the more recent values in the average calculation. Finally, the scaling step is applied to avoid biased model performance.

3.6.2.2 Stream Data Analyzer

This module is intended to run the deployed model to run predictions or classification tasks. It can serve one or multiple models based on services required to be exposed by the edge device. Most recent data are transmitted to the analyzer periodically and continuously from the data aggregator module on a vector size of the predefined time window (TW). The recursive forecasting strategy Bontempi *et al.* (2012) is employed to predict multiple next values that represent the required prediction horizon. In this strategy, sensor data are collected in a queue of constant length that corresponds to the length of the chosen time window. The model is configured to perform one-step-ahead forecasting at each forecasting step in the prediction horizon. At every prediction step in the prediction horizon, the newest obtained predicted value is being added to the sequence concurrently with the removal of the oldest value in the sequence as depicted in figure 3.6. The main shortcoming of this strategy is the accumulated prediction error with each step that is likely to increase with longer time horizon. This arises because data predicted from earlier steps is used to predict the value of the current step. Prediction scores generated by this module are stored in the log file as depicted in figure 3.5. The recursive strategy has been selected since it demands fewer computing resources compared to the direct prediction strategy Taieb *et al.* (2012). To alleviate the impact of the prediction error on the decision making process, we have proposed the model credibility concept.

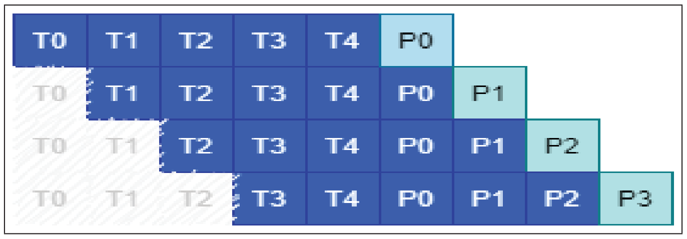


Figure 3.6   Recursive multi-step prediction

3.6.2.3   Model Credibility Calculator

We define the ‘model credibility’, in the context of the proposed approach, as the degree of the reliability in the model performance, so it can be used later as an input to support identifying the action recommended to respond to a detected event. In our approach, R^2 is used as an indicator of the model’s credibility. Compared to RMSE, which is not scaled to any particular values, R^2 is conveniently interpreted and scaled between 0 and 1. This significant, especially in the referenced use case, which carries out the prediction of four different parameters that come in different scales.

R^2 is computed as an iterative procedure during the prediction process upon the availability of the actual data that corresponds to data predicted priorly within a specific prediction horizon.

This process can be configured to run periodically, for instance hourly or daily. However, to ensure a better impact of the model's credibility on the generated recommended action, it would be preferable to run this process in a shorter period. For instance, if the ML model is set to generate predictions for a prediction horizon of 30 minutes, it would be more reasonable to compute model credibility hourly, and use this value as input for the FL for the following hour. Moreover, model credibility can be used as an indicator of the degradation in model performance. This can be done by setting a credibility threshold and track times the model credibility exposes values under the predefined threshold. Moreover, tracking model credibility in real time will enable launching retraining procedures that better reflect the status of the model performance. Furthermore, model credibility is likely to reduce data transfer operations by sending only data that impacted the model performance to be used later for model retraining.

3.6.3 The Edge Fuzzy Logic Based Event Detector and Controlling Module

This module serves our approach as an event detector and promotes decision making functionalities on IoT edge-based solutions. This is significant, especially when addressing a use case such as the one considered in this work. More specifically, real-time event detection would empower dwellings by the ability to behave in advance to anticipated events. To accomplish this, we have adopted the fuzzy logic concept to identify events from predicted sensory data received from the previous module and generate recommended actions that can be either applied by users or configured to run autonomously. The latter case highlights the importance of deploying such a mechanism on the edge, where it is likely to provide a faster response time. Since the predicted sensory data is imprecise in nature, fuzzy logic is more favored than crisp logic to handle this data. As shown in figure 3.7 two compatible sub-modules that is composed of independent FL inference engines have been proposed namely: event and action detector.

3.6.3.1 Event Detector

This step aims to generate events based on predicted monitoring values. The Event Detector sub module deduces events of interest using a set of fuzzy inference rules. The premise consists of

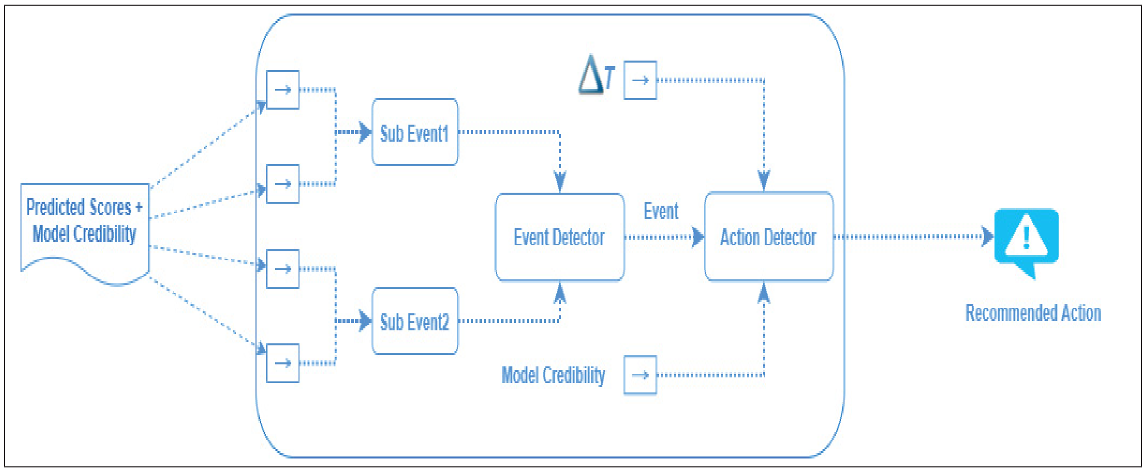


Figure 3.7 The Edge Fuzzy Logic Based Event Detector and Controlling Module

fuzzy input variables (e.g. Temperature and Humidity) joined by logical operators (e.g. AND and OR). The consequent represents the event and its importance as fuzzy output variables. In the following explanation, more concentration will be detected on the use case presented in Sec 3.3. Thus, the input variables of the event detection controller are Temperature, Humidity, CO2, and PM2.5. Consider the following rule form to detect an event from a set of recorded observations:

IF Co2 is High and PM25 is Medium and Temp is High and Humid is Low Then Event is Alert.

In this rule, the degree of Co2 *'High'* is derived by a membership function that defines the range of possible values of Co2. Similarly for the other parameters as shown in figure 3.9 which presents the membership function of the different linguistic variables used by the event detection controller.

Taking into consideration the four parameters are involved in forming the event detection rule, and each one of them might be assigned four possible linguistic values namely: Low (L), Medium (M), High (H), and Very High(VH). To create an event rule base, it is essential to exhaustively include all possible value-combinations for the input linguistic variables. Theoretically, the number of elementary rules can be about $4^4 = 256$ (the number of possible fuzzy sets to the

Table 3.2 Event Detection Rule Base

L = Low M = Medium H = High VH = Very High
N = Normal W = Warning A = Alert

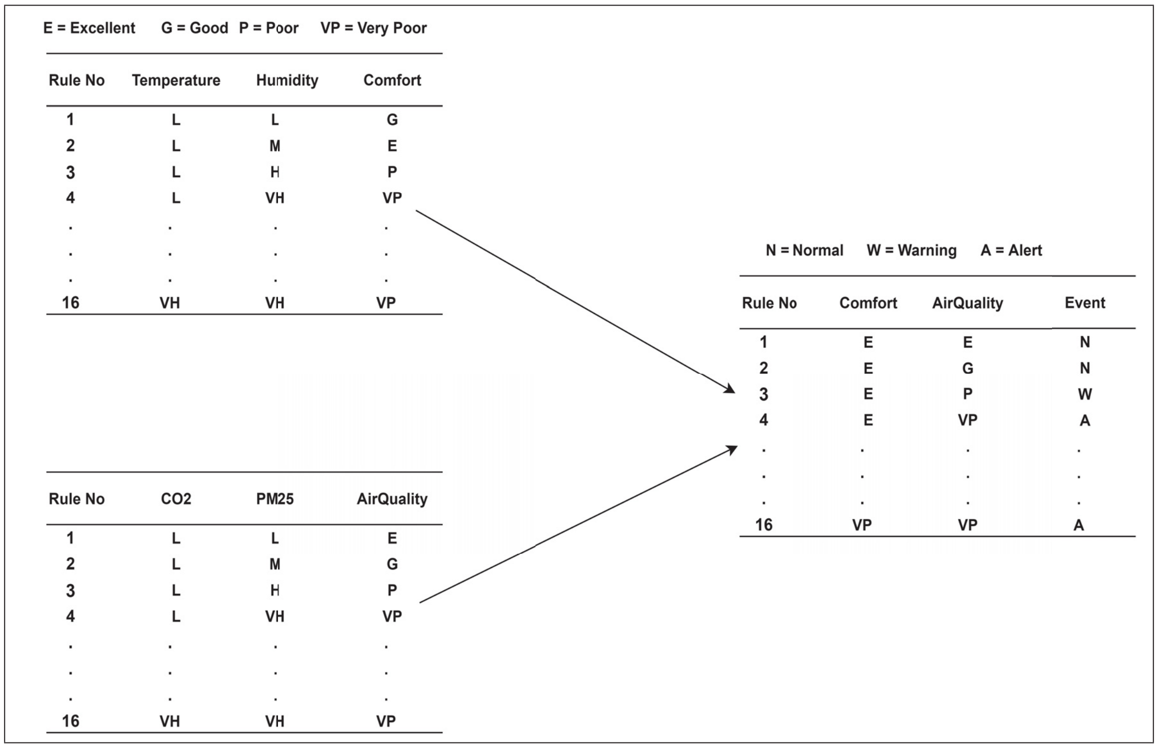
Rule No	Temperature	Humidity	CO2	PM25	Event
1	L	L	L	L	N
2	L	L	L	M	N
3	L	L	M	H	W
4	L	L	M	VH	A
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
254	VH	VH	H	M	A
255	VH	VH	VH	H	A
256	VH	VH	VH	VH	A

power of a number of inputs) as shown in Table 3.2. Adding and learning rules rapidly may be difficult for resource-constrained IoT edge devices. In addition, the rule-base will need to be traversed each time a new inference process is preformed Kapitanova *et al.* (2012). To reduce this complexity of proposing and managing a large number of fuzzy rules, we propose the following points:

1) Splitting the event detector into two sub-systems as presented in figure 3.7. SubEvent1 and SubEvent2 for generating comfort events and air quality events respectively. The Event detector, by its role, combines both subevents and deduces the event accordingly. By doing this, each sub-system uses separately its own rules required to reveal the event behind its input. Therefore, the number of rules has been reduced to become in a total of 48 rules, divided to (4^2), for each of the SubEvents (Event1, and Event2), as well as (4^2) for the EventDetector as presented in Figure 3.8.

2) For further optimization, we combine rules that produce similar outcomes. The simplest way is to put an 'OR' logical operator between the premises of the combined rules. For example:

   IF Co2 is High and PM25 is Low Then AirQuality is Poor.



E = Excellent G = Good P = Poor VP = Very Poor

Rule No	Temperature	Humidity	Comfort
1	L	L	G
2	L	M	E
3	L	H	P
4	L	VH	VP
.	.	.	.
.	.	.	.
.	.	.	.
16	VH	VH	VP

Rule No	CO2	PM25	AirQuality
1	L	L	E
2	L	M	G
3	L	H	P
4	L	VH	VP
.	.	.	.
.	.	.	.
.	.	.	.
16	VH	VH	VP

N = Normal W = Warning A = Alert

Rule No	Comfort	AirQuality	Event
1	E	E	N
2	E	G	N
3	E	P	W
4	E	VP	A
.	.	.	.
.	.	.	.
.	.	.	.
16	VP	VP	A

Figure 3.8 Separation of Event Detection Rule Base

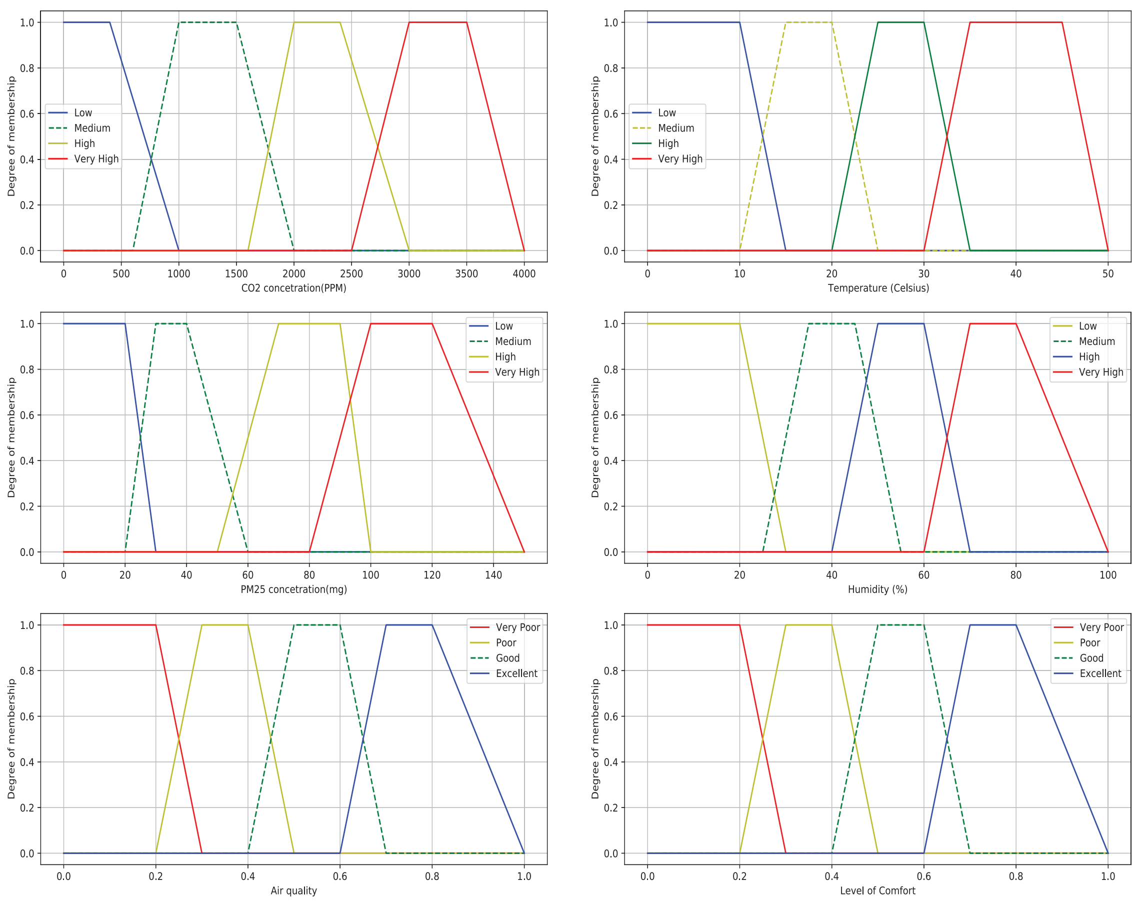
IF Co2 is High and PM25 is Medium Then AirQuality is Poor.

The combination of these two rules is as follows:

IF (Co2 is High and PM25 is Low) Or IF (Co2 is High and PM25 is Medium) Then AirQuality is Poor.

3.6.3.2 Action Detector

In order to obtain the decision required to proactively act for a certain situation, first, we need to identify the event of interest before occurring, and the period of time that it is potentially to persist, which is defined as Δ T. First, the event of interest is identified from predicted sensory data. These data are likely to be fluctuating and imprecise; thus, we proposed the concept of model credibility, which calculated in previous steps as a parameter to determine the recommended course of action as shown in figure 3.7. The membership function of the model credibility takes three values namely: low, medium, and high. Second, Δ T, which refers to the



a) Air Quality Membership Function b) Comfort Level Membership Function

Figure 3.9 Fuzzy Logic Memberships for the observed parameters and their outputs

temporal aspect of an event. It records the period during which a particular event continued to take place. Mainly, ΔT membership function takes three linguistic values, namely: short, medium, and long as shown in figure 3.10. The influence of ΔT on the obtained decision is significant since it represents the period in which a pattern from predicted sensor data constitutes an event of interest.

Taking into consideration the use case addressed in this work, a fan speed controller is envisioned as the receiver of the generated action. Thus, the decision (recommended action) takes four linguistic values as a fuzzy output. First: No change (NC), which means that the prospected status of air quality and comfort does not require any further action to manipulate fan speed. Or,

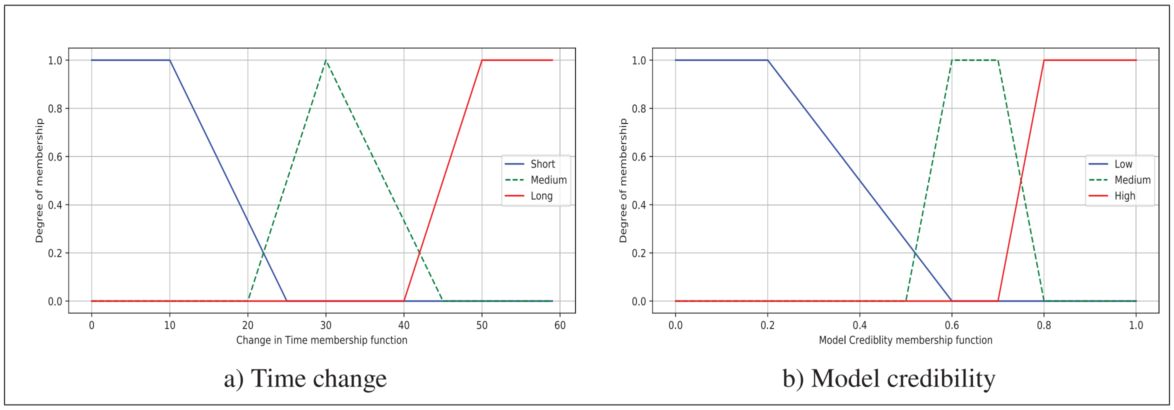


Figure 3.10 Fuzzy Logic Memberships for Time change and Model credibility

for instance, it might result from low model credibility input, so it is preferable to keep to the fan mode as it is. In addition, Slow (S), Moderate (M), and Fast (F) linguistic values have been introduced to identify the recommended action. Table 3.3 shows an example of possible rules combination used to generate the action.

Table 3.3 Recommended Action Rule Base

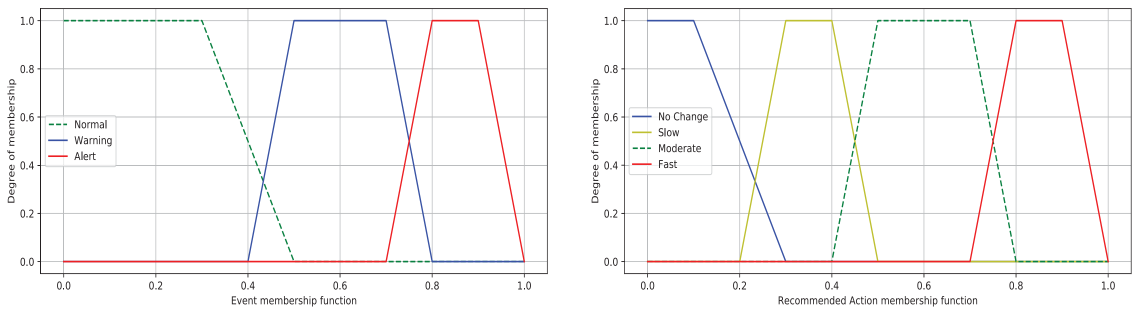
NC = No Change S = Slow M = Moderate F = Fast

Rule No	Event	Model Credibility	ΔT	Action
1	N	L	S	NC
2	N	L	M	S
3	N	L	L	M
4	N	M	S	M
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
27	A	H	L	F

3.7 Experiments

Several experiments were carried out using a time series dataset. In particular, Gams dataset [3]is used . It is an indoor air quality dataset that includes indoor (PM10, PM2.5, humidity, temperature, CO2, noise, and VOCs) environment sensor values. Table 3.4 provides some

[3] https://github.com/twairball/gams-dataset



a) Event Detection b) Recommended Action

Figure 3.11 Fuzzy Logic Memberships for event and recommended action

details about sensors and their measurement range. These data were sampled on a one-minute resolution during the period from Nov 21, 2016, to March 30, 2017, and it contains 135000 samples of a variety of sensors' values.

Table 3.4 Specifications of sensors used to collect Gams dataset

Type	Sensor name	Unit	Measurement Range	Accuracy
Temperature	SHT30 / SHT20	°C	0 - 50	+/- 0.5°C
Humidity	SHT30 / SHT20	%	0 - 100	+/- 3.0 %
Carbon Dioxide (CO2)	Telaire T6713	ppm	0 - 5000	+/- 40ppm
Particulate Matter (PM2.5)	Proprietary design	µg/m3	0 - 1000	+/- 10 %
Particulate Matter (PM10)	Proprietary design	µg/m3	0 - 1000	+/- 10 %
Volatile Organic Compounds (VOC)	iAQ-core	µg/m3	0 - 4.0	+/- .01 mg

In our experiments, we used data collected in the period from Feb 01, 2017, to Feb 28, 2017, for offline model training, testing, and evaluation. This includes 40220 samples composed of different sensors' readings. The distribution information of the train data set is summarized in Table 3.5. While data recorded in the period from March 01, 2017, to March 07, 2017, have been used to assess the performance of the models chosen to run in real-time scenarios on the edge device. This selection of data range can be justified by the stability in the sensory data collection process during the mentioned period compared to the prior months. More specifically, fluctuations and instability due to some failures of sensors' functionalities lead to some gaps in the sampling interval. This gap ranges from minutes to hours and sometimes days. This in turn breaks the sequential recording of sensor readings and most importantly negatively impacts the performance of the ML model.

Table 3.5 Data distribution information of indoor environmental factors

	CO2	Humidity	Temperature	PM25	PM10	VOC
Count	40220	40220	40220	40220	4220	4220
Mean	704.18	38.89	23.06	17.03	19.32	0.14
STD	388.23	4.41	2.17	12.61	14.01	0.10
Min	385	28.83	17.71	1.80	2.00	0.062
Max	1903	44.05	27.50	72.20	142.60	1.33

In our experiments, we used data collected in the period from Feb 01, 2017, to Feb 28, 2017, for offline model training, testing, and evaluation. This includes 40220 samples composed of different sensors' readings. The distribution information of the train data set is summarized in Table 3.5. While data recorded in the period from March 01, 2017, to March 07, 2017, have been used to assess the performance of the models chosen to run in real-time scenarios on the edge device. This selection of data range can be justified by the stability in the sensory data collection process during the mentioned period compared to the prior months. More specifically, fluctuations and instability due to some failures of sensors' functionalities lead to some gaps in the sampling interval. This gap ranges from minutes to hours and sometimes days. This in turn breaks the sequential recording of sensor readings and most importantly negatively impacts the performance of the ML model.

In general, three sets of experiments have been carried out in this work namely: Data exploration and visualization, and machine learning models creation and evaluation. In addition, the evaluation of the edge-based ML analyzer and FL inference modules in real time settings, as well as, edge resource utilization.

3.7.1 Data Exploration and Visualization

This explanatory set of experiments is intended to reveal trends and correlations that characterize samples collected from different sensors. This leads to the extraction of more valuable features

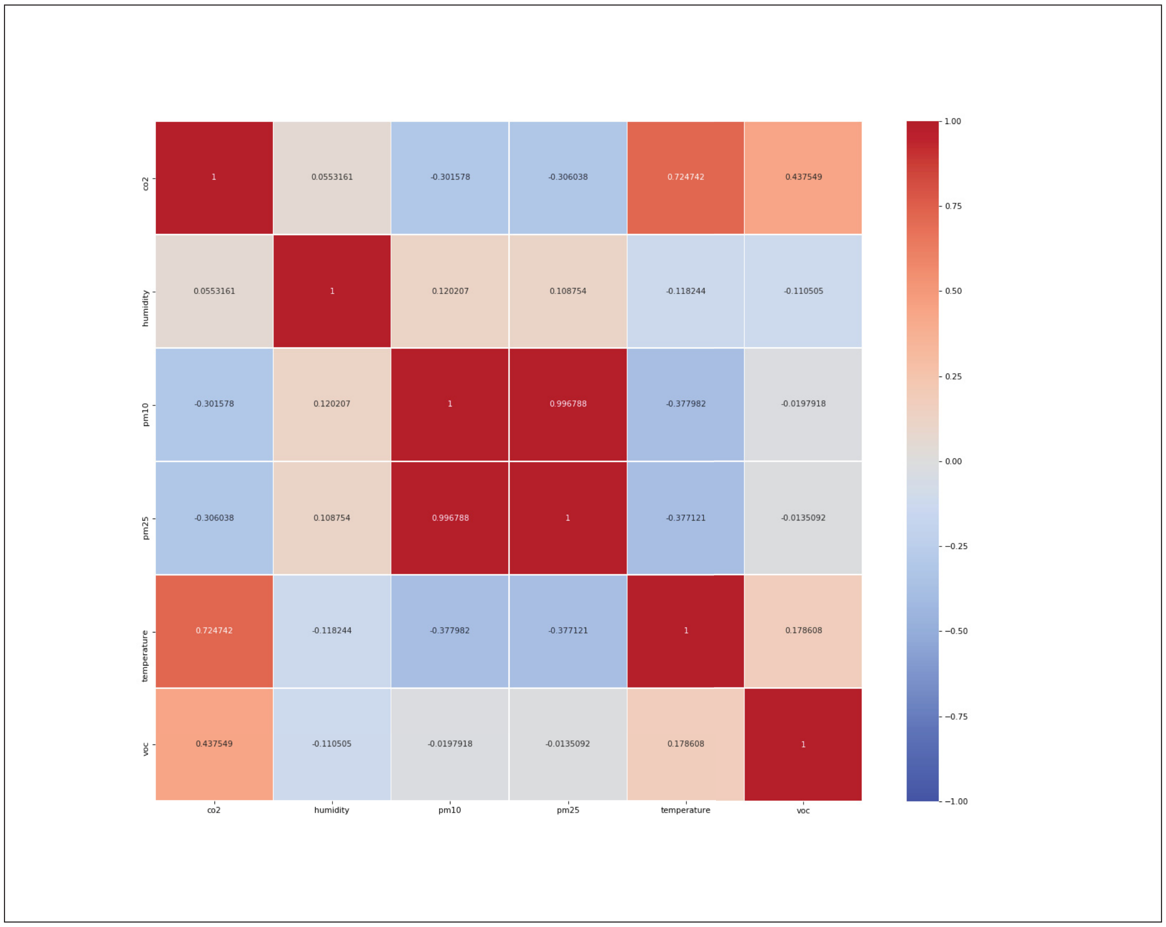
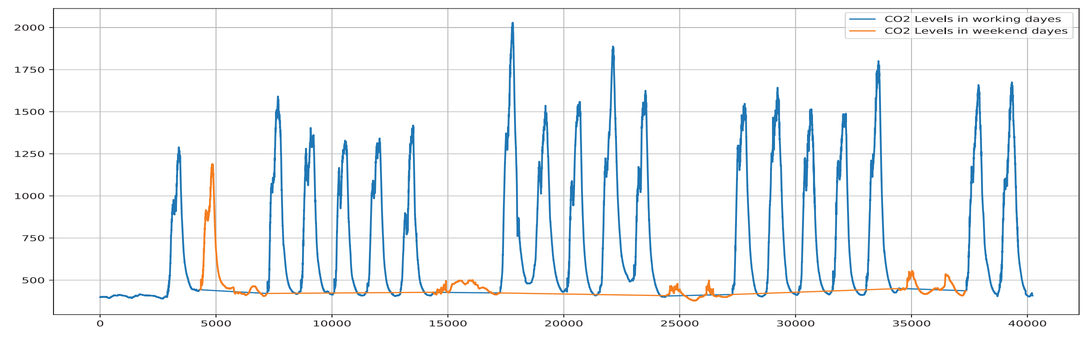


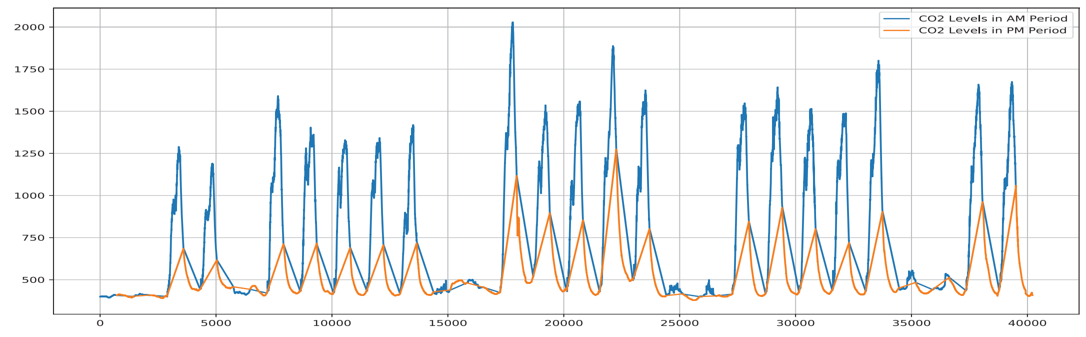
Figure 3.12 HeatMap Features Correlation

that can be used to tune the performance of the predictive model. Heatmap-based visualization technique is used to analyze the correlation between individual time-series data sources used in our experiments as shown in figure 3.12. This correlation ranges from 0 to 1 as an indicator of strong/weak positive correlation or no correlation. While negative figures refer to strong/weak inverse correlation between data sources.

For instance, there is a strong positive correlation between PM10 and PM2.5 which implies that any change in either one of them will probably lead to affect the other. Thus, it is possible to use them in combination to train a predictive model as independent features. Also, it can be observed that there is a relatively strong correlation between CO2 and temperature. So, the temperature can be used to predict CO2 levels and vice versa. In this case, scaling is required to assign equal weights to features values and to avoid unbiased performance. Also, a positive correlation is observed between CO2 and VOC samples. On the other hand, a negative correlation is observed between temperature, CO2, and both PM10 and PM2.5, which is coordinated with results obtained by Ameer *et al.* (2019).



a) Weekdays



b) Day periods

Figure 3.13 Distribution of CO2 readings

In this work, more concentration is dedicated to the prediction of CO2, PM2.5, temperature, and humidity levels. While the first two factors are strongly related to the indoor air quality, the other two are strongly related to the comfort situation for inhabitants. Therefore, the development of four ML predictive models have been investigated in this work. An important step is to analyze the variation in the recorded samples of the aforementioned factors during weekly cycles. More specifically, this step is intended to reveal trends in data distribution in accordance with time or weekdays. For instance, CO2 readings have shown a significant decrease in weekends compared to weekdays as shown in figure 3.13. This emphasizes the use of the day_of_week as

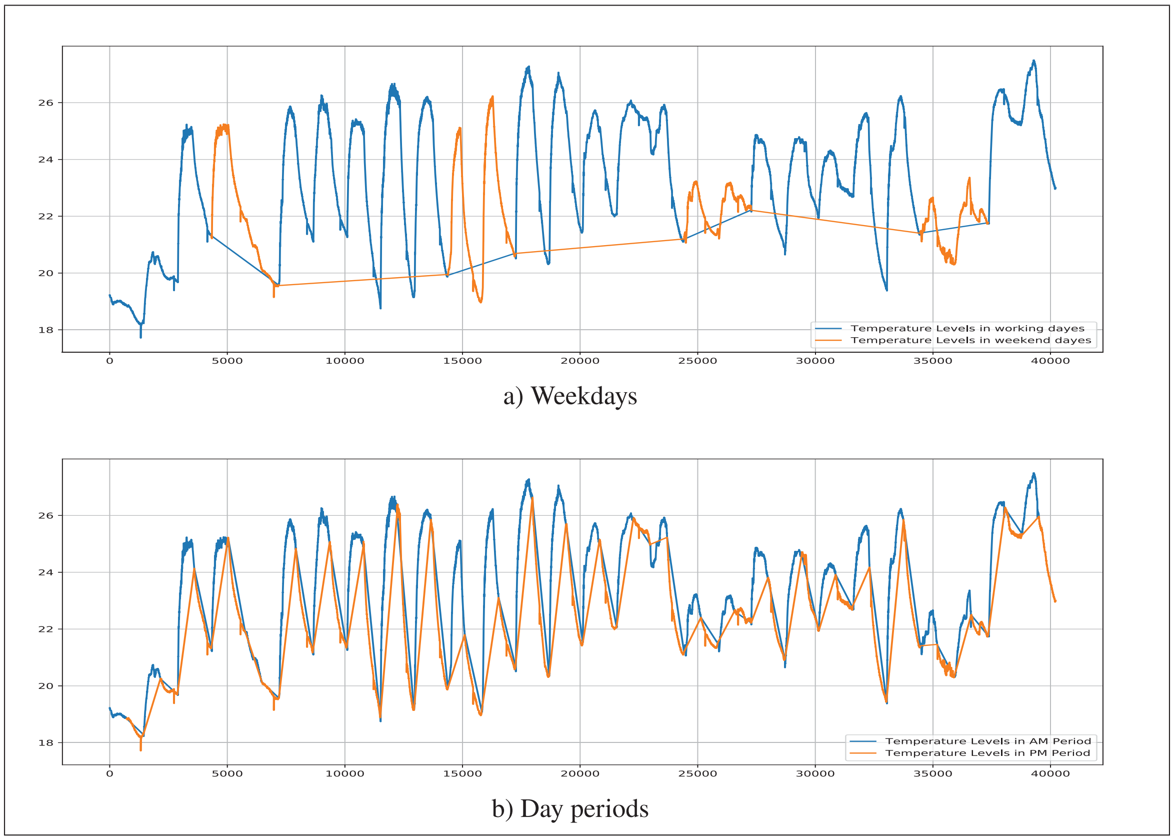
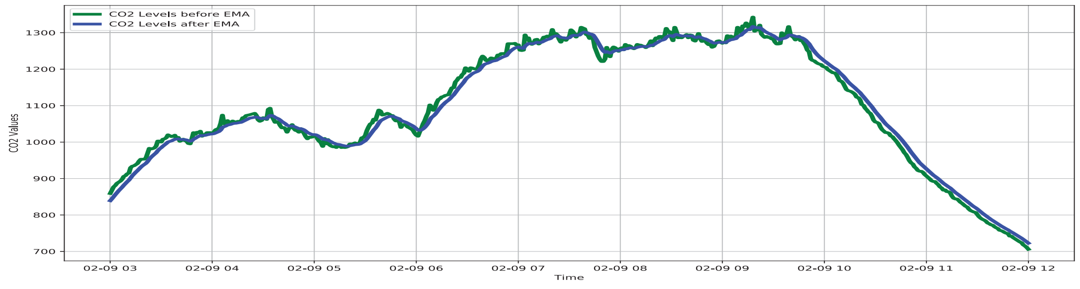


Figure 3.14 Distribution of Temperature readings

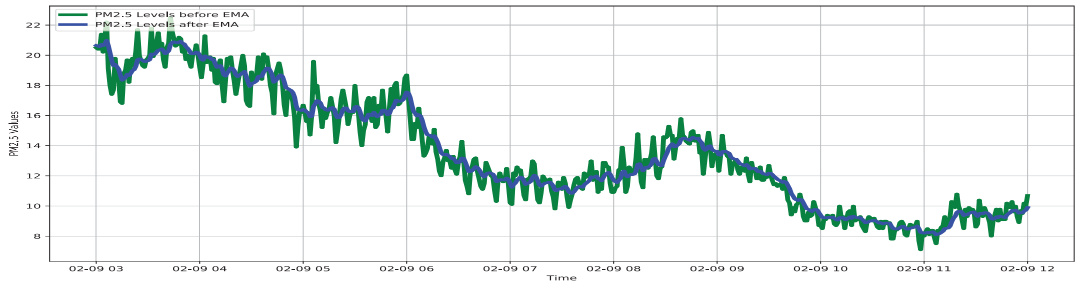
an extracted feature from ‘date’ while predicting CO2 levels. This feature has been assigned the value 1 for weekend days and 0 for the remaining weekdays.

Moreover, some factors such as CO2 and temperature have shown some variation in their readings in association with the period of the day ’AM’ or ’PM’. This is as shown in figures 3.13,3.14; similarly, ’AM’ or ’PM’ is extracted and added as a feature of value 0 or 1 to enhance models prediction efficiency.

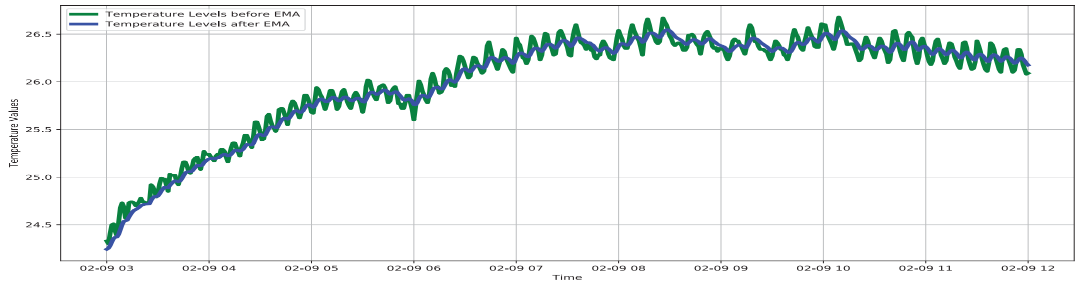
In addition, the exponential moving averages (EMA) have been calculated for each environmental factor as shown in figure 3.15. The EMA has been adopted as it is likely to contribute to the enhancement in model accuracy. This due to its ability to alleviate the impact of some operational issues incurred during the sampling stage such as fluctuations in some environmental conditions and sensors’ drift.



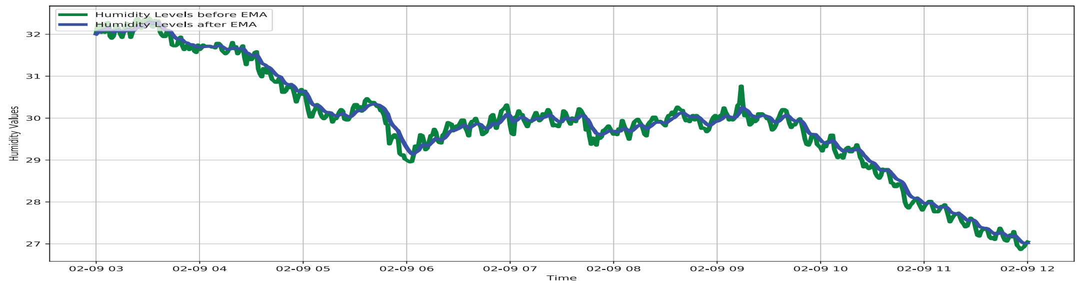
a) CO2 Levels with EMA



b) PM2.5 Levels with EMA



c) Temperature Levels with EMA



d) Humidity Levels with EMA

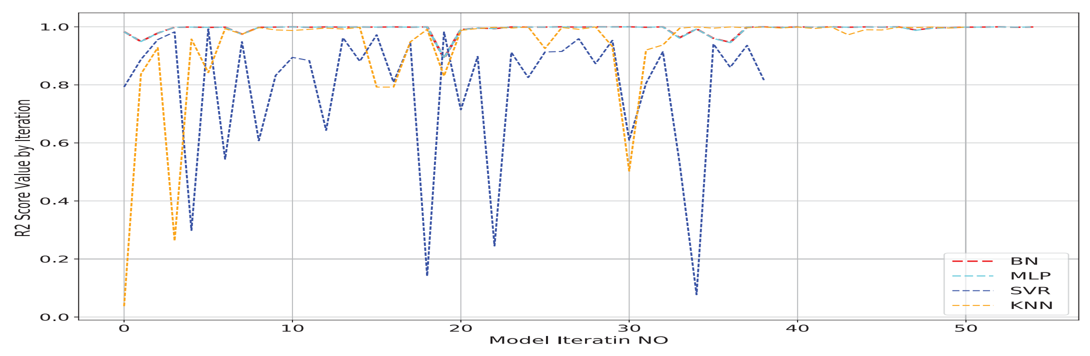
Figure 3.15 Data Distribution with EMA

3.7.2 Machine Learning Models Creation and Evaluation

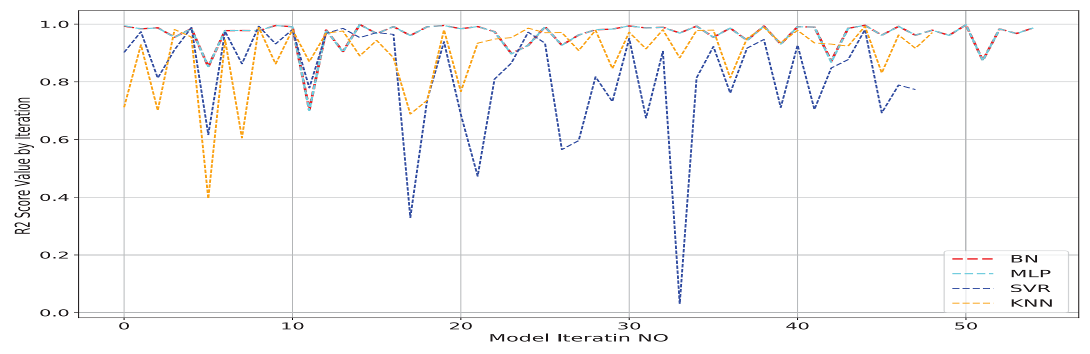
In this section some experiments are carried out based on a series of comparative analyses on ML models and their capabilities and limitations, some conclusions regarding the model selection and improvement are drawn. These experiments have been conducted to guide the selection process of ML models to be further used in real-time. This selection is based on a better accuracy score and the overall execution time during the training and evaluation stage. As previously mentioned four ML algorithms are under consideration in this work namely: KNN, SVR, MLP, and BN. These models have been created and implemented under the python 3 environment and using Scikit-learn, Pandas, NumPy, and Matplotlib packages.

The development process followed in this stage is stimulated from real-time settings. Where our models and their parameters are initially trained using data from the first two days of the training dataset. These data are time series and they are sequentially ordered. Afterward, these parameters are applied to data within the next 6 hours time window, and evaluation criteria are computed and recorded. The next step is to update the train data set by adding the 6 hours evaluation data and retrain and update model parameters in a continuous process. Similarly, the evaluation window will be shifted forward in a fixed walk-forward time window and updated model parameters are used to make predictions on these unseen data. This iterative process has been configured to run till the loop reaches the end of the training dataset. Thus, it has resulted in the creation of a set of trained models. Each model is represented by iteration no in the presented figures.

Figures. 3.16,3.17 depicts the variations of the values of the recorded R^2 score. Obviously, the BN, and MLP have converged to better accuracy score corresponding to the model iteration no. Additionally, KNN and SVR have exposed poor performance compared to BN, and MLP. This is clearly spotted from the low level of recorded R^2 score, in addition to negative values they generated for the R^2 score that reflected in the number of missing iteration as shown in the mentioned Figures.



a) CO2

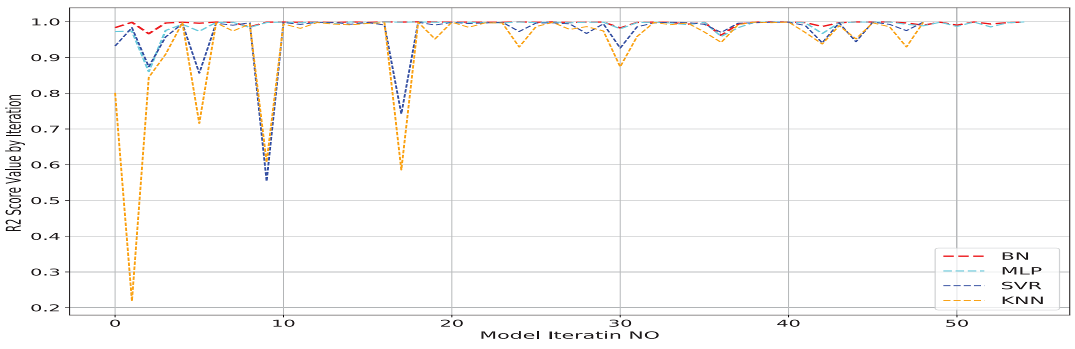


b) PM2.5

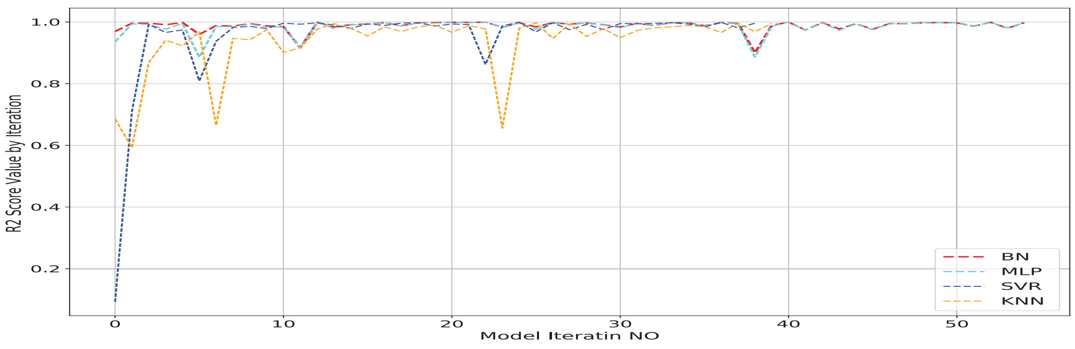
Figure 3.16 R scored value by Training Iteration

Finally, figure. 3.18 shows the overall execution time for each model and it is organized by the predicted parameter. It is observable that BN has performed with the lowest execution time among all investigated models with all the environmental factors. This indeed contributes to the scalability of the model, taking into consideration that in IoT data is accumulated periodically, which enlarges the size of the used dataset. On the other hand, KNN has shown the highest execution time. This is likely due to computing the value of the K parameter prior to the model training and evaluation in every training iteration. In addition, although MLP provided a satisfactory accuracy score, as explained above, it has generated a higher execution time compared to BN and SVR.

To sum up, in all aforementioned experiments, BN has overpassed the rest of the models in terms of evaluation criteria as well as has proven its scalability when compared to the other



a) Temperature



b) Humidity

Figure 3.17 R scored value by Training Iteration

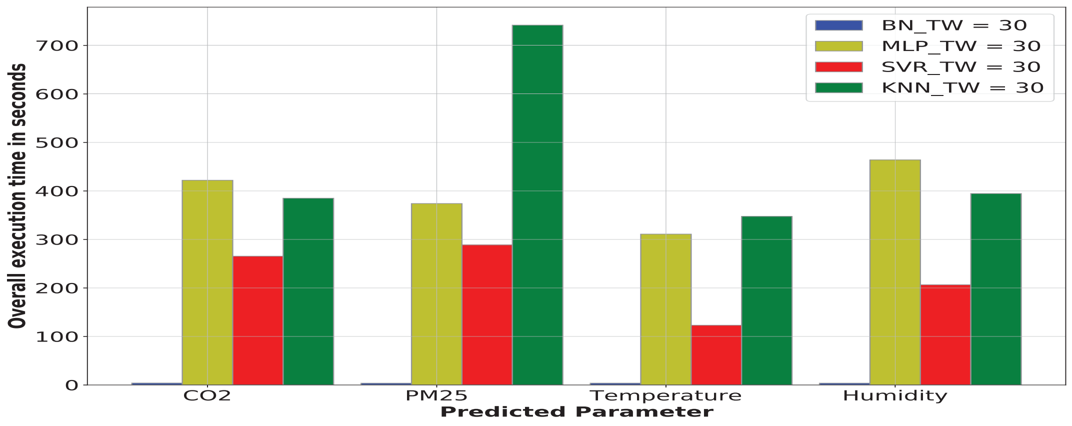


Figure 3.18 Overall Execution time

models in terms of execution time. Also, MLP has shown a reasonable performance taking into consideration its ability to generate acceptable prediction scores compared to KNN and

SVR. However, MLP has shown a high value of execution time compared to BN. Thus, in the following experiments, only BN will be considered.

3.7.3 Evaluation of the Edge Modules

The main process run on the edge has been derived from modules designed for our proposed approach. As depicted in figure. 3.19, the main process receives and collects data from virtual sensors reading from the dataset. Afterward, data are analyzed by the analyzer component and delivered to the FL inference module to detect events and derive the recommended decision. Raspberry pi3 [4] has been chosen as a hardware platform to serve the proposed modules on the edge. As a single-board computer, Raspberry pi 3 is featured by its small size, reasonable price, and its relatively high processing capabilities compared to other boards and microcontrollers. It comes equipped with a processor speed of 1.4 GHz, 1GB of RAM, an embedded Wi-Fi, and an Ethernet port. Python 3.7 was selected as a programming language to develop all the functionalities of the main process unit.

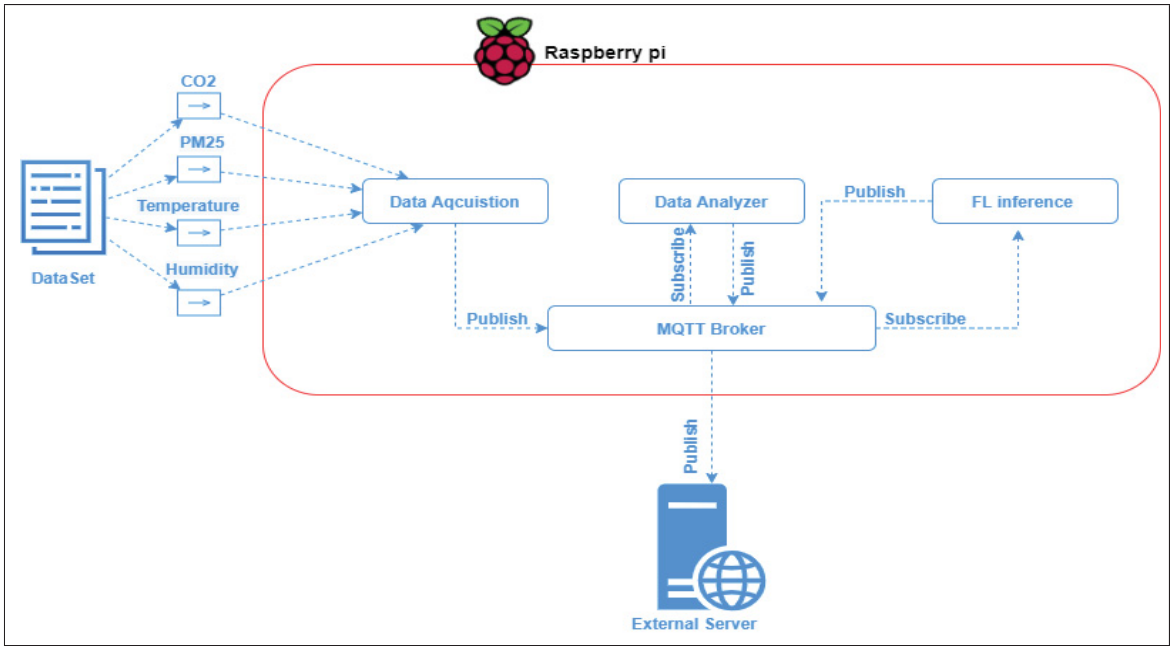


Figure 3.19 The Edge Experimentation Setup

[4] https://www.raspberrypi.org/.

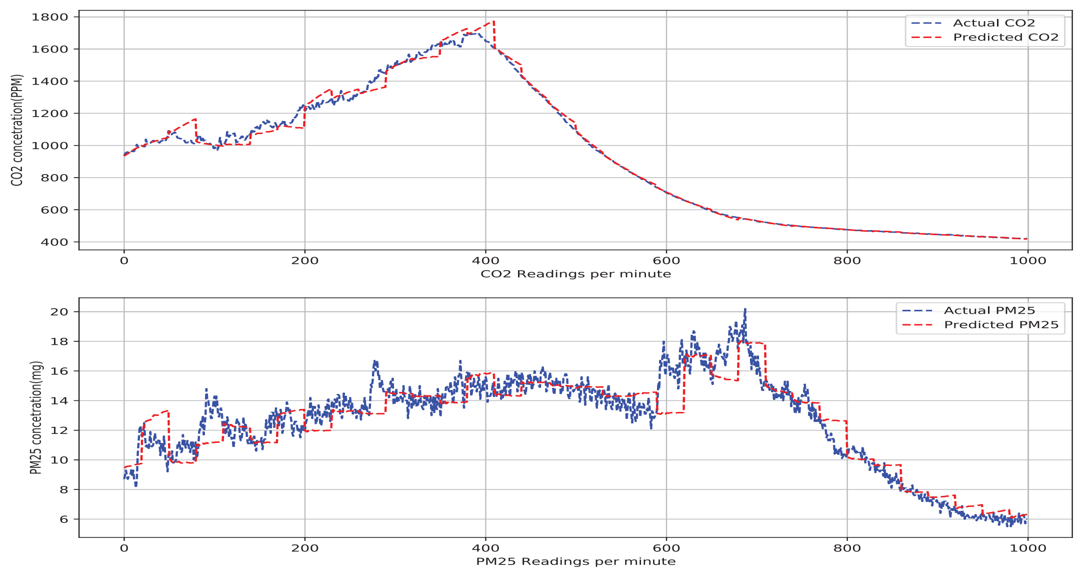
In addition, Message Queue Telemetry Transport (MQTT) Hillar (2017) has been used as the communication protocol in our experimental setup. MQTT is a lightweight publish-subscribe model-based protocol designed on top of the TCP/IP stack. Mainly, it is composed of two components, namely MQTT broker and MQTT clients. MQTT broker is implemented to communicate messages between MQTT clients such as publishers and subscribers. This is accomplished through dynamically created and uniquely defined MQTT topics over the broker. The implemented software was divided into the following four procedures:

- data acquisition module reads sensory data from the dataset on 5 secondes interval time and publishes them on the MQTT broker through a predefined topic;
- data analyzer has two MQTT clients, namely publisher, and subscriber. While the latter is intended to communicate with the data acquisition module to collect data for the analyzer, the former publishes the predicted data to the broker to be later fed to the FL inference module;
- similarly, the FL inference module is served by two MQTT clients. The subscriber, which reads from the topic created by the data analyzer module. While the publisher creates a topic to be later used to transfer the output to an external server through the MQTT broker;
- MQTT uses 'localhost' as a broker to establish communication between edge components. While 'mosquitto'[5] is used as a public broker to communicate with the external server, which in our case is a ubuntu virtual machine deployed on AWS.

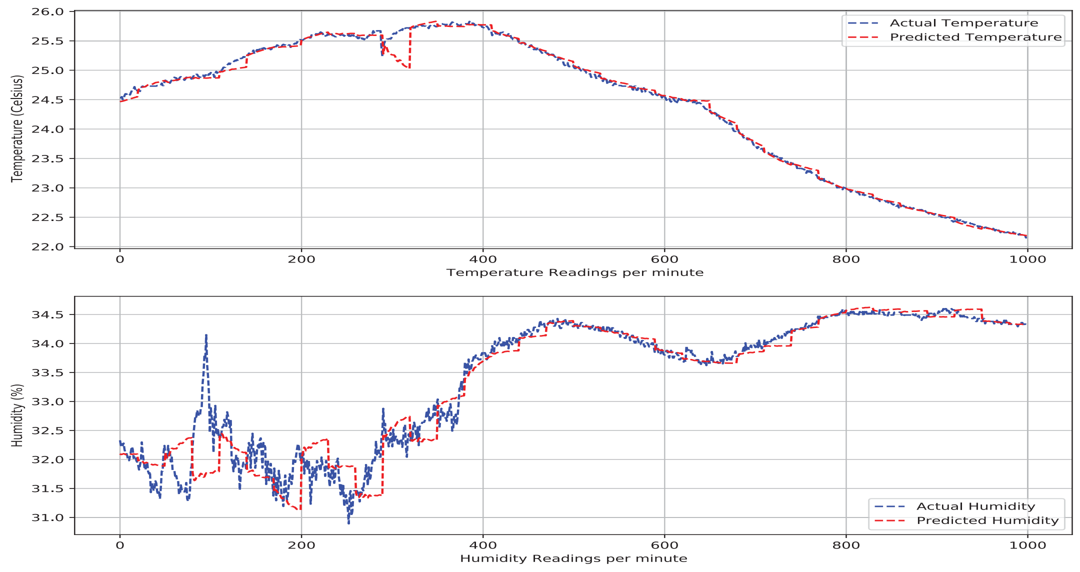
3.7.3.1 Model Accuracy in Real Time Prediction

ML models created in previous steps have been deployed in 'pickle' format on the edge device and configured to run real-time prediction tasks. To do this, we have streamed data packets received from simulated sensors to the prediction and analysis module. This module is equipped with four BN models; each one of them is applied to generate predictions for a particular parameter (e.g, co2,pm25, temperature, and humidity). Received data are collected for each

[5] https://test.mosquitto.org.



a) CO2 and PM25 predictions



b) Temperature and Humidity predictions

Figure 3.20 Models Accuracy in real time settings

parameter and, models are set to start generating predictions when the length of vector V = 30, which is the time window used during model training. Also, models are set to generate predictions for the horizon of the next 30 minutes.

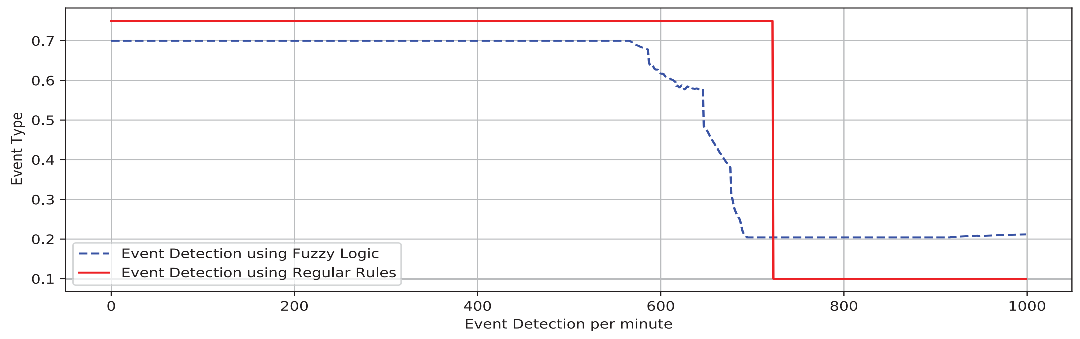
Figures 3.20 shows the prediction scores in the dashed red line and the actual values in the dashed blue line. As it can be seen, the performance of ML models fluctuates with time, but in most cases, the models have performed well and generated predictions that do not diverge significantly from the actual data. This will likely have a positive impact on the performance of the implemented FL inference engine and its output. However, this point, in particular, requires more investigation to ensure the stability of the model performance since its impact on the generated decision is profound.

3.7.3.2 Fuzzy Logic Evaluation

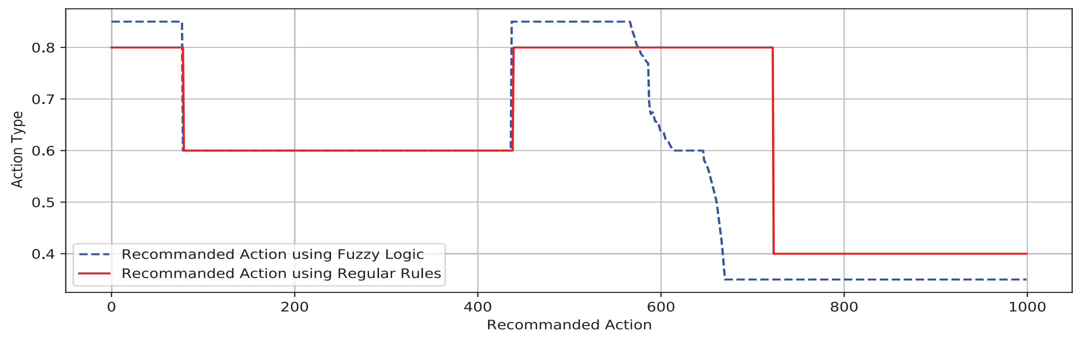
This category of experiments is intended to reveal the ability of our fuzzy logic rules in identifying events of interest and producing the appropriate action. First, the scikit-fuzzy api [6] have been used to implement the proposed edge-based FL inference engine. Second, a customized rule engine have been developed that runs regular IF rules with inputs similar to the inputs used for running the FL. Figure. 3.21 shows the impact of using the FL rules to detect events compared to regular rules in the context of the presented use case. More specifically, We have compared the behavior of both sub-models air quality and comfort. In addition, their perspective impacts the detected event and corresponding recommended decision using FL and regular rules.

As depicted in figure. 3.22, air quality is derived from Co2, PM2.5, while he comfort level is derived from temperature and humidity. Also, it can be observed that the FL rules track the fluctuation and uncertainty resulted from the predicted data and their impact on identifying both cases with more confidence. For instance, FL rules respond more accurately in identifying the air quality level at episode no ‘400’ on the chart, which corresponds to the increase in the Co2 levels shown at the same episode in figure. 3.20. Furthermore, while regular rules shift the level of both the comfort and air quality in a sharp pattern, the FL rules track both and apply changes in a gradual manner that is better reflects the gradual change in the values generating both conditions.

[6] https://pythonhosted.org/scikit-fuzzy/.



a) Event detection derived from Air Quality and Comfort Levels



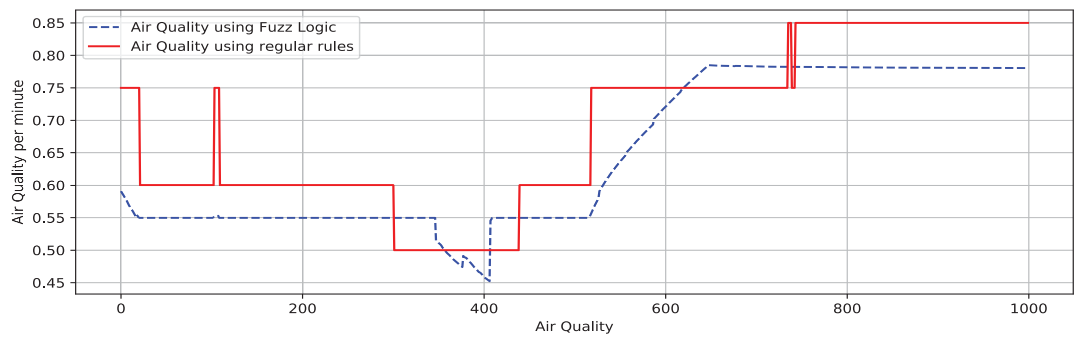
b) Recommended Action

Figure 3.21 Event Detection and Recommended Action

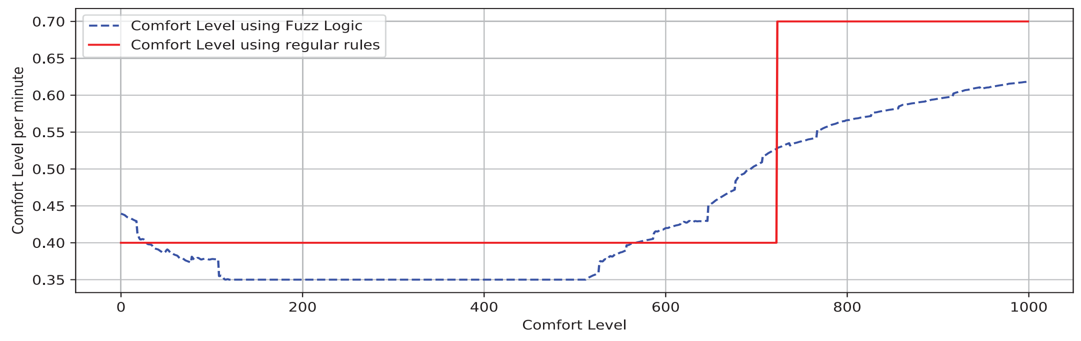
A similar observations can be spotted in figure. 3.21. Where, while the FL rules gradually decrease the level of the event and consequently the corresponding recommended action. The regular rules sharply apply these changes. In an actual use-case scenario, this would be reflected on the cost considering the operation of a fan and keeping its speed on the Fast mode longer than needed as applied by the regular rules.

3.7.3.3 Response Time

Response time is conceived as the time needed to process data packets and generate the recommended action type. This process starts from launching the prediction task, generating the predicted data, deliver it to the fuzzy logic module to identify the event they represent, and in turn, the corresponding recommended action, and eventually deliver the latter to the



a) Air Quality Level Derived from CO2 and PM25 predictions



b) Comfort Level Derived from Temperature and Humidity predictions

Figure 3.22 FL and Regular Rules comparison in detecting air quality and comfort

edge. Mainly, two implementation scenarios have been investigated. More specifically, in the first scenario, the entire modules constitute the proposed approach have been deployed on the raspberry pi as shown in figure.3.19. These modules are configured to functions independently and communicate with each other through the MQTT broker. While, in the second scenario, the modules that represent our approach were divided into two layers namely the edge layer that is combined from the data reading and transmission module, in addition to the module that receives the result of the data processing step sent by the cloud. In this case, we deployed Ubuntu VM on the Amazon Web Service (AWS).

As shown in figure.3.23, the response time in the first scenario is relatively lower compared to the second scenario. However, the figure also depicts some peaks in the response time generated by the cloud-based scenario. This due to the time needed to communicate data to

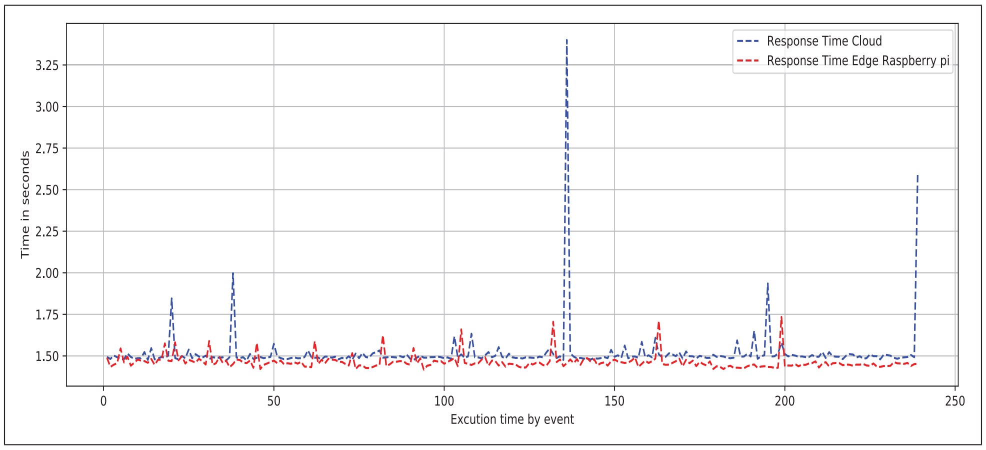


Figure 3.23 Response Time by Event in two different scenarios

the cloud for processing and communicating back the results to the edge. Also, note that in our implementation the cloud communicated with only one edge device which is likely not the case in real use cases. Where mostly cloud platforms handle communication overhead and data processing tasks from multiple IoT edge devices. This is likely to increase the response time and lead to more pressure on the network, as well as, raise the bandwidth consumption.

3.7.3.4 Resource Utilization

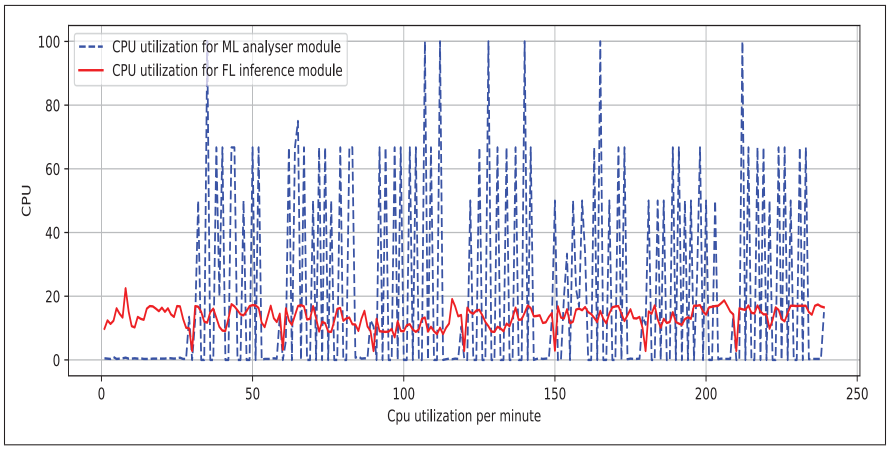


Figure 3.24 CPU utilization

Two resource utilization parameters have been considered namely: CPU and RAM. As shown in figure.3.24, the CPU utilization of the ML analysis module is significantly higher than the FL inference module. This is likely due to the overload induced by the prediction process performed by the ML analysis, which is essentially reliant on running four ML models simultaneously. Indeed, this point will require more investigation to achieve optimal resource utilization in similar cases. While on the other hand, it seems that the strategies followed to reduce the size of the rule base used by the FL inference module have contributed to reducing its CPU utilization. Additionally, both modules have shown relatively similar RAM utilization, which as depicted by figure. 3.25 is reasonable and compatible with resources limited devices such as raspberry pi.

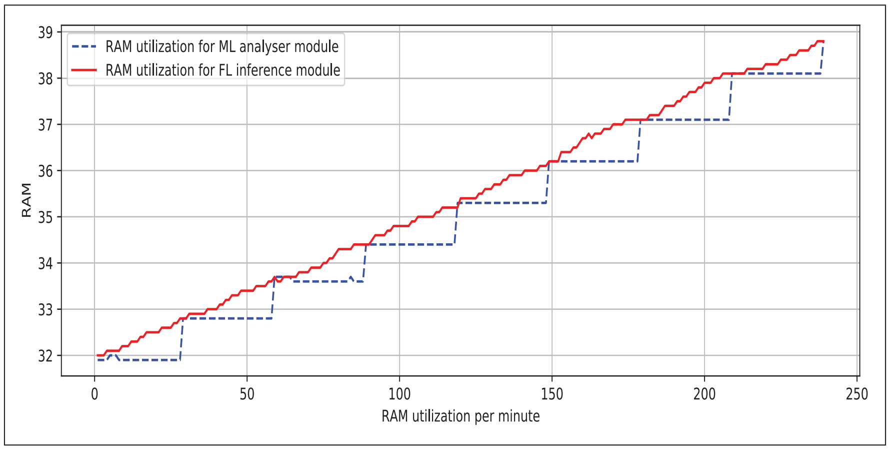


Figure 3.25 Memory utilization

3.8 Conclusion and Future Work

Conveying intelligence to affordable IoT edge devices has contributed to the alleviation of the workload imposed on IoT networks resulting from relying on the cloud as a sole source of computing and reasoning capabilities. This work proposed an approach to enable IoT data processing and decision-making functionalities on the network edge and close to data sources. Two artificial intelligent concepts have been adopted namely: machine learning and fuzzy logic. While the former is intended to analyze received data packets in real-time and generate predictions to identify prospective future patterns. The latter is intended to model these predictions in the form of predefined events using fuzzy logic rules based on linguistic values that represent a set of events in the form of proactive alerts and warning notifications. These events in addition to the model credibility criterion, presented in this work, are fed to another FL rule base proposed to identify the required action to proactively respond to the anticipated event. The proposed approach has been evaluated in the context of an indoor air quality and comfort use case and implemented on an IoT edge device. The obtained results have shown its suitability to IoT edge devices considering running several ML models simultaneously on resource-limited devices. Moreover, the approach has proved its efficiency in terms of achieving faster response time avoiding the round trip to and from the cloud. Furthermore, the fuzzy logic inference engine has shown its efficiency in tracking events of interest compared to regular rules. As a future perspective and to improve the deployment and the scalability of the approach, we are keen to investigate the concept of microservices to enable more resilience solutions. Furthermore, more consideration will be devoted to investigate approaches that keep tracking of the performance of the ML model and enable the model retraining and redeployment without interrupting the whole process.

CHAPTER 4

A FUZZY LOGIC BASED APPROACH FOR EFFICIENT MACHINE LEARNING APPLICATIONS ON IOT EDGE DEVICES

Mahmud Alosta[1], Ahmed Bali[1], Abdelouahed Gherbi[1]

[1] Department of Software Engineering and IT, École de Technologie Supérieure
1100 Notre-Dame Ouest, Montréal, Québec, Canada H3C 1K3
Paper submitted to the journal of Concurrency and Computation: Practice and Experience on MARCH 2021.

ABSTRACT Extending machine learning applications to the network edge has significantly underpinned the real-time data analysis tasks in the IoT domain. A typical edge-based IoT data analysis workflow involves several steps including training ML models using historical data. Afterward, these models will be deployed on the edge to perform real-time analysis on streaming sensory data. In addition, retraining procedures are initiated periodically to update and calibrate model parameters. This topology might perform insufficiently, where IoT data characteristics are highly exposed to variations due to the unexpected fluctuations and the dynamic nature of the observed context. These issues will be reflected in the machine learning models' performance especially on the long-term deployment which eventually would be reflected in the overall performance of the IoT system. Furthermore, transferring all IoT data to the cloud for retraining is likely to lead to amplify network bottleneck issue. In this work, a fuzzy logic (FL) based approach is proposed to keep track of the ML models' performance, on IoT edge devices, and to mitigate the impact of any degradation. FL inference engine is used to apply monitoring rules and it functions in two steps. First, generating a clear and interpretable indicator that reflects the actual current model's status. Second, reasoning over the generated indicator to derive operations needed to sustain a competent model performance. The proposed approach has been evaluated using several ML models and under air quality monitoring use-case. Several criteria have been considered including improvement of model efficiency, resource utilization, and data reduction.

Keywords: Internet of Things (IoT), Fuzzy Logic (FL), Machine Learning (ML), Edge Computing (EC).

4.1 Introduction

Nowadays, the globe is witnessing a new wave of information and communication technology that is featured by the unprecedented dissemination of smart devices. A recent cisco estimate announced that Internet-connected devices could reach up to 29.3 billion by 2023 CISCO (2020). Notably, IoT devices is expected to consolidate 50% of these figures resulting in nearly 14.7 billion connected IoT devices CISCO (2020). These devices have the capabilities to constantly observe the surrounding and to capture observations of interest. IoT devices are increasingly penetrating our daily life activities. This has contributed to the evolution of a spectrum of novel applications and services; leading to the generation of enormous amounts of data Gupta & Quamara (2020). Analyzing data generated by the large scale deployed IoT networks is of utmost significance for several IoT applications Gubbi *et al.* (2013) . These data are likely to be captured in different forms and from a diversity of devices. In order to manipulate data generated by such types of devices, they are likely to be transferred to the cloud Al-Osta *et al.* (2019). However, in several IoT use cases, real-time analysis of data is of utmost priority, and maintaining the lowest response time possible is vital.

Thus, a recent tendency toward delegating such type of processing tasks to the devices at the network edge and close to data sources is gaining more attraction Shi *et al.* (2016). Thanks to the recent efforts to improve storage, computation, and networking capabilities of edge devices, more IoT use cases have become reliant on them not only to maintain communication between cloud and sensor nodes; but also to perform some data processing tasks Dautov *et al.* (2018). For instance, edge devices can be utilized to reduce the amount of data transferred to the cloud by performing some kind of filtration algorithms Papcun *et al.* (2020); Bali *et al.* (2017); Jain & Mohapatra (2019). Also, these devices can improve quality of services by decreasing latency especially in scenarios where a quick response is essential Stojmenovic *et al.* (2016); AlSuwaidan (2020). However, a considerable bunch of these devices are featured by their relatively limited resources, which limits their capabilities to perform some data analysis tasks that rely on some heavyweight artificial intelligent technologies such as machine learning. These technologies are of significant vitality to decode the meaning behind raw sensory data and to improve IoT services abilities to detect and identify events in real time especially at the edge Merenda *et al.* (2020).

In particular, Machine learning has been widely accepted as a data analysis means that facilitates revealing ambiguity behind raw sensor data. This is achieved by running ML models in real time to generate predictions that can be used in further steps as an input to proactive decision-making procedures Abdulkareem *et al.* (2019); Merenda *et al.* (2020); Murshed *et al.* (2019). In this case, ML models are likely to be trained on a large amount of historical data to deduce correlations and calibrate their parameters. This process is likely to be performed on resource-rich platforms such as the cloud. While the prediction process is preferred to be performed on the edge side and close to data sources to enable faster response time and to avoid any consequences resulted from the edge-to-cloud round trip. In addition, a more significant concern is induced by the dynamic characteristic that features both IoT systems and data generated by such systems that more likely reflect the changes in the observed environment Aggarwal *et al.* (2013). Such dynamicity is likely to lead to fluctuating the distribution of data which will be used later to train ML models. Thus, it is undoubtedly that exploiting timely data can define the status of interest in real time and in an unambiguous manner Ma *et al.* (2013). However, this type of change might occur while the ML model performing prediction tasks. This implicitly would lead to some kind of inconsistency between the patterns of data used to train the model and the upcoming received data. Consequently, it is more likely that the ML model would fail in generating predictions that reflect the ground truth data.

On the other hand, IoT sensory data are very likely to be time-series. More specifically, this means data are chronologically ordered and the prior data are likely to convey valuable insights for the future status of the observed event of interest. One of the steps required to enable time series forecasting is the ’lag feature engineering step’, which is also known as a training window or sliding window (SW) Ranjan *et al.* (2021). In this step data to train a model are grouped into a sequence of attribute-value format Parmezan *et al.* (2019). This sequence represents the set of features and the target label Braei & Wagner (2020). Each sliding window comes in length ‘L’ that is equal to the number of the used features. For instance, if a model was trained on a sliding window of length L = 5; then, the value of order 6 is deemed the target label. In this case, 5 time series values are used as features to train the model. This point is vital taking into consideration two trade-offs. First, the longest the sliding window is likely to lead to better accuracy. Conversely, the longest sliding window implies more features to be used to train the model, which will result in more time and computing resources to train the model Akbar *et al.* (2017). Moreover, the model will be more complex and using it in real time and resource-limited devices, such as IoT edge devices, will result in more resource consumption and longer processing time.

Another important parameter that will likely influence the ML model performance in real time is the prediction horizon (PH) or the prediction window. The prediction horizon refers to the time steps ahead that a trained model is required to forecast Munir *et al.* (2018). For instance, if PH = 5, that will likely imply that the model is configured to generate a prediction for the next 5 steps in time. These steps might be recognized in the length of minutes, hours, days, or even months considering the context where the model is running. A multi-step recursive prediction strategy is widely applied for time series forecasting Bontempi *et al.* (2012). In this strategy, if an SW of length L is used to produce predictions of a PH of length M and L<=M. In this case, L+1 is the first step in the PH and the recursive strategy is used to generate its value. The generated value will be appended to the SW sequence; while removing the first element in the SW to keep the same length used while training and to avoid any inconsistency faults Imdoukh *et al.* (2019). This prediction and the replacement process will continue until the PH maximum length is reached. Implicitly, this would involve the use of several predicted values to generate predictions for the next time steps. Therefore, the larger the PH is, the higher the error-rate is expected. This is because in every time step ahead the error will accumulated, since more predicted data will be involved in the prediction process Bontempi *et al.* (2012).

Several accuracy measures have been applied to assess the model performance including Root Mean square error (RMSE), R scored, and Mean absolute percentage error (MAPE). In this work, we use the RMSE as a measure of accuracy. This selection is motivated by the conclusion drown by Chai & Draxler (2014) in which the authors stated that unlike MAPE, RMSE avoids the use of the absolute value, which causes inefficient calculation of the model sensitivity. On the other hand, Chen *et al.* (2017) points out to some difficulties when it comes to the interpretation of the RMSE value. This leads to some ambiguity of meaning. To elaborate more, RMSE value varies based on the parameter under consideration for a prediction task. Considering an air quality prediction as an example. In this case the air quality is based on the prediction of two parameters namely : CO2 and PM2.5. Employing ML models to predict the two parameters values and using the RMSE as a measure of accuracy might be misleading. Since both factors have different range of measurements scale, the value of CO2 RMSE is different from the PM2.5 RMSE. For instance CO2 may take a value in the range form (0 to 5000); while PM2.5 readings may vary from (0 to 150). In this case, it is very likely that the RMSE value of CO2 is higher than the PM2.5; but it is not necessarily implies that the model of PM2.5 is performing better than the CO2 model. This emphasizes the need to a mechanism to clearly recognize the model performance regardless of the scale of the value of the parameter under consideration.

This work addresses the above-mentioned challenges by proposing a fuzzy logic approach to monitor and partially adapt the edge-deployed ML models. Thus, mitigating the impact of any potential deviations in the distribution of the gathered sensory data. The workflow of the proposed approach is composed of two main layers namely: cloud and edge layers based modules. More specifically, the cloud layer is intended to create, evaluate, and select the ML models to be later deployed on the edge. Furthermore, the cloud layer is proposed to receive notification of model retraining and re-engineering requests based on the reasoning over monitoring criteria at the edge layer. On the other hand, the edge layer is composed of two modules namely: the analyzer and the monitoring. While the analyzer is proposed to serve the edge deployment of ML models to perform real-time predictions and constantly compute the accuracy measure (RMSE), which will be used later for the monitoring process. The monitoring module is based on the fuzzy logic concept and it functions in two steps. The first step is intended to convert the RMSE value to a crisp value that it is interpretable and it can be smoothly used regardless of the scale of the parameter it represents. This value is called the model quality (MQ) and will be fed in a later step to another FL-based reasoning module to infer the decision needed to respond to any model degradation. In addition, to forward a feedback procedure to the analyzer module in the form of adjusting either the PH or the SW in order to cope with any prospective model deterioration incident. In the following points, we summarize the man contributions of the presented work in the light of the aforementioned challenges

- the investigation of an efficient way to perform real-time data analysis over IoT data on resources limited devices;
- the evaluation of multi-step recursive prediction strategy, taking into account the propagation of the error-rate and how to alleviate it;
- a fuzzy logic-based mechanism to convert the RMSE value from scale-dependent to scale-free measure that can be interpreted and used by the engine rule regardless of the predicted data. This mechanism is not domain-specific and can be applied with minor modifications to meet the requirements of different use cases;
- fuzzy logic-based rule engine to calibrate the SW and PH values in order to maintain a balanced prediction quality and resource utilization on IoT edge devices;
- evaluating the proposed monitoring mechanism using a real dataset and several machine learning algorithms. In addition, assess its feasibility in comparison with a baseline approach in improving models' performance, maintaining balanced resource utilization, and reducing the amount of data to be sent to the cloud.

The remainder of the paper is organized as follows: Section 4.2 presents the related work and highlights the gab in the current literature regarding the ML model monitoring . Then, several concepts that are applied in this work is given in Section 4.3. The conceptual design of the components forming our proposal is detailed in Section 4.4. While Section 4.5 presents the experiments that have been carried out to evaluate the proposed approach including a two different implementation scenario comparisons. Finally, Section 4.6 concludes the paper and highlights perspective directions for future work.

4.2 Related Work

There is a growing tendency toward incorporating of artificial intelligence algorithms into the network edge. This attraction has been pushed by the significant increase in the development of IoT applications and the diversity of its use cases. Merenda *et al.* (2020) have presented in their work a detailed review of models, architecture, and requirements of IoT edge-based solution to implement machine learning. They carried out several compression experiments deploying ML models on IoT edge devices with limited resources. Also, Murshed *et al.* (2019) discussed several machine learning and neural network models, and several IoT edge-based use cases such as healthcare and smart home monitoring. This study has shed some light on common frameworks used to deploy ML models on edge devices, and several resources-constrained hardware platforms capable to run ML models at the network edge. Another interesting studies Cui *et al.* (2018); Samie *et al.* (2019) have surveyed the importance of embedding ML into IoT applications, and in summary, they emphasize the great potential of ML as a key enabler of endowing IoT devices by the capabilities of information inference, data processing, and intelligence.

Due to the diversity of IoT data sources, and its distinguished temporal and spatial characteristics, it should be considered differently from other homogeneous inputs, such as image and audio during preprocessing steps. Several initiatives in the literature attempt to propose approaches to facilitate the creation of ML models to serve different IoT use cases. Alves *et al.* (2019) proposed Machine Learning Framework for IoT data (ML4IoT). It mainly concentrates on facilitating the creation of ML workflows on large amounts of different IoT time-series datasets using different ML algorithms. Two types of workflows were proposed namely Batch and Online learning. The batch workflow comprises steps followed for data pre-processing, model selection, parameter configuration, training, and testing. While the online workflow considers the model evaluation using new data chosen from sources similar to the ones used for model training.

In addition the literature demonstrates several proposals that have been applied using conventional ML methods for real-time predictions. To provide faster and more efficient decision-making the authors in Al-Rakhami *et al.* (2018) proposed an edge-based solution using ML models for fall detection scenario. Support Vector Machine (SVM) algorithm is used to classify human activities based on data received from sensors. This framework divides the data collection and classification tasks at the edge into small micro-services. To achieve more precision and accuracy, a similar approach based on deep learning presented by Sarabia-Jácome *et al.* (2019). Additionally, several efforts have been devoted to enable the combination of FL and ML techniques for IoT data processing. To maintain faster response time and latency reduction in IoT healthcare applications, the authors in Shukla *et al.* (2019) proposed a hybrid approach of fuzzy logic and reinforcement learning for data processing. A 3-tire IoT architecture is applied, in which data are first gathered at the first layer and classified using a fuzzy-based inference engine. Then the systems forward the data to the second layer (fog) that runs a reinforcement learning model to select the most time-sensitive data to directly sent to the end-user, while data with less importance are sent to the cloud for storage and future use. In Iram *et al.* (2019) the authors proposed a real-time traffic controlling system using machine learning and fuzzy logic. The system was deployed on several edge devices to perform context-aware traffic signals controller based on the occupancy information recorded in a specific location. Occupancy information is gathered on a large-scale; analyzed and clustered using the K-Means algorithm. Then fuzzy logic applied on the clusters, thus a longer duration is assigned to edges with a higher density of traffic. Additionally, a recent work by Guillén-Navarro *et al.* (2021) proposed an edge and ML based solution to alleviate the impact of frost in stone fruit fields. The proposed system encompasses ML based intelligent module to perform predictions, which used to activate or deactivate the anti-frost technique. For proof of concept, the authors created their own testbed consisted of several wind, temperature, and humidity sensor nodes. To improve their system predictions, first, the authors developed two outliers detection techniques based on k-nearest neighbors and k-means clustering. Although obtained results have shown the efficiency of the proposed approach, implementing k-means clustering technique is computationally expensive for a considerable portion of IoT edge devices.

On the other hand, a combination of ML and Complex event processing (CEP)based approach is proposed by Akbar *et al.* (2017). The authors proposed an adaptive moving average approach to train ML models and a flexible prediction window mechanism is used to perform real-time predictions. This as revealed in the experiments has lowered the propagation of error-rate. They used MAPE as an accuracy measure and ordinary If rules to control the adjustment of the PH. In comparison to the presented work, we propose to use RMSE as model performance tracking measure; in addition, a FL based rule engine is proposed to guide the selection of either adjusting the sliding window; hence switching ML models. Or to adjust the prediction horizon; thus, more accurate result of prediction are maintained.

Additionally, the concept of sliding window (SW) has been applied in several works in the literature including Vafaeipour *et al.* (2014); Kromkowski *et al.* (2019); Hota *et al.* (2017) to enable segmented-based time series prediction tasks. However, they are mostly oriented for off-line implementation settings and do not meet real time predictions tasks requirements. Further, to decrease the false alarm rate and improve system reliability, authors in Smrithy *et al.* (2019) proposed a sliding window-based approach for real-time detection of anomalies in wireless body area networks applied for healthcare systems. The adjustment of the SW is applied by constantly analyzing the disparity between the previous and current sliding window values. To sum up, except Akbar *et al.* (2017); Smrithy *et al.* (2019), most of the state of the art literature adopts fixed models with large amounts of historical data for training, or rely on initiating the retraining process periodically. This, in turn, might not be sufficient to meet some critical IoT use cases. Especially taking into consideration that the performance of ML models may deteriorate over time as the statistical properties of the underlying sensory data, which form the model variables, may change over time. In addition, this will be reflected in the overall efficiency of the IoT system to run proactive decision-making tasks based on accurately predicted observations.

4.3 Background

In this section presents the main concepts used to form the approach proposed in this work including fuzzy logic, machine learning, and RMSE.

4.3.1 Fuzzy Logic

Fuzzy logic (FL) is a multi-valued logic used to reason over multiple parameters and derive a meaningful conclusion. These parameters can be envisioned as sensory data packets that might be noisy, imprecise, vague, and ambiguous. More specifically, the FL is likely to be used to addresses the 'partial truth' concept. In which the range of the truth value may span from completely true to completely false Kapitanova *et al.* (2012). This is a significant feature that can be used to improve IoT systems responsiveness, especially in handling and modeling uncertainty events. In addition, several operational factors have motivated applying of the FL concept to IoT solutions, including its ability to imitate human reasoning, tolerate unreliable and imprecise sensor readings, improve decision-making tasks. Furthermore, the implementation of the FL is likely to lead a relative low resource utilization, which makes it an optimal candidate to IoT edge based solutions Palevi *et al.* (2019); Gozuoglu *et al.* (2019). As shown in figure 4.1, FLS is consisted of fuzzification, fuzzy rules base, fuzzy inference system and defuzzification.

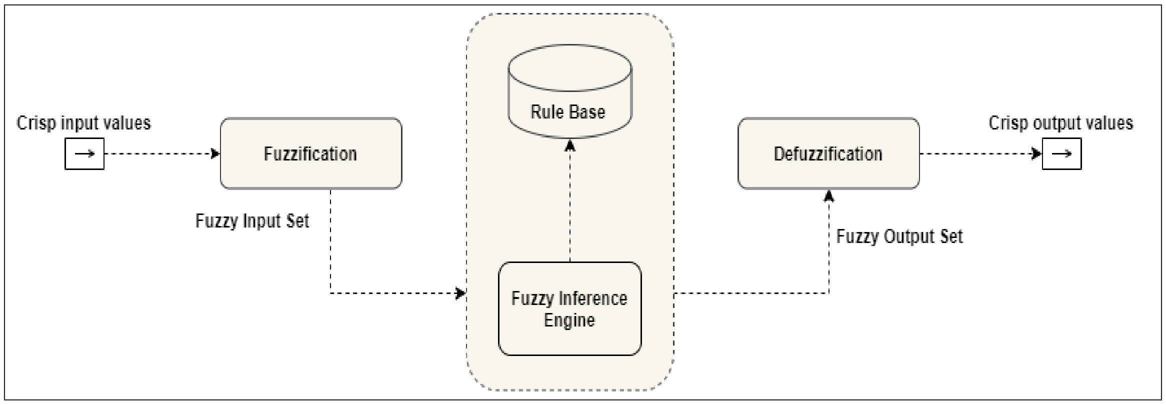


Figure 4.1 The process of a Fuzzy Logic System

- Fuzzification: In this step, the input crisp value is converted to a fuzzy value based on a membership-function that represents the degree in which the value belong to a specific lingustic membership. In an IoT use case, a crisp inputs is likely to be in the form of sensor readings including, CO2, PM2.5, temperature, or humidity captured observations. For instance, a room temperature = 20 C, can be converted using the membership function to a fuzzy linguistic value (eg. the temperature is warm). The membership function might be represented graphically using different shapes. Some of the most commonly used shapes include triangular, trapezoidal, and Gaussian shapesKapitanova *et al.* (2012);

- Fuzzy Inference Engine: It is composed of a set of membership functions, rules, and logical operators. The rules are grouped in the fuzzy rule base, which is consisted of a set of linguistic statements that describe how the fuzzy inference system should identify an input or control an output. The rule-base is likely to be organized taking into consideration all possible value combinations connected by logical operators(e.g. AND, OR, NOT);

- Defuzzification: Applying rules that have been established in the previous step creates several shapes that represent the modified membership function. For instance, let's consider a rule-base is intended to identify the possibility of having an inappropriate air quality level as an incident of an unexpected level of CO2 or PM2.5 levels. In this case, the output of the fuzzy inference engine might be as follows: Low(65 %), Medium (30%), and High(5%). The role of the defuzzification is to transform this set of percentages into a single crisp value that precisely describes the event of interest. The implementation of rules located in the rule-base results in the obtaining several shapes that form the modified membership function Kapitanova *et al.* (2012).

4.3.2 Machine Learning in IoT

ML techniques have been widely applied as a key enabler of data analysis in several computer science fields including pattern recognition, image processing, and predictive applications. They are featured by their capability in performing data classification, prediction, and clustering tasks. Recently, they have attracted both IoT academic and industrial communities to apply such intelligent algorithms to improve the outcome of IoT systems. Essentially, ML techniques are intended to promote IoT systems' efficiency to handle data using statistical models. These models are trained using data samples that are identified by measurable characteristics called features. ML models seek to uncover the correlation between these features and the target output values called labels. The obtained labels during the training phase are exploited to support ML models' parameters, which enable them to make inferences on new data samples and generate scores. These scores can be used to identify patterns or make decisions. The underlying principle of using ML in performing decision-making tasks is that a score can be an indicator of an event of interest that might require attention. In this work, three well known ML algorithms have been considered in the evaluation of the proposed approach. Below, a brief description is presented:

- Bayesian network (BN): the principle design of the BNs involves the combination of graph and probability theories. While the graph theory is intended to visually represent the structural model of BN. The probability theory is used to represent a function that assesses the uncertainty of two variables' correlation. The structured model is represented by a directed acyclic graph (DAG) composed of nodes and arcs. Given node on the graph corresponds to a discrete or a continuous variable, while arcs demonstrates the interactive dependency relationship between such variables and represented by probability function. Generally, the learning process of a Bayesian network is launched by the induction of its two components and then proceeds to the interpretation. The graphical structure of the model's dependencies indicates the values selected by the model. Within the parameter estimation step, the probability distributions and quantifying the dependency structures are estimated Sebastiani *et al.* (2009);

- Support Vector regression (SVR): this algorithm is based on the statisical learning theory and structural risk minimization principle. SVR is the regression oriented version derived from support vector machine (SVM). SVM is a learning algorithm that functions based on high dimensional feature space. The prediction function in SVM is subject to the expansion of a subset of support vectors (SVs). These SVs can be envisioned as data points that achieve the maximal margin. Mainly, SVR functions based on two parameters namely: Kernel function, and penalty and non-sensitive coefficients. While the purpose of the former is to transforms input data points into a lower-dimensional feature space resulting in the reduction of the estimated parameters. The latter two parameters are intended to optimize the prediction efficiency Steinwart & Christmann (2008); Iram *et al.* (2019);

- Multilayer Perceptron (MLP): MLP is a form of the artificial neural network (ANN). In ANN the model functions similarly to the human brain in fulfilling information processing. ANN is constructed of a set of nodes (generally arranged in layers) that are interlinked to each other. They are capable of recognizing unseen patterns and generalize through learning from both input patterns and errors they observe during the training stage. MLP is known for being capable of representing an effective functional link between the inputs (predictors) and the output (predictands). In addition to the input and the output layers, a typical MLP also composes a set of the hidden layer, which map between the other two layers through means of connector units. Setting an appropriate number of hidden layers and specifying the number of neurons in each layer has been recognized as a significant factor in assessing the efficiency of the MLP model Taud & Mas (2018).

In addition, several ML validation parameters are applied to measure a model performance efficacy including, for instance, RMSE,MAPE, and MSE. These criteria can be used to guide the selection of a certain model over another. In the following, we introduce the accuracy measure used to assess the model performance in this work.

4.3.3 Root Mean Squared Error (RMSE)

RMSE measures the absolute fit of a regression line to the actual data points by calculating the distances between them. These distances are the ”errors” and to obtain the RMSE they are being squared and then extracting the root of the final value Mehdiyev *et al.* (2016). The lower RMSE value indicates a better model fit, and it is computed based on the following equation:

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}\left(O_i' - O_i\right)^2}{n}} \tag{4.1}$$

where O_i' is the predicted observation, while the O_i is the actual value and n is the number of observations available for analysis. RMSE is is a scale-dependent measure of accuracy, which means that its value is dependent on the scale of the used data. Thus, RMSE is more effective in comparing forecasting methods on the same set of data and not across datasets that are on different scales Chen *et al.* (2017).

4.4 The Proposed Approach

Three-layered architecture, including sensor, edge and cloud layers, is considered in this work. At the lowest level of the proposed framework, sensor nodes are deployed to collect data and forward it to the upper layer. Also, it is configured to receive and fulfill actuation instructions. They are likely to be found within a limited physical area and they are connected to one or more edge devices. Edge devices on layer two offer more capabilities than devices at layer one, thus, it is feasible to direct them to perform some relatively resource-demanding tasks such as data aggregation and analysis. This is significant to avoid the round-trip delay imposed by sending data for processing to the cloud and to realize a faster response time.

In addition, cloud platforms have a wide scale view of all deployed edge nodes. Furthermore, these platforms are capable of carrying out complicated tasks such as machine learning training, which is resource-greedy task. Also they can be used for large scale analysis and visualization tasks on large datasets that are stored on demand. Thus, it becomes feasible to extract information, patterns and correlations from diverse data gathered from various IoT networks and devices. In our approach, the cloud also plays an important role in the selection of the ML model that is best suited to the specific task and dataset. In this work, we propose a Machine Learning Framework Manager that demonstrates steps followed to select, create and test machine learning models to be deployed in later steps to generate real-time prediction on the edge. In addition, the edge side will be equipped with model assessment module that functions as a real-time monitoring tool to catch any deterioration in the model performance and response accordingly. The following sub-sections provides more details.

4.4.1 Machine Learning Framework Manager (MLFM)

The creation of ML models is likely to be a resource-intensive task involving several steps required to tune the model parameters. Thus, in our approach, we assume that the MLFM as a core module to be deployed and functions on the cloud or capable fog nodes. As shown in figure 4.2, MLFM represents the steps followed to select data, pre-processing steps, creating, testing and evaluating ML models and eventually storing it or deployed on the edge side. It also receives retraining and re-engineering requests from the edge to apply the required updates on used ML models and redeploy them in a form that is better reflect changes in data gathered in real time. Thus, three essential tasks are applied by the MLFM as follows:

4.4.1.1 Model Creation

This task represents the primary stage of the model life cycle. It starts with the creation of the dataset to be used to train a model. This dataset might be generated from single or multiple data sources given the task of the service to be equipped with the trained model. In addition, several processes including data cleaning and pre-processing steps, feature engineering, and model selection are applied to fulfill this task. More specifically, the feature engineering process includes the transformation of time series data points into sequences of 'attribute-value table'. The length of these sequences represents the Sliding window (SW) that specifies the number of variables used to train the created ML model.

4.4.1.2 Model Retraining

The task of re-training arises when a certain edge device sends a notification request to retrain a model due to inefficient performance. It also might be configured to be initiated periodically.

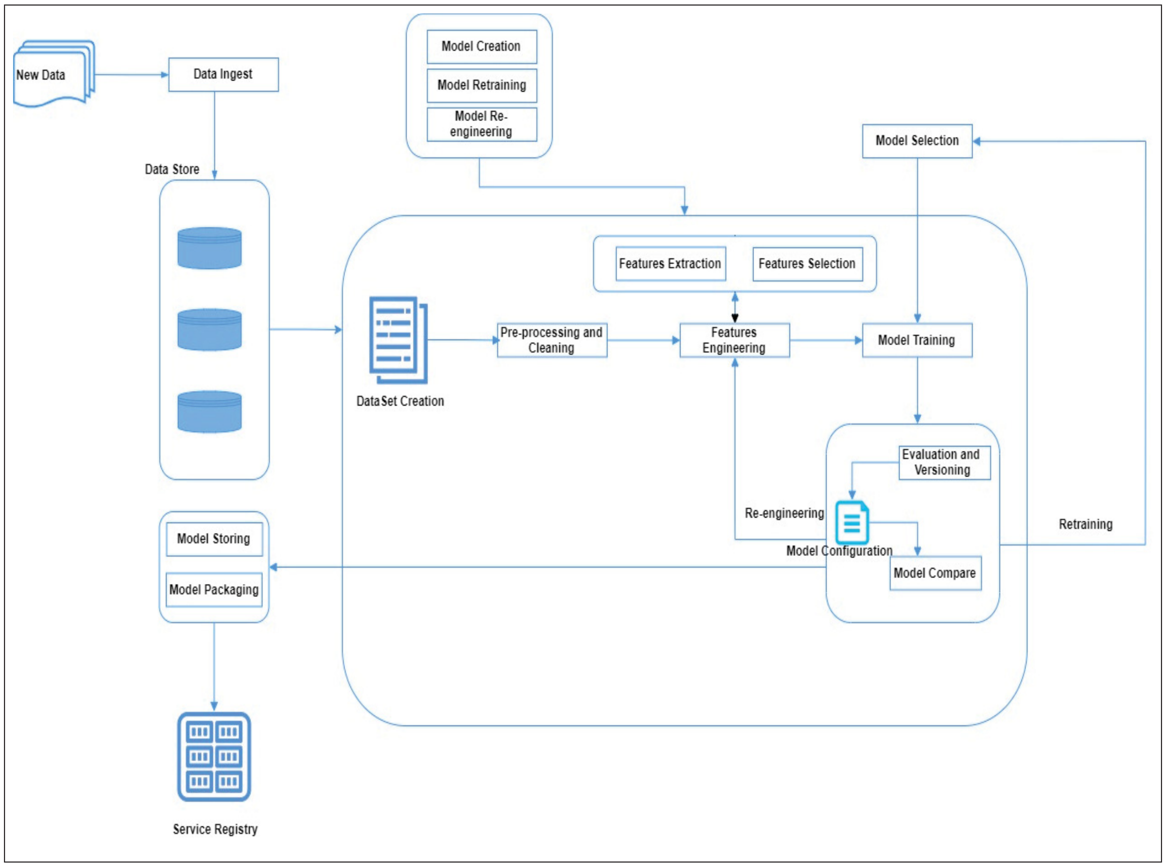


Figure 4.2 Machine Learning Workflow Manager.

This step involves updating the dataset with new data instances coming from the edge side. To reduce data to transmitted from the edge, our approach seeks to identify data points that had led to the degradation of the model performance and exclusively transmit them to the cloud to calibrate the model parameters.

4.4.1.3 Model Re-engineering

This task is launched in case of a change of the features set to be selected to train the model. This is likely to be driven by either the change of the sensors used to generate data (i,e. adding or removing sensors), or by the need to train the model using different features to preform another inference task. In addition, given the task that require the deployment of ML model and the prediction horizon that the model is demanded to predict, a feature re-engineering task may involve the adjustment of size of the sliding window. For instance, increase the sliding window to generate more accurate predictions especially when the prediction horizon is long. Or decreasing the sliding window for short predictions and to enable optimal use of computing resources on the edge device.

In addition the MLFM is encompassed of several sub-modules as illustrated in Figure 4.2, which are organized as set of consequence steps in order to create and evaluate ML models.

4.4.1.4 Data Ingest

In this step, datasets stored in the data-store component are updated using incoming data instances. This is important to perform the training and retraining processes on up-to-date data which will be reflected on the performance on the created model. Data stored in the data store are called historical or cold data, while new data are called fresh or warm data. The ingestion process could be scheduled to run periodically, for instance hourly, daily or weekly. Or, it might be configured to run on demand according to requests coming to the MLFM for either model retraining or re-engineering. Afterwards, a dataset creation step is performed to select data required for model creation from different data sources placed in the data-store.

4.4.1.5 Model Selection and Training

It defines the ML algorithms along with their parameters to be used to train ML models. For example, the type of algorithm (e.g.BN, MLP, SVR), specific algorithm configurations, and how the datasets are split between training and testing are some of the parameters defined in this step.

In addition, several ML validation parameters are calculated by this module such as RMSE. RMSE is used to guide the selection of a certain model over another. Afterward, a versioning function will be used to tag the model with a unique ID. The ID will be used later to facilitate access to the model for deployment or comparison purposes.

4.4.2 Model Configuration and Comparison

In this component, a log file is proposed to models configuration data. These data includes some descriptive information about the used model and training configuration including the Model type, selected features, and execution time. The attributes of the file is presented in Table 4.1. Furthermore, to ensure the selection of the optimum model, this component carries out a model comparison process. Models either with different or similar configurations and different version IDs are compared in order to guide the selection process. In case the model dose not meet the required performance score threshold, a retraining or re-engineering process will be launched.

Table 4.1 ML Model Description

No	Attribute name	Description
1	Model-Id	Each ML model has a unique identification.
2	Type	Specifies the used algorithm (i,e. SVM, MLP, BN)
3	Selected features	Set of features used to train a model
4	Dataset size	Number of instances of the used training dataset
5	Preprocessing steps	Includes data aggregation, imputation,
6	Training and Testing data	Splitting Percentage of the used dataset
7	Performance Evaluation Criteria	MSE, RMSE, R2score.
8	Execution Time	Time needed to create, test, and evaluate the model.

4.4.3 The Edge-Based Framework Components

The timely detection of patterns of interest has been consistently driven by the real-time analysis of sensory data. In the context of IoT applications, obviously, this would have a constructive effect on the overall outcomes of the system and its ability to respond proactively. To realize compatibility with these conditions, the presented edge framework is intended to facilitate the deployment process of ML models close to data sources. Moreover, to maintain both the stability of the performance of deployed ML models and, balanced resource utilization of the edge device, FL based monitoring module is proposed. More details are presented in the following subsections.

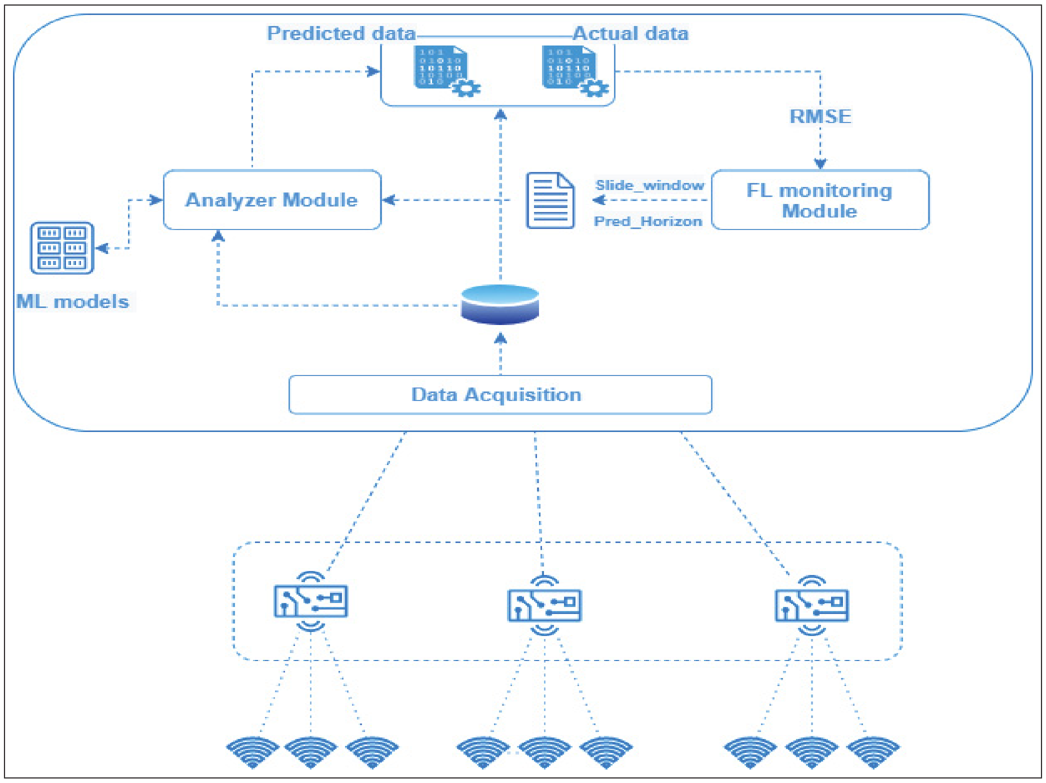


Figure 4.3 The Edge-Based Framework Components

4.4.4 Data Acquisition Module

In a typical IoT scenario, data collection tasks are performed through means of data acquisition APIs. These APIs are conceived as an abstraction layer intended to hide the complexity of the underlying infrastructure. Furthermore, to maintain a channel for bidirectional communication between devices at the sensor nodes and the gateway device at the edge node. In most cases, IoT data are generated in the form of time series sequences and stored on local database file on the edge. In our approach, these data packet will transferred in further steps to the analyzer module for analysis and to generate the prediction of the future patterns. Also, the model evaluation sub module is linked to the local database to acquire the actual data to be compared with the predicted data and calculate the loose function, which in our case is the RMSE. The RMSE will be computed upon the availability of the actual data points that correspond to specific prediction horizon.

4.4.5 The Analyzer Module

The analyzer module is intended to serve the deployment process of ML models at the edge and close to data sources. Several ML algorithms will be trained and deployed on the edge and in real time settings. The intention behind this is to evaluate the performance of these ML models in terms of the progression in the value of RMSE. using the regular approach and without the monitoring in comparison with using our approach. While in regular approach data are sent to the cloud periodically for retaining tasks. Our approach uses a real-time assessment of the model performance. This assessment is used to adapt the model functionality and to alleviate any possible degradation. It is initiated iteratively based on the length of the prediction horizon. This is done by generating prediction scores using the analyzer module. Then calculating the loose function (RMSE) upon the availability of the actual data that correspond to a specific prediction horizon. Afterward, obtained predictions and the values of RMSE, SW, and PH will be transmitted to the model assessment sub-module.

For instance, if we have an SVR model that were trained on sliding window configuration of size 30 and used to make prediction of the next 30 minutes (i,e, prediction horizon of length 30). First, the analyzer module will start aggregating data and applying the predefined preprocessing and feature engineering steps to convert these data points into a format that can be used by the model. Second, after aggregating the 30 real values forming the sliding window, the prediction process will be launched to generate 30 values representing the size of the prediction horizon. The following aggregated 30 real data points, in addition to using it to generate prediction, they will be used to calculate the RMSE which will be used later to assess the model performance.

As shown in figure 4.3, the analyzer module is linked to the ML model registry file. In this file, several machine learning with different sliding window configurations is registered. The analyzer module selects the model to be run based on the feedback received from the monitoring module. This feedback notification is intended to adjust either the sliding window size or the prediction horizon. In the former case, the analyzer will switch between the deployed ML models. For instance, to switch from a model with a sliding window of 30 minutes to a model with a sliding window of size 60. This type of convergence is called in the context of our approach 're-engineering', and it was proposed to avoid the fast deterioration of the model performance; especially if the prediction horizon was longer than the sliding window. Moreover, switching between ML models is likely to lead to optimal use of the limited available resources on the IoT edge device. This is can be conceived in cases where the switch is from a model with a longer SW to a model with a shorter SW.

On the other hand, adjusting the prediction horizon, which called in the context of the proposed approach model 'retraining', is intended to boost the model permanence even if its performance is not deteriorating significantly. This is likely to be constructive especially in two main cases. First, if the prediction horizon(ph) is longer than the sliding window(sw) (e.g, ph= 60, and sw=30) and the model is deteriorating gradually. Second, the available computing resources might not be sufficient to run a model with a longer sliding window. So it is preferable to not switch models. In both cases, the monitoring module will keep tracking the model performance and decrease/increase the prediction horizon until obtaining a satisfactory model performance.

The multi-step recursive forecasting strategy Bontempi *et al.* (2012) is employed to predict multiple next values that represent the required prediction horizon. In this strategy, sensor data are grouped in a queue of constant length that corresponds to the length of the chosen time window. The model is configured to perform one-step-ahead forecasting at each forecasting step in the prediction horizon. At every prediction step in the prediction horizon, the newest obtained predicted value is being added to the sequence concurrently with the removal of the oldest value in the sequence. Although, it is likely that using this strategy would lead to accumulating the error-rate as a result of involving some predicted values in the prediction process. The recursive strategy has been selected since it demands fewer computing resources compared to the direct strategy Taieb *et al.* (2012). To lessen the impact of using the recursive strategy, we adopt an adjustable prediction horizon mechanism.

4.4.5.1 FL Model Monitoring Module

Since our models will be deployed on IoT edge devices and will perform online predictions, a model monitoring and assessment module has been proposed. This module seizes a focal role in our approach where it will be in charge of tracking the model performance and verifying whether the current version of the ML model outcomes does not significantly deviate from the actual observations. The Root mean square error (RMSE) has been selected to measure the performance of the model. Although some other accuracy measures such as R2 scored and MAPE are much easier to interpret and to be used for comparison purposes, they are likely to represent a relative measure of fit. While, RMSE represents an absolute measure of fit, but it's value is not easy to interpret. As previously mentioned, RMSE value series-dependent and it is not easy to interpret especially when comparing two difference scale value parameters such as CO2 and PM2.5 Chai & Draxler (2014); Chen *et al.* (2017). Where the value of CO2 sensor reading ranges from 0-5000 ppm, and the PM2.5 readings ranges from 0-150 µg/m^3. This implies that the value of CO2 RMSE would be likely higher, but it does not necessarily mean that the model used to predict future CO2 readings is performing worse than the PM2.5 model.

Thus, the presented approach is attentive to convert the value of the RMSE into an interpretable and understandable format that can be easily conceived by other software agents. Thus, in this work, the concept of FL has been used to perform reasoning in two steps as shown in figure 4.4. First, converting the RMSE value to a crisp value that is scale-independent. This value is called, in the context of the proposed approach, the model quality (MQ) and. It is a perspicuous representation of the current status of the model performance. Second, eliciting the response needed to alleviate any detected model performance degradation. This is achieved using a set of FL rules that reason over the obtained MQ; taking into consideration other two parameters namely the SW, and PH. Both steps are elaborated with more details in the below-listed points.

1. RMSE to Crisp value: in this step, the RMSE value, in addition to, the value of the Sliding window(SW) and the prediction horizon (PH) are fetched to FL-based sub-module. The resulted value will be used later in a straightforward manner for detecting any degradation

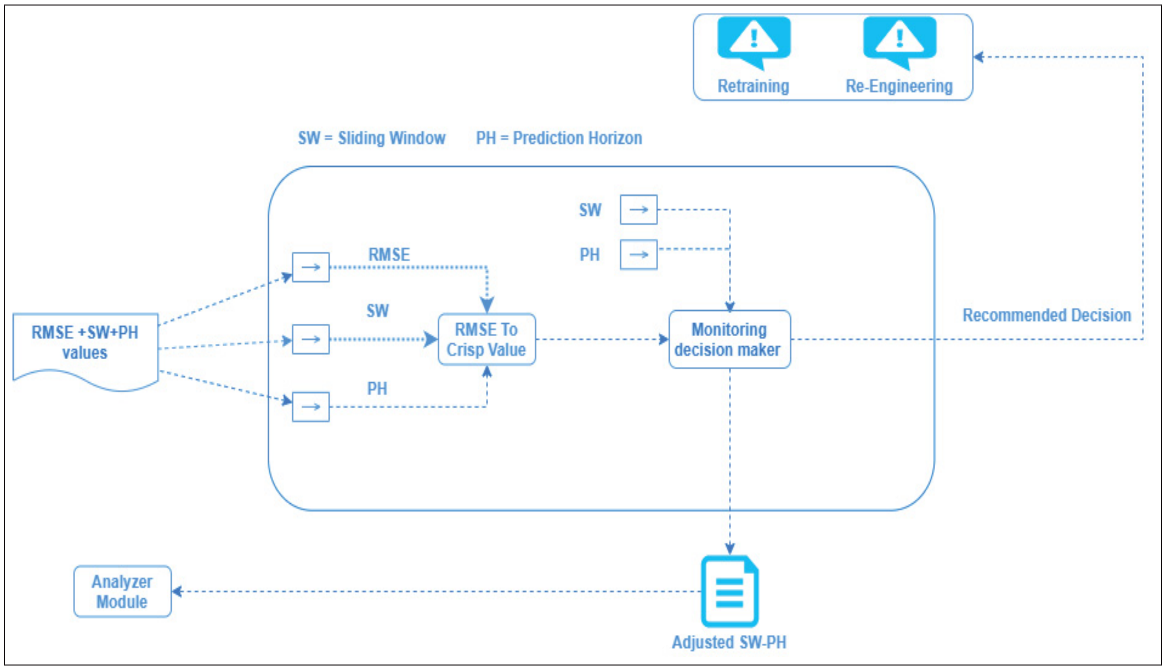


Figure 4.4 The FL based Monitoring Module

Table 4.2 The Fuzzy Rule Base of Model Quality

MQ=Model Quality SW=Sliding Window PH=Prediction Horizon
RMSE=(VL=Very Low, L=Low, M=Medium, H=High)
MQ=(E=Excellent, G=Good, P=Poor, VP=Very Poor)
SW,PH=(S=Short, M=Medium, L=Long)

Rule No	RMSE	SW	PH	MQ
1	VL	S	S	E
2	VL	S	M	E
3	VL	S	L	E
4	L	M	S	G
5	L	M	M	G
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
33	M	S	S	P
34	M	S	M	P
35	H	M	S	VP
36	H	L	L	VP

in the model performance that would require further attention. The values of SW and PH have been involved in this process to add more reasonability to the MQ recognition process.

Taking into consideration that the longer PH is likely to lead to higher RMSE. Likewise, the shorter SW is likely to lead to higher RMSE especially if used to generate predictions for medium or long PH. To cope with this diversity of configurations, this step involves FL rule base as shown in Table 4.2. These rules represent all possible combinations of RMSE, SW, and PH. PH can be conceived as ΔT, which is the time in which the RMSE value has been accumulated. Furthermore, it is important to note that SW represents the number of features used to make prediction of the next step. An example of MQ computing rule is as follows:

IF RMSE is High and Sliding Window is Medium and Prediction Horizon is Short Then Model Quality is Very Poor. ***IF RMSE is High and Sliding Window is Medium and Prediction Horizon is Long Then Model Quality is Poor***.

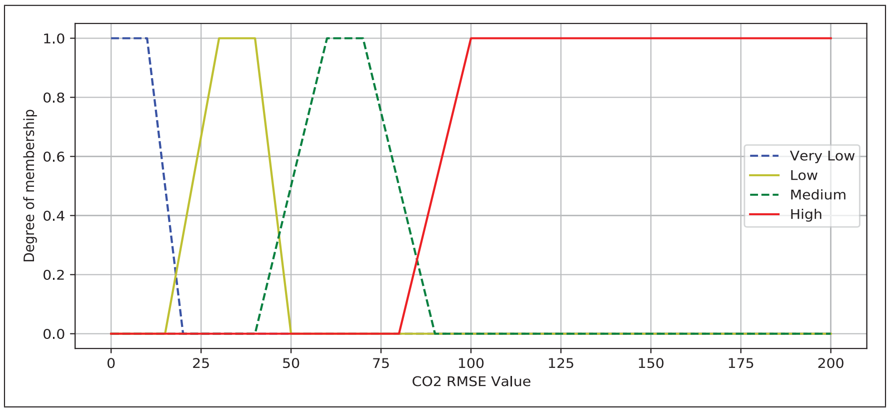


Figure 4.5 CO2 RMSE Membership Function

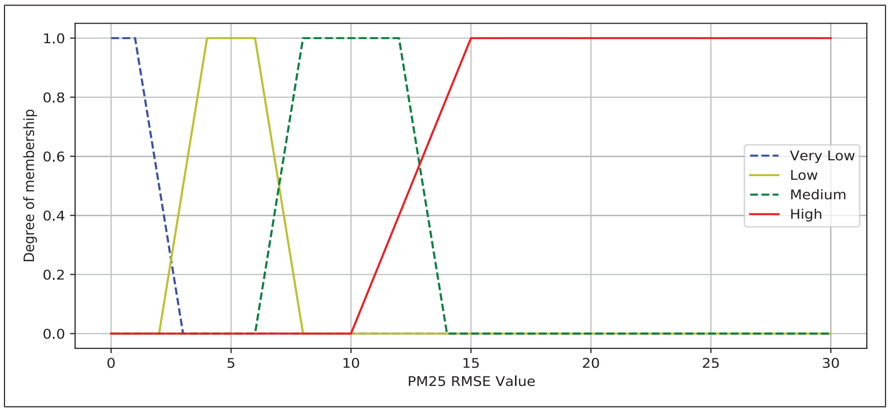


Figure 4.6 PM2.5 RMSE Membership Function

Figures 4.5 and 4.6 describes an example of RMSE membership function of CO2 and PM2.5. While the membership function of the PH and SW is as presented in 4.8. Although

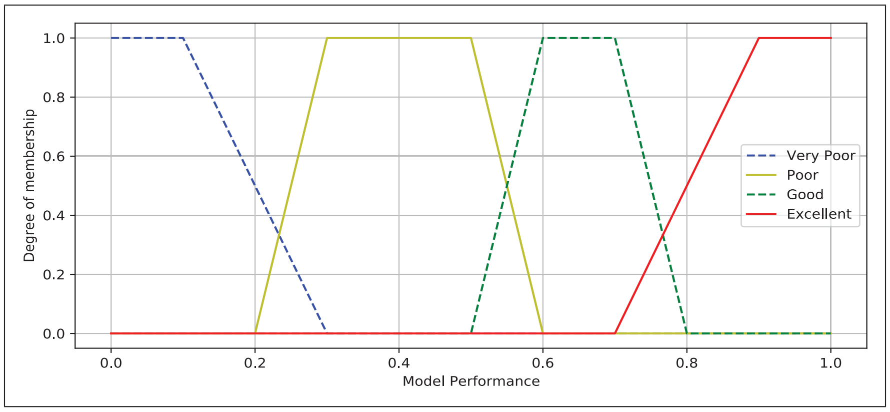


Figure 4.7 Model Performance Membership Function

the RMSE and the SW have similar values in both the above-listed rules, the model quality is different. This because our approach assumes that if the PH>SW is likely to lead to higher RMSE and Vice-versa. Since the recursive multi-step prediction is used in the analyzer module, it is likely that the prediction error to be accumulated with each step in the prediction horizon. This arises because data predicted from earlier steps is used to predict the value of the next time step. Therefore, the longer is the prediction horizon the higher RMSE is expected. However, in the first rule, the RMSE is high; although the SW>PH. This is an obvious indication of very poor model quality. On the other hand, in the output of the second rule, it is assumed that, while the SW is medium and the PH long leads to poor model quality. This output is conceived as a result of having a PH>SW in this rule. Poor quality might be treated by decreasing the PH rather than replacing the model with another model that has a different SW. MQ membership function is demonstrated in figure 4.7, its value ranges from (0 to 1) regardless of the parameter considered in the prediction process and the RMSE value;

2. Monitoring decision-maker: this step is intended to obtain the decision needed to alleviate any degradation in the model performance. The input of this step is the model quality generated by the previous step in addition to the sliding window and the prediction horizon.

Table 4.3 The Fuzzy Rule Base of the Monitoring decision maker

Retraining, Re-Engineering = (L = Low, M = Medium, H = High)
MQ = Model Quality SW = Sliding Window PH = Prediction Horizon

Rule No	MQ	SW	PH	Retraining	Re-Engineering
1	E	S	S	L	L
2	E	S	M	L	L
3	E	S	L	L	M
4	G	M	S	L	L
5	G	M	M	L	L
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
33	P	S	S	H	M
34	P	S	M	M	H
35	VP	L	S	M	H
36	VP	L	L	H	M

While the output is a crisp value that ranges from (0 to 1). This value represents two categories of recommended processes namely: Retraining and Re-Engineering. Although both cases have similar inputs as shown in Table 4.3, and they share similar member function description as shown in figure 4.9, they are likely to be initiated following different circumstances and conditions.

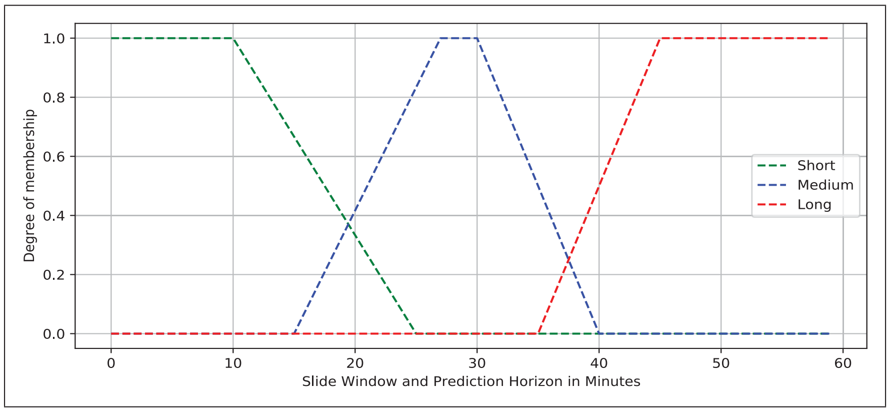


Figure 4.8 Sliding Window and Prediction Horizon Membership Function

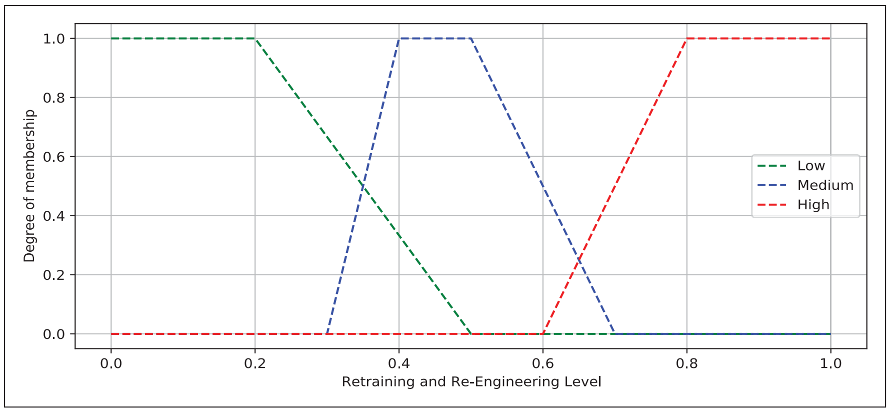


Figure 4.9 Retraining and Re-Engineering Level Membership Function

Therefore, they generate different outputs in most cases of the applied rules. The output of both cases is passed to a simple rule engine to determine the priority of one case over the other as shown in figure 4.10. In our assumption, the re-engineering option has a higher priority on the training. This is because the re-engineering will likely be triggered when the MQ parameter yields a very poor indication. Thus, the re-engineering option will have a more positive and faster impact on the model performance, since it will lead to switching the current model to a model with a larger SW.

1) Retraining: this category of processes is mainly induced to alleviate the impact of the prediction horizon on the quality of the prediction task. In this case, whenever a retraining decision is identified, the system will run two tasks. First, transmitting a notification of retraining request to the system on the cloud. This task is likely to involve transmitting data to the cloud in order to update the dataset used previously to train the model. In the second task, the system assesses the crisp-value of the retraining request and automatically increases/decreases the prediction horizon to minimize the error rate. For example, if a model with medium SW= 30 and long PH = 60 and an MQ = poor. In this case, the retraining process will decrease the PH and keep monitoring the model for each prediction episode till obtaining a satisfactory MQ. In this case, we avoided switching our model with another model with longer SW (e.g. 60), which

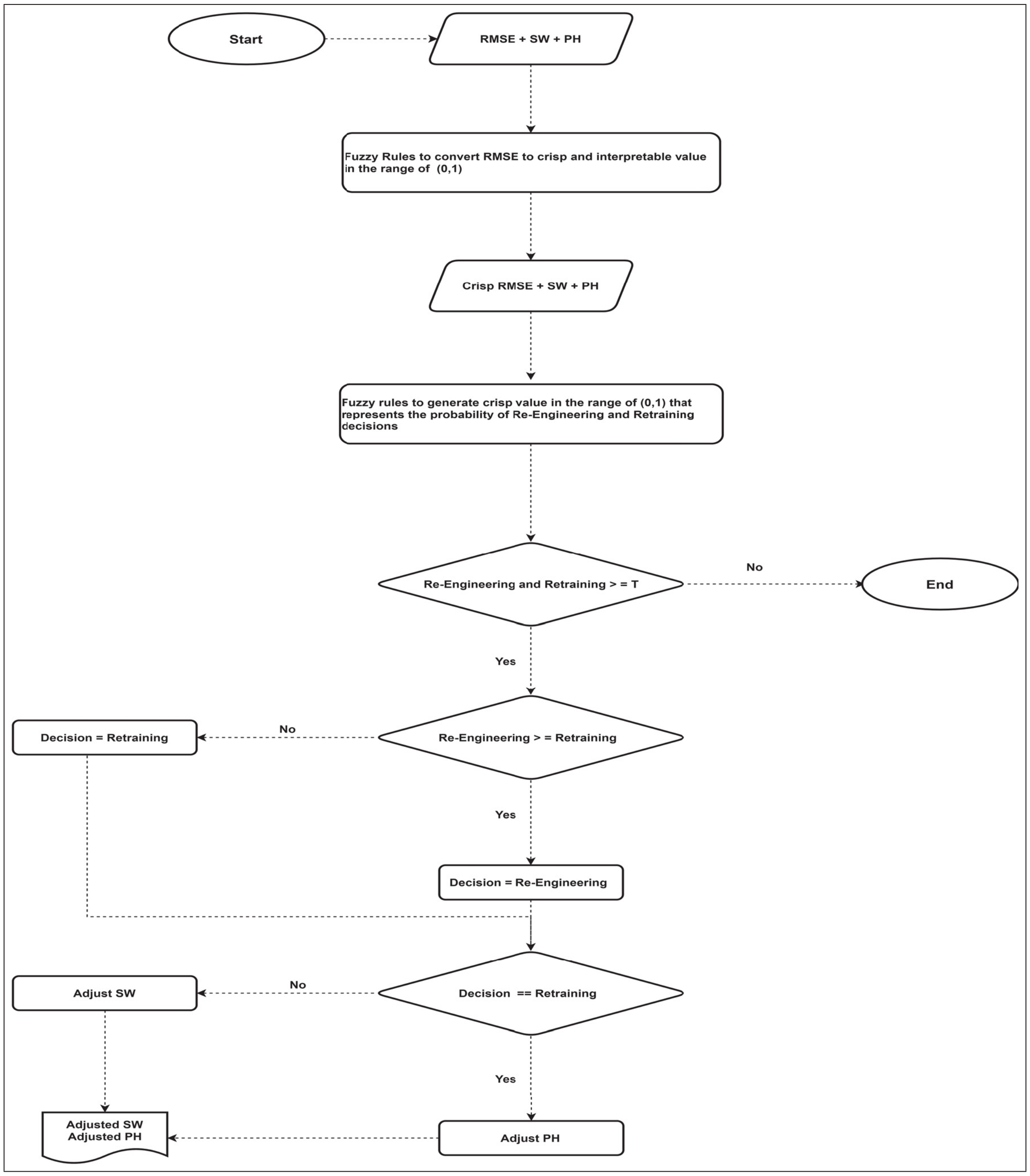

Start
RMSE + SW + PH
Fuzzy Rules to convert RMSE to crisp and interpretable value in the range of (0,1)
Crisp RMSE + SW + PH
Fuzzy rules to generate crisp value in the range of (0,1) that represents the probability of Re-Engineering and Retraining decisions
Re-Engineering and Retraining > = T
No
End
Yes
Re-Engineering > = Retraining
No
Decision = Retraining
Yes
Decision = Re-Engineering
Decision == Retraining
No
Adjust SW
Yes
Adjust PH
Adjusted SW
Adjusted PH

Figure 4.10 The Flowchart Of The Monitoring Process

although it might provide better predictions; but, is likely to consume more resources. Consider the following rule form to identify a retraining request from a set of inputs:

IF MQ is Poor and SW is Medium and PH is short Then Retraining is High.

IF MQ is Poor and SW is Medium and PH is Medium Then Retraining is Medium.

Assuming that medium and short SW and PH are 30 and 15 respectively, in the first rule the retraining decision is high. This implies the necessity of launching a retraining process on the cloud. While in the second rule, the decision is medium, which implies adjusting the length of the PH. These assumptions can be justified by considering that in the first rule the PH is already short and can not be adjusted to a lower value. Meanwhile, adjusting the PH, in this case, to a higher value would likely result in a higher error rate. In addition, the output in the second rule can be justified by considering that both SW and PH are equal; so minimizing the PH and measuring the error rate for the next round would give a finer perception of the model quality;

2) Re-Engineering: in the proposed workflow, the Re-Engineering option is introduced to overcome the performance degradation that can not be treated by the Retraining option. More specifically, the re-engineering refers to the process where the need to switch ML models. For instance, to replace a model with a shorter SW to another model with a longer SW or contrariwise. To elaborate more, the former case is induced to alleviate the consequences of the unexpected and the phenomenal decline in the model performance that has led to very poor model quality. This case is more likely to result from using a model with SW that is shorter than the PH. On the other hand, if the SW is longer from the PH, and the model performance is excellent; the re-engineering request might be initiated to switch to a model with a shorter SW to achieve stabilized resource utilization.

   As shown in figure4.10, the re-engineering process has a higher priority over the retraining. This is can be justified by our assumption expressed by the FL rules. In this assumption, the re-engineering option is likely to be imposed when the MQ is very poor (VP). In this case, the retraining option might be also triggered by some rules, but a higher priority would be given to the re-engineering if they are on similar scales. As shown in Table4.3, the re-engineering option might take three linguistic values namely: Low (L), Medium,(M), and High(H). More specifically, Low implies that no need to run the re-engineering process. While medium implies that the re-engineering will run locally on the edge device by switching models. Finally, High refers to switching models; in addition, sending a re-engineering request notification to the system on the cloud. In this case, new data will be offloaded to the cloud and a feature engineering process will be recommended.

4.5 Experiments

Several experiments were carried out using a time series dataset. In particular, Gams dataset [1]is used. It is composed of indoor air quality sensors' readings including PM10, PM2.5, humidity, temperature, CO2, noise, and VOCs. In our experiments, the concentration was solely on two parameters namely: CO2, and PM2.5. This selection can be justified by the impact of both parameters on the indoor air quality, which eventually has significant consequences on the dwellings well being. Two sets of experiments have been carried out to evaluate the feasibility of our approach. In the first category, the process of model creation and evaluation has been carried out in settings that are similar to the real-time implementation. Where the training dataset is partitioned into segments of time indexation and fed iteratively to the model creation process and several criteria have been recorded. While the second category has been implemented to assess the performance of the monitoring module at the edge and in real time settings. Several criteria have been considered including ML models performance, resources utilization, and data reduction.

4.5.1 Model Creation and Evaluation

In this section some experiments are carried out based on a series of comparative analyses on ML models and their capabilities and limitations, some conclusions regarding the model selection and improvement have been drawn. As previously mentioned three ML algorithms have been explored in this work namely: SVR, MLP, and BN. These models have been created and implemented under the python 3 environment and using Scikit-learn, Pandas, NumPy, and Matplotlib packages.

[1] https://github.com/twairball/gams-dataset

The development process followed in this stage is stimulated from real-time settings. Where our models and their parameters are initially trained using data from the first two days of the train dataset. These data are time series and they are sequentially ordered. Afterwards, these parameters are applied on data within the next 6 hours time horizon and evaluation criteria are computed and recorded. The next step is to update the train dataset by adding the 6 hours evaluation data and retrain and update model parameters in a continuous process. Similarly the evaluation window will be shifted forward in fixed walk-forward time window and updated model parameters are used to make predictions on these unseen data. This iterative process have been configured to run till the loop reaches the end of train dataset. Thus, it has resulted in the creation of set of trained models. Each model is represents by iteration No in the presented figures.

Figures (4.11 - 4.14) show the variations in the values of the evaluation criteria recorded throughout the iterations of our experiments. For each environmental factor, we have listed the results of a comparison evaluation criteria for the 3 used models in two different sliding window configurations, namely 30 and 60 minutes. As shown in figures (4.11 - 4.14) SVR model exposes the maximum value of RMSE in both cases 30 and 60 minutes sliding window. On the other hand, both BN and MLP revealed the minimum RMSE values which demonstrates their efficiency minimizing the loss function, and produce quality predictions. However, the MLP has demonstrated a high RMSE value at the first iteration, but in overall it has stabilized as the model continues retraining and updating its parameters.

Finally, figure 4.15 shows the execution time by model iteration. It is clearly observed that BN has performed with the lowest execution time among all investigated models with both sliding window configurations. This indeed contributes to the scalability of the model when we consider enlarging the dataset used for training. On the other hand, MLP has shown the highest execution time among the selected ML models. While SVR has shown lesser time than MLP but it is still high compared to the BN execution time.

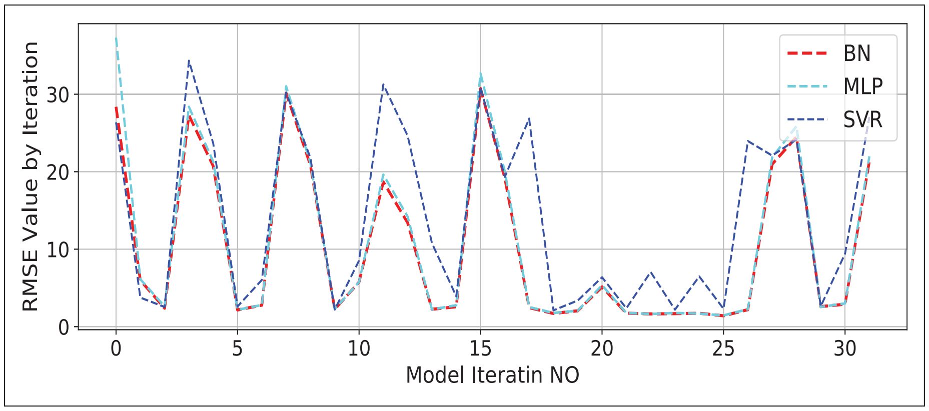


Figure 4.11 RMSE value Per Training Iteration for CO2 and with Sliding window = 30

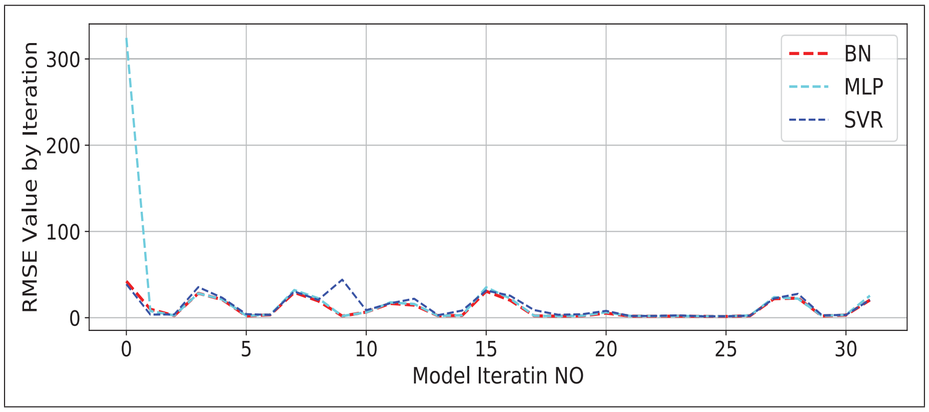


Figure 4.12 RMSE value Per Training Iteration for CO2 and with Sliding window = 60

4.5.2 Edge-Based Evaluation

The main process run on the edge has been derived from modules designed for our proposed approach. The main process receives and collects data from virtual sensors reading from the dataset. Afterward, data are analyzed by the analyzer component and the RMSE is calculated based on the length of the prediction horizon and upon the availability of the ground through data. Then, the RMSE, SW, and PH will be delivered to the FL inference module to detect model quality and apply the corresponding option as well as derive the recommended decision. Raspberry pi3 has been chosen as a hardware platform to serve the proposed modules on the edge. As a lightweight, inexpensive board computer, the Raspberry pi 3 is designed with a competent

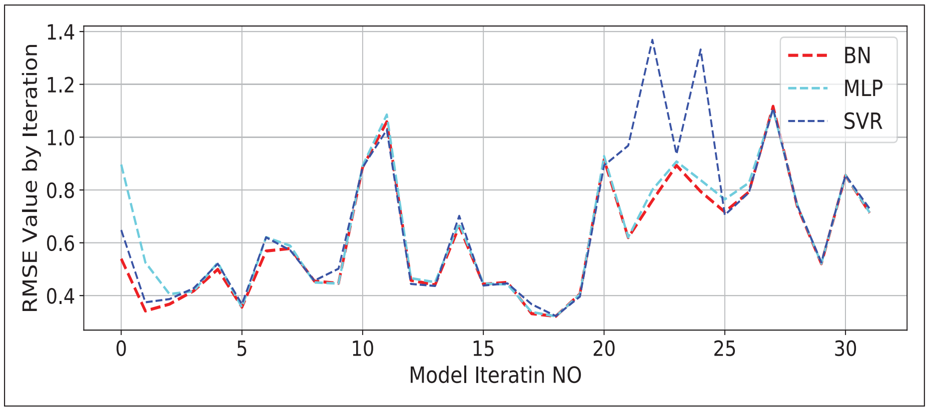


Figure 4.13 RMSE value Per Training Iteration for PM2.5 and with Sliding window = 30

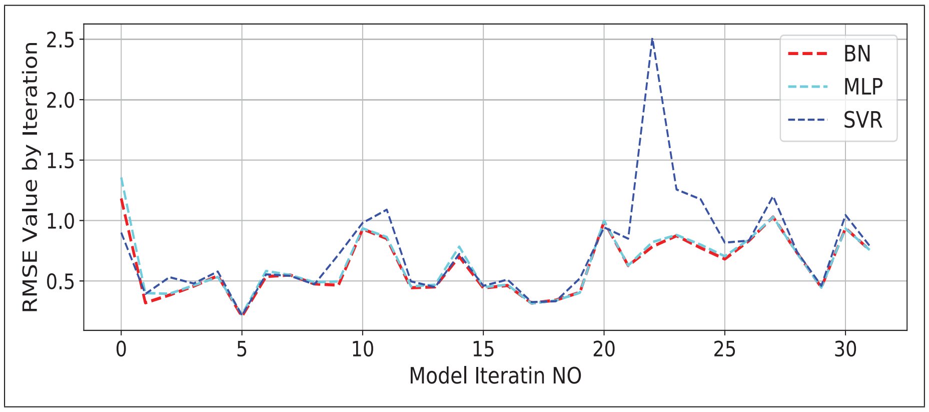


Figure 4.14 RMSE value Per Training Iteration for PM2.5 and with Sliding window = 60

processing abilities in mind compared to other boards and microcontrollers. It possesses a processor speed of 1.44 GHz, 1 GB of RAM, a built-in Wi-Fi, and an Ethernet port. Python 3.7 has been chosen for the development of all the functionalities of the main process unit.

ML models created in previous steps have been deployed in ‘pickle’ format on the edge device and configured to run real-time prediction tasks. To do this, we have streamed data packets received from simulated sensors to the analyzer module. This module has been configured to run two ML models synchronously. Each one of them is applied to generate predictions for a particular parameter (e.g, CO2 and PM2.5). Received data are collected for each parameter and models are set to start generating predictions when the length of vector V = SW. SW is the sliding window used during model training, which in our implementation is set to take one of

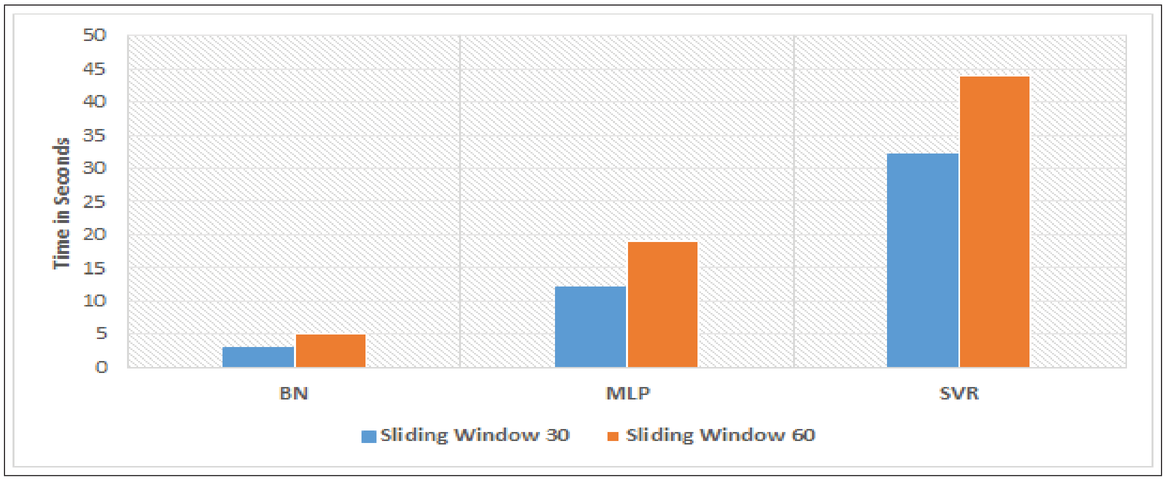


Figure 4.15 Overall Execution Time in two different sliding window configuration

two values namely 30 or 60. While the prediction horizon (PH) is set to has a value that ranges from 15 to 60. This value is adjustable and may increase/decrease upon the reasoning over the obtained RMSE value. Furthermore, this adjustment in both cases could take the value of 10 or 5 based on the length of PH and the value of RMSE, which provides our approach more resilience to respond to variances and degradation in the deployed models performance.

4.5.3 Evaluation of ML models efficiency

All ML models have been evaluated in real-time settings using two implementation configurations namely: With and without the monitoring module. In the latter, we have deployed ML models with fixed SW and PH. While in the former, we have applied the procedures presented in the above-described approach. In both cases, the process started with a model of SW =30 and PH =60. This selection is intended to reveal the efficiency of our approach in a critical use case scenario, where the PH is double the length of the SW. This implies that the RMSE is likely to expose relatively higher values and our aim is to demonstrate how our approach has contributed to the minimization of the RMSE.

To do this, the selected models (SVR, MLP, and BN) have been tested in both aforementioned implementation scenarios. It can be clearly spotted in figures (4.16 - 4.21) that modes configured

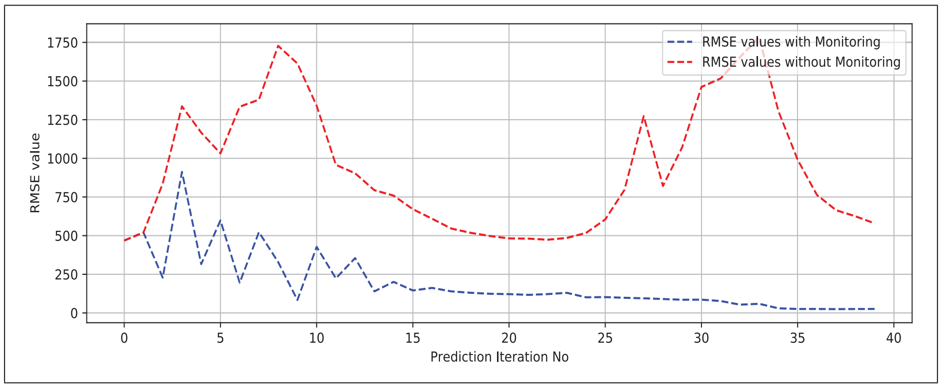


Figure 4.16 RMSE value Per real-time prediction iteration for SVR Model (CO2)

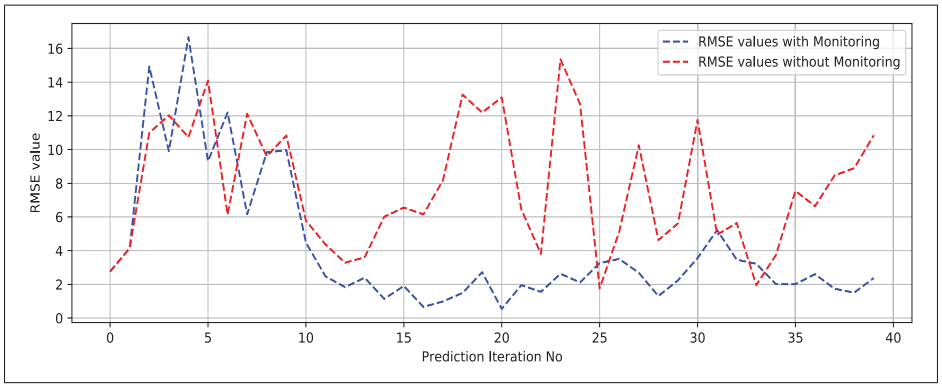


Figure 4.17 RMSE value Per real-time prediction iteration for SVR Model (PM2.5)

to run with the proposed monitoring module has outperformed models without monitoring in most of the evaluation period. In this period, the prediction process has been divided into several iterations; each one of them corresponds to a specific prediction horizon. In both cases, the value of RMSE starts from the same point; but with time, it can be observed that our approach has the advantage of lowering the RMSE gradually and steadily. This will eventually lead to the improvement of the Model quality and the overall performance of the IoT edge-based system that uses its outcomes. The positive impact of the monitoring module varies depending on the performance of the used ML models. As shown in figures 4.16 4.17, the monitoring module has significantly minimized the RMSE value of the SVR. This is also has been confirmed by

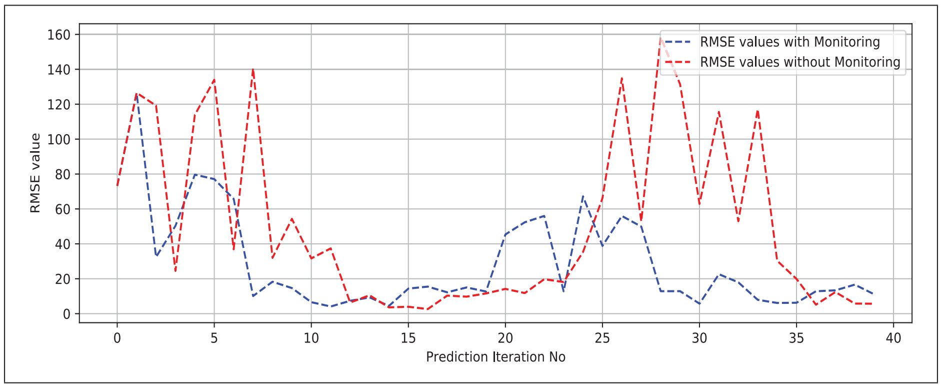


Figure 4.18 RMSE value Per real-time prediction iteration for BN Model (CO2)

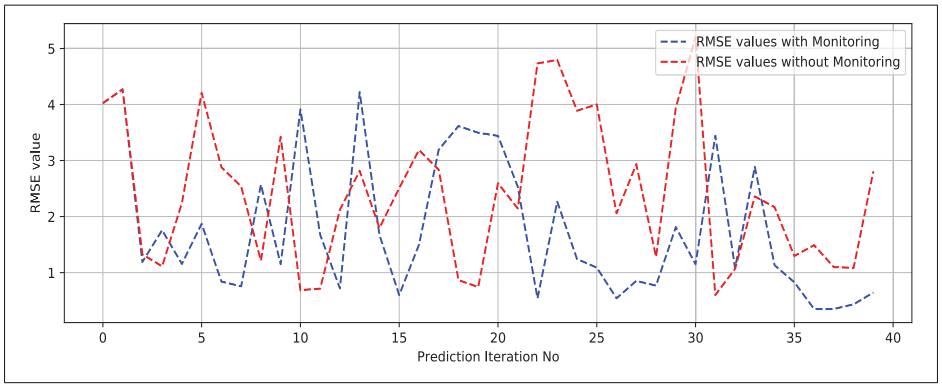


Figure 4.19 RMSE value Per real-time prediction iteration for BN Model (PM2.5)

calculating the average of the recorded RMSE values for each ML model as presented in Table 4.4.

Table 4.4 RMSE Average of each Model Implementation on the Edge

Model	RMSE Average (CO2)		RMSE Average (PM2.5)	
	With Monitoring	Without Monitoring	With Monitoring	Without Monitoring
BN	29.35	51.31	1.79	2.62
MLP	43.41	73.93	1.70	2.92
SVR	174.34	885.618	3.89	7.52

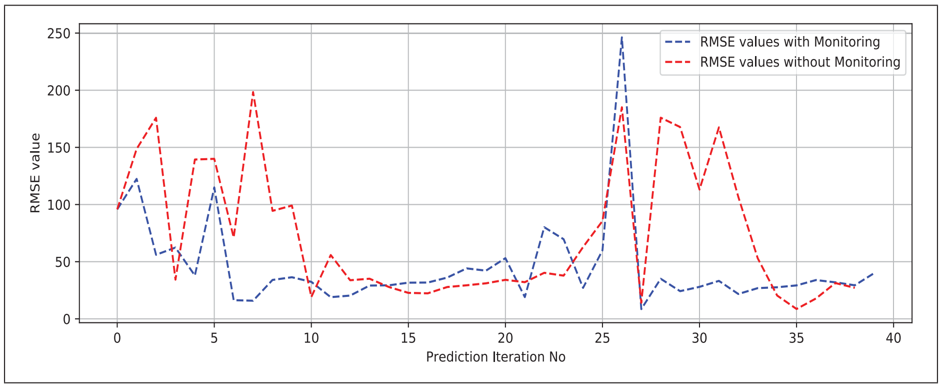


Figure 4.20 RMSE value Per real-time prediction iteration for MLP Model (CO2)

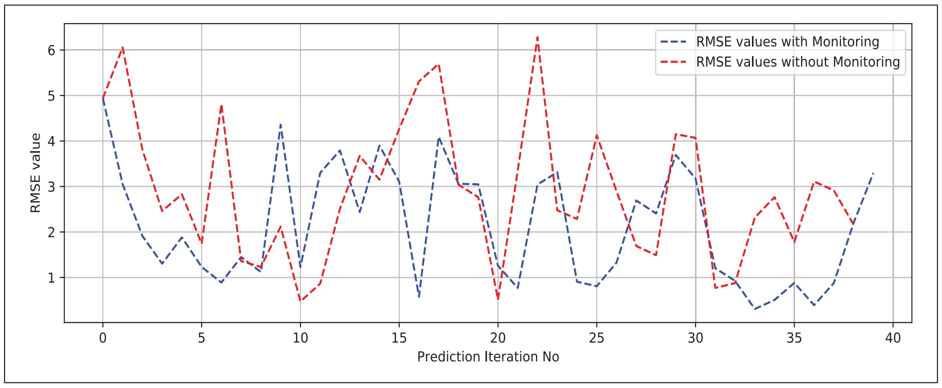


Figure 4.21 RMSE value Per real-time prediction iteration for MLP Model (PM2.5)

4.5.4 Resource Utilization

In this experiment, the CPU and RAM utilization have been considered to assess the impact of the proposed approach and its compatibility with IoT edge devices. The experiment has been carried out in two configurations namely: With and without the monitoring module. While running the former scenario involves executing all three modules consolidating the edge part of the proposed approach. The latter scenario involves executing two modules and excluding the assessment and monitoring module. Figure 4.22 shows the overall CPU utilization recorded per second. It can be observed that there are some peaks in the CPU utilization when using the monitoring module represented by the dashed blue line. These peak periods represent the

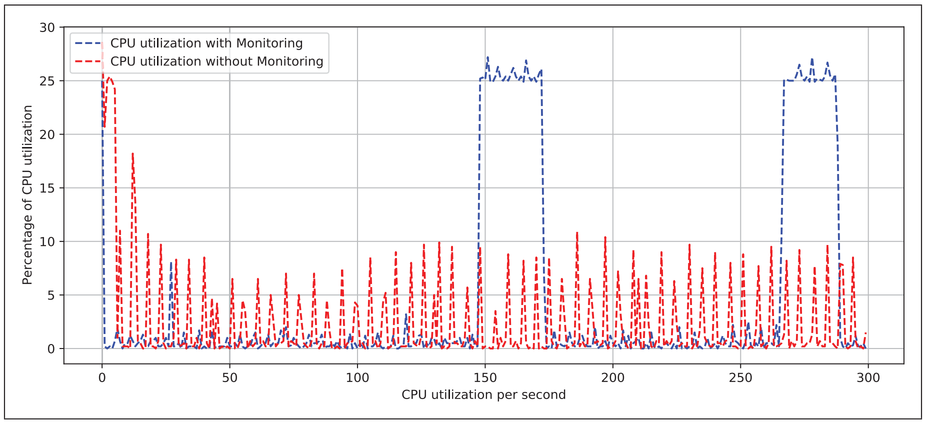


Figure 4.22 CPU utilization

time where the monitoring module is active and performing reasoning tasks upon receiving the input from the analyzer module. Despite these peaks, the monitoring module has significantly contributed to lessen the CPU consumption of the system compared to the CPU utilization without the monitoring module. This is can be justified by the dynamic adjustment mechanism of the SW and PH applied by the monitoring module. On the other hand, figure 4.23 shows the RAM utilization recorded in both mentioned configurations. Although the RAM utilization has exhibited a slight increase as a result of using the monitoring module, the overall RAM consumption has remained within an acceptable range that is likely would not lead to any overhead in the resource consumption.

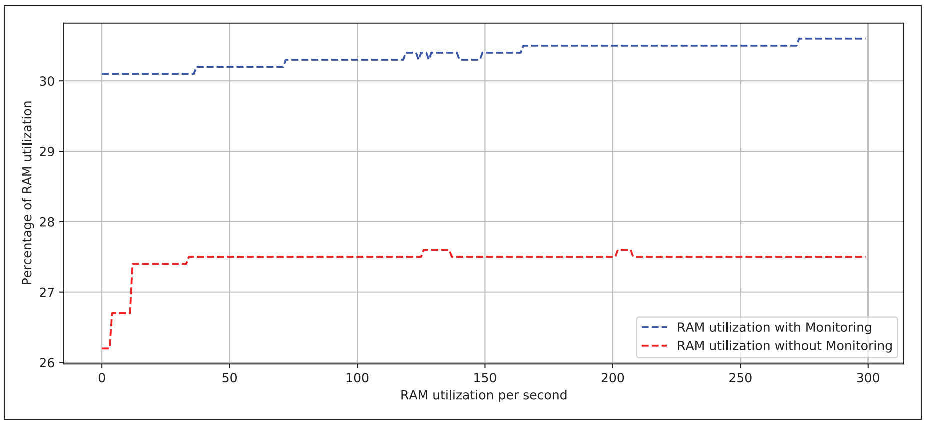


Figure 4.23 RAM utilization

4.5.5 Data Reduction

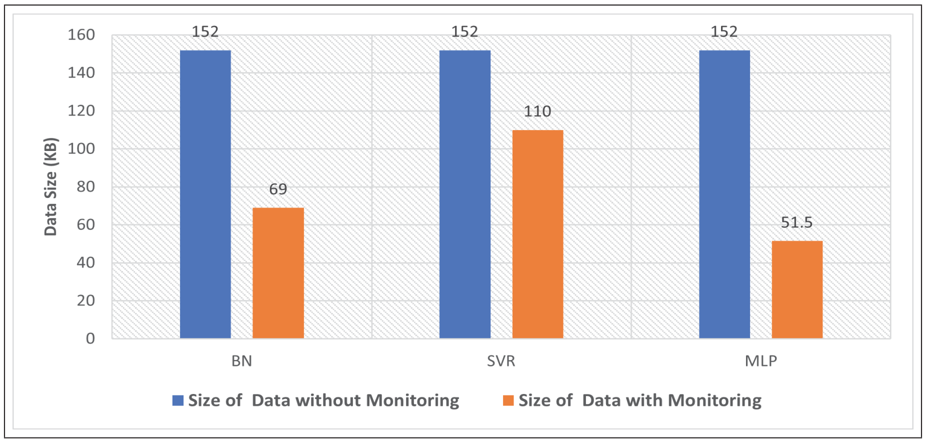


Figure 4.24 Data Reduction in two different configurations

The aim of this experiment is to assess the feasibility of the monitoring module in reducing the amount of data to be transmitted to the cloud for updating the dataset and to maintain model retaining or re-engineering tasks. Further, data reduction is likely to contribute in alleviating network traffic and bandwidth consumption. We streamed a portion of the used dataset with a fixed size (152 KB). It represents CO2 and PM2.5 readings for 4 days. The monitoring module has been configured to filter data based on the obtained value of the model quality (MQ). Mainly MQ takes either one of four linguistic values as shown in figure 4.24. Each one of these linguistic values covers a specific numerical range. In the data reduction use case, the monitoring module is set to identify data points that are combined with (MQ< 'good'). To enable more data reduction MQ can be set with value 'poor' or 'very poor', but our selection is expanded to capture patterns that led to the model performance deterioration. Figure 4.24 shows that using the monitoring approach has reasonably reduced the data when both the BN and MLP are used. While using the SVR model has shown a slight reduction in the data size compared to BN and MLP. This due to the poor performance of the SVR model especially at the beginning of the prediction process and before the impact of the monitoring process is gradually applied. It is worth mentioning that the size of the filtered data is likely to increase especially; in a real implementation scenario, the edge device is connected with more than one sensor node and data will be accumulated with time.

4.6 Conclusion

The incorporation of ML techniques into the IoT realm has paved the way toward the development of innovative solutions applied in several aspects of our daily life. As a recent tendency, IoT edge devices have become a major component in most IoT deployments. These devices have witnessed computing capabilities development that enabled them to run some relatively resource-intensive tasks such as real-time ML analysis based tasks. However, deploying and running ML models in real time and under the dynamic and unstable IoT environment has exposed them to model performance deterioration issues. This issue is mainly induced by the likely frequent change in the statical characteristics of the data generated by IoT sensors that reflect changes in the observed environment. To alleviate this issue, this work proposed a fuzzy logic-based approach for real-time assessment of ML models deployed on IoT edge devices. The proposed approach decomposes this process into two main steps. First, converting the used loose function measure (RMSE) into a crisp value that can be reasoned over by FL rules and generate a model quality criterion. Second, using the model quality as a parameter to determine the required adjustment in order to cop with any model degradation event. The proposed approach has been evaluated in the context of indoor air quality using a dataset with CO2 and PM2.5 readings and using several ML algorithms. The obtained results have demonstrated its feasibility in gradually minimizing the RMSE and improving models' efficiency. Moreover, it has demonstrated its feasibility in reducing data and maintaining balanced use of resources. As a future perspective, more experiments will be considered using an actual sensor testbed, which will enable better evaluation of the impact of using such an approach on the overall workflow. Moreover, to improve the deployment and the scalability of ML-based approach, more interest will be thrown toward investigating the concept of microservices and the trade-offs of applying them in such an environment.

CONCLUSION AND RECOMMENDATIONS

Despite the remarkable deployment of IoT networks and the diversity of their applications, several challenges have hindered the feasibility of unleashing the full potential of such technology. The common feature that stamps most IoT applications is their reliance on sensory data for optimal functionality and service delivery. These data are envisioned as representative of actual events of interest that are taking place in the real world. Thus, in order to enable efficient event detection from these data, it is of high significance for IoT applications to be able to convey raw sensory data to gain meaningful insights. However, several operational factors of IoT systems, which are reflected in the characteristics of the data they generate, have posed more complications in the event detection process.

This thesis has presented work that demonstrates the significance of applying several AI-based techniques to enable data processing solutions on IoT edge devices. The presented solutions are intended to facilitate an efficient event detection process taking into consideration the resource restrictions of a substantial spectrum of IoT edge devices. We first investigated several concepts related to the IoT and studied in detail its architecture, components, and deployment models. Further, the special characteristics of both IoT applications and data were considered. In addition, the concept of edge computing was discussed and the advantages of incorporating it into the IoT realm have been demonstrated. Furthermore, the thesis has shed light on the prominence of SW, ML, and FL techniques and the role they can play in order to maintain balanced and efficient sensory data processing.

To support this perspective, we presented three case studies that represent the main contributions of the presented work. They have been introduced to reveal the feasibility and trade offs of extending such intelligent based solutions to the edge of IoT networks and how AI-based techniques can contribute to full or partial fulfillment of demanding research challenges.

Throughout different stages in this research, several questions have been rendered and several objectives and sub-objectives were identified. The research began by investigating the requirements needed to be met for efficient IoT data processing workflow. This includes latency, network and bandwidth consumption, resource limitations of IoT edge devices, and unstable environments and their impact on the generated data. After, we analyzed the existing literature and solutions and their associated AI-based approaches. Based on this investigation, we started to build our solution.

In the first article (Chapter 2), we have proposed an edge-based IoT data processing solution that targets alleviating the burden of event detection and reactive manipulation. The proposed approach has been designed taking into consideration the vitality of events that have the utmost impact on the observed situation, as well as limited resources on IoT gateways. It is encompassed by two layers for real-time data processing, which are based on the combination of the semantic web and complex event-processing techniques. It is designed to enable platform independent implementation, where regardless of the gateway and the connected sensors specifications, developers can customize their event and semantic rules required for data processing. Moreover, we take into consideration the minimization of data to be processed at the gateway then sent to the cloud.

In the second article (Chapter 3), we have investigated the applicability of ML and FL concepts to promote IoT edge-based solutions performing real-time data analysis. This analysis is intended to support proactive event detection and decision planning processes. The proposed approach is able to predict and identify events from multiple sources of sensory data, then classify them in accordance with a set of FL rules. Classified events in addition to the model credibility criterion, presented in the proposed work, are fed to another FL rules base to facilitate proactively drawing the required action to respond to any anticipated event.

Finally, in the third article (Chapter 4), we have highlighted several factors that impact the performance of ML models performing real-time IoT edge data analysis-based solutions. More specifically, in this work, we have proposed a fuzzy logic-based approach for real-time monitoring and assessment of ML models deployed on IoT edge devices. The proposed approach decomposes this process into two main steps. First, converting the used loose function measure (RMSE) into a crisp value that can be reasoned over by a set of FL rules, which will result in obtaining a model quality parameter. Second, the model quality is used as a parameter to set the course of the adaptation needed to gradually prevent model performance deterioration.

In summary, the proposed solutions were fulfilled after a deep investigation of the related subjects of research and they were supported by in-depth exploration of real-life use cases. Several proofs of concept implementations have been carried out. The obtained results have demonstrated the efficiency of each of the proposed approaches.

Future Research Directions

In this thesis, we have highlighted the vitality of the IoT edge devices as a fundamental component of the IoT networks. More specifically, the functionalities that can be delegated to them to address several challenges resulted from blindly transmitting sensory data to the cloud for further manipulation. The proposed approaches are intended to facilitate the integration of AI-based techniques and IoT edge resource-limited devices to fulfill the requirements of real time data processing and event detection tasks. These approaches have been developed in a successive manner. Thus, each contribution can be conceived as a subsequent step of the preceding stage. Nevertheless, we still believe there are many considerable and interesting suggestions that can be tracked in the future. We highlight some of them below:

- testing with more data sets that represent other more critical use cases such as fire and gas detection. In addition, data sets that contain a more diverse collection of data;

- investigating the applicability of more complex ML algorithms such as LSTM on performing in real time and under resource- limited environments;
- to improve the deployment and the scalability of ML-based applications, more interest will be focused toward investigating the concept of microservices;
- performing more experiments especially using a real sensor test bed, with which a better evaluation of the proposed approach will be obtained.

BIBLIOGRAPHY

Abdulkareem, K. H., Mohammed, M. A., Gunasekaran, S. S., Al-Mhiqani, M. N., Mutlag, A. A., Mostafa, S. A., Ali, N. S. & Ibrahim, D. A. (2019). A review of Fog computing and machine learning: Concepts, applications, challenges, and open issues. *IEEE Access*, 7, 153123–153140.

Adeleke, J. A., Moodley, D., Rens, G. & Adewumi, A. O. (2017). Integrating statistical machine learning in a semantic sensor web for proactive monitoring and control. *Sensors*, 17(4), 807.

Aggarwal, C. C., Ashish, N. & Sheth, A. (2013). The internet of things: A survey from the data-centric perspective. In *Managing and mining sensor data* (pp. 383–428). Springer.

Ahamed, F., Shahrestani, S. & Cheung, H. (2020). Internet of Things and Machine Learning for Healthy Ageing: Identifying the Early Signs of Dementia. *Sensors*, 20(21), 6031.

Akbar, A., Khan, A., Carrez, F. & Moessner, K. (2017). Predictive analytics for complex IoT data streams. *IEEE Internet of Things Journal*, 4(5), 1571–1582.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M. & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4), 2347–2376.

Al-Osta, M., Ahmed, B. & Abdelouahed, G. (2017). A lightweight semantic web-based approach for data annotation on IoT gateways. *Procedia computer science*, 113, 186–193.

Al-Osta, M., Bali, A. & Gherbi, A. (2019). Event driven and semantic based approach for data processing on IoT gateway devices. *Journal of Ambient Intelligence and Humanized Computing*, 10(12), 4663–4678.

Al-Rakhami, M., Alsahli, M., Hassan, M. M., Alamri, A., Guerrieri, A. & Fortino, G. (2018). Cost efficient edge intelligence framework using docker containers. *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, pp. 800–807.

Alghamdi, T. A., Lasebae, A. & Aiash, M. (2013). Security analysis of the constrained application protocol in the Internet of Things. *Second international conference on future generation communication technologies (FGCT 2013)*, pp. 163–168.

AlSuwaidan, L. (2020). The role of data management in the Industrial Internet of Things. *Concurrency and Computation: Practice and Experience*, e6031.

Alves, J. M., Honorio, L. M. & Capretz, M. A. (2019). ML4IoT: A Framework to Orchestrate Machine Learning Workflows on Internet of Things Data. *IEEE Access*, 7, 152953–152967.

Ameer, S., Shah, M. A., Khan, A., Song, H., Maple, C., Islam, S. U. & Asghar, M. N. (2019). Comparative analysis of machine learning techniques for predicting air quality in smart cities. *IEEE Access*, 7, 128325–128338.

Ansari, D. B., Rehman, A. & Ali, R. (2018). Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP. *International Journal of Computer Applications*, 975, 8887.

Arce, J. M. M. & Macabebe, E. Q. B. (2019). Real-Time Power Consumption Monitoring and Forecasting Using Regression Techniques and Machine Learning Algorithms. *2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS)*, pp. 135–140.

Atzori, L., Iera, A. & Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15), 2787–2805.

Awad, M. & Khanna, R. (2015). Support vector regression. In *Efficient learning machines* (pp. 67–80). Springer.

Badidi, E., Mahrez, Z. & Sabir, E. (2020). Fog Computing for Smart Cities' Big Data Management and Analytics: A Review. *Future Internet*, 12(11), 190.

Bali, A., Al-Osta, M. & Abdelouahed, G. (2017). An ontology-based approach for IoT data processing using semantic rules. *International SDL Forum*, pp. 61–79.

Bansal, K., Mittal, K., Ahuja, G., Singh, A. & Gill, S. S. (2020). DeepBus: Machine learning based real time pothole detection system for smart transportation using IoT. *Internet Technology Letters*, 3(3), e156.

Barnaghi, P., Wang, W., Henson, C. & Taylor, K. (2012). Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 8(1), 1–21.

Boguda, S. K. (2020). The Revolutionary Paradigm of Enterprise Applications through the Lens of Data and Technology.

Bontempi, G., Taieb, S. B. & Le Borgne, Y.-A. (2012). Machine learning strategies for time series forecasting. *European business intelligence summer school*, pp. 62–77.

Borgia, E. (2014). The Internet of Things vision: Key features, applications and open issues. *Computer Communications*, 54, 1–31.

Braei, M. & Wagner, S. (2020). Anomaly detection in univariate time-series: A survey on the state-of-the-art. *arXiv preprint arXiv:2004.00433*.

B.Tao. (2013). Vivante Internet of Things (IoT) Solutions.

Burns, A. (1993). *Preemptive priority based scheduling: An appropriate engineering approach*. Citeseer.

Carrez, F. (2009). TD 3.2–Reference Architecture. SENSEI, Public Deliverable D. 3.2, 2009.

Carvalho, G., Cabral, B., Pereira, V. & Bernardino, J. (2020). Computation offloading in Edge Computing environments using Artificial Intelligence techniques. *Engineering Applications of Artificial Intelligence*, 95, 103840.

Casagras, E. (2009). Casagras final report: Rfid and the inclusive model for the internet of things. *EU FP7 Project CASAGRAS*.

Chai, T. & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. *Geoscientific model development*, 7(3), 1247–1250.

Chen, C., Twycross, J. & Garibaldi, J. M. (2017). A new accuracy measure based on bounded relative error for time series forecasting. *PloS one*, 12(3), e0174202.

Chen, C. Y., Fu, J. H., Wang, P.-F., Jou, E. & Feng, M.-W. (2014). Complex event processing for the internet of things and its applications. *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pp. 1144–1149.

Chiesa, G., Di Vita, D., Ghadirzadeh, A., Herrera, A. H. M. & Rodriguez, J. C. L. (2020). A fuzzy-logic IoT lighting and shading control system for smart buildings. *Automation in Construction*, 120, 103397.

Christophe, B. (2012). Managing massive data of the internet of things through cooperative semantic nodes. *International Journal of Semantic Computing*, 6(04), 389–408.

Chun, S., Seo, S., Oh, B. & Lee, K.-H. (2015). Semantic description, discovery and integration for the Internet of Things. *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pp. 272–275.

CISCO. (2020, March, 9). Cisco Annual Internet Report (2018–2023 [Format]. Retrieved from https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html.

Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A. et al. (2012). The SSN ontology of the W3C semantic sensor network incubator group. *Journal of Web Semantics*, 17, 25–32.

Cui, L., Yang, S., Chen, F., Ming, Z., Lu, N. & Qin, J. (2018). A survey on application of machine learning for Internet of Things. *International Journal of Machine Learning and Cybernetics*, 9(8), 1399–1417.

Da Rosa Righi, R., Goldschmidt, G., Kunst, R., Deon, C. & da Costa, C. A. (2020). Towards combining data prediction and internet of things to manage milk production on dairy cows. *Computers and Electronics in Agriculture*, 169, 105156.

Dautov, R., Distefano, S., Bruneo, D., Longo, F., Merlino, G. & Puliafito, A. (2018). Data agility through clustered edge computing and stream processing. *Concurrency and Computation: Practice and Experience*, e5093.

Davis, N. (2019, July, 8). Indoor carbon dioxide levels could be a health hazard scientists warn [Format]. Retrieved from https://www.theguardian.com/environment/2019/jul/08/indoor-carbon-dioxide-levels-could-be-a-health-hazard-scientists-warn.

de Lima Pinto, E. M., Lachowski, R., Pellenz, M. E., Penna, M. C. & Souza, R. D. (2018). A machine learning approach for detecting spoofing attacks in wireless sensor networks. *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 752–758.

de Mattos Neto, P. S., Madeiro, F., Ferreira, T. A. & Cavalcanti, G. D. (2014). Hybrid intelligent system for air quality forecasting using phase adjustment. *Engineering Applications of Artificial Intelligence*, 32, 185–191.

Delicato, F. C., Pires, P. F. & Batista, T. (2013). *Middleware solutions for the Internet of Things*. Springer.

Deng, S., Xiang, Z., Zhao, P., Taheri, J., Gao, H., Yin, J. & Zomaya, A. Y. (2020). Dynamical Resource Allocation in Edge for Trustable Internet-of-Things Systems: A Reinforcement Learning Method. *IEEE Transactions on Industrial Informatics*, 16(9), 6103–6113.

Desai, P., Sheth, A. & Anantharam, P. (2015). Semantic gateway as a service architecture for iot interoperability. *Mobile Services (MS), 2015 IEEE International Conference on*, pp. 313–319.

Dillon, T., Chang, E., Singh, J. & Hussain, O. (2012). Semantics of Cyber-Physical Systems. *International Conference on Intelligent Information Processing*, pp. 3–12.

Ding, Z., Yang, Q. & Wu, H. (2011). Massive heterogeneous sensor data management in the Internet of Things. *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, pp. 100–108.

Dubey, S., Singh, P., Yadav, P. & Singh, K. K. (2020). Household Waste Management System Using IoT and Machine Learning. *Procedia Computer Science*, 167, 1950–1959.

El Emary, I. M. & Ramakrishnan, S. (2013). *Wireless sensor networks: from theory to applications*. CRC press.

El Kaed, C., Khan, I., Van Den Berg, A., Hossayni, H. & Saint-Marcel, C. (2017). SRE: Semantic rules engine for the industrial Internet-of-Things gateways. *IEEE Transactions on Industrial Informatics*, 14(2), 715–724.

EPA. (2017, November, 6). Volatile Organic Compounds' Impact on Indoor Air Quality [Format]. Retrieved from https://www.epa.gov/indoor-air-quality-iaq/volatile-organic-compounds-impact-indoor-air-quality.

Ericsson. (2015, June, 1). Ericsson Mobility Report [Format]. Retrieved from https://www.epressi.com/media/userfiles/13896/1433253550/ericsson_mobility_report_2015_06.pdf.

Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. *CISCO. White Paper*, 1(2011), 1–11.

Fang, L., Ge, C., Zu, G., Wang, X., Ding, W., Xiao, C. & Zhao, L. (2019). A Mobile Edge Computing Architecture for Safety in Mining Industry. *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, pp. 1494–1498.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4), 1–37.

Gartner. (2013, October, 7). Economic Impact of the Internet of Things [Format]. Retrieved from https://www.businesswire.com/news/home/20131007006209/en/Gartner-Says-ItE28099s-the-Beginning-of-a-New-Era-The-Digital-Industrial-Economy.

Giordani, I. & Archetti, F. (2016). Models and architectures for emergency management. *Journal of Ambient Intelligence and Humanized Computing*, 1(8), 1–8.

Goldstein, M. (2008). Carbon monoxide poisoning. *Journal of Emergency Nursing*, 34(6), 538–542.

Goyal, R., Chandra, P. & Singh, Y. (2014). Suitability of KNN regression in the development of interaction based software fault prediction models. *Ieri Procedia*, 6, 15–21.

Gozuoglu, A., Ozgonenel, O. & Karagol, S. (2019). Fuzzy Logic Based Low Cost Smart Home Application. *2019 11th International Conference on Electrical and Electronics Engineering (ELECO)*, pp. 64–68.

Gubbi, J., Buyya, R., Marusic, S. & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7), 1645–1660.

Guillén-Navarro, M. A., Martínez-España, R., López, B. & Cecilia, J. M. (2021). A high-performance IoT solution to reduce frost damages in stone fruits. *Concurrency and Computation: Practice and Experience*, 33(2), e5299.

Guinard, D., Trifa, V., Mattern, F. & Wilde, E. (2011). From the internet of things to the web of things: Resource-oriented architecture and best practices. In *Architecting the Internet of things* (pp. 97–129). Springer.

Gupta, B. & Quamara, M. (2020). An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols. *Concurrency and Computation: Practice and Experience*, 32(21), e4946.

Gyrard, A. (2013). An architecture to aggregate heterogeneous and semantic sensed data. *Extended Semantic Web Conference*, pp. 697–701.

Hachem, S., Teixeira, T. & Issarny, V. (2011). Ontologies for the internet of things. *Proceedings of the 8th Middleware Doctoral Symposium*, pp. 3.

Haller, S. (2010). The things in the internet of things. *Poster at the (IoT 2010). Tokyo, Japan, November*, 5(8), 26–30.

Hawedi, M., Talhi, C. & Boucheneb, H. (2018). Multi-tenant intrusion detection system for public cloud (MTIDS). *The Journal of Supercomputing*, 74(10), 5199–5230.

Hillar, G. C. (2017). *MQTT Essentials-A lightweight IoT protocol*. Packt Publishing Ltd.

Hota, H., Handa, R. & Shrivas, A. (2017). Time series data prediction using sliding window based RBF neural network. *International Journal of Computational Intelligence Research*, 13(5), 1145–1156.

IDC. (2019). The Growth in Connected IoT Devices Is Expected to Generate 79.4ZB of Data in 2025.

Imdoukh, M., Ahmad, I. & Alfailakawi, M. G. (2019). Machine learning-based auto-scaling for containerized applications. *Neural Computing and Applications*, 1–16.

Iram, T., Shamsi, J., Alvi, U., ur Rahman, S. & Maaz, M. (2019). Controlling Smart-City Traffic using Machine Learning. *2019 International Conference on Frontiers of Information Technology (FIT)*, pp. 203–2035.

Jagannath, J., Polosky, N., Jagannath, A., Restuccia, F. & Melodia, T. (2019). Machine learning for wireless communications in the Internet of Things: A comprehensive survey. *Ad Hoc Networks*, 93, 101913.

Jain, K. & Mohapatra, S. (2019). Taxonomy of edge computing: Challenges, opportunities, and data reduction methods. In *Edge Computing* (pp. 51–69). Springer.

Janjua, Z. H., Vecchio, M., Antonini, M. & Antonelli, F. (2019). IRESE: An intelligent rare-event detection system using unsupervised learning on the IoT edge. *Engineering Applications of Artificial Intelligence*, 84, 41–50.

Jara, A. J., Olivieri, A. C., Bocchi, Y., Jung, M., Kastner, W. & Skarmeta, A. F. (2014). Semantic web of things: an analysis of the application semantics for the iot moving towards the iot convergence. *International Journal of Web and Grid Services*, 10(2-3), 244–272.

Jia, X., Feng, Q., Fan, T. & Lei, Q. (2012). RFID technology and its applications in Internet of Things (IoT). *2012 2nd international conference on consumer electronics, communications and networks (CECNet)*, pp. 1282–1285.

Johnston, S. J., Basford, P. J., Perkins, C. S., Herry, H., Tso, F. P., Pezaros, D., Mullins, R. D., Yoneki, E., Cox, S. J. & Singer, J. (2018). Commodity single board computer clusters and their applications. *Future Generation Computer Systems*, 89, 201–212.

Kapitanova, K., Son, S. H. & Kang, K.-D. (2012). Using fuzzy logic for robust event detection in wireless sensor networks. *Ad Hoc Networks*, 10(4), 709–722.

Keskisärkkä, R. (2017). *Towards Semantically Enabled Complex Event Processing*. Linköping University Electronic Press.

Khan, I., Jafrin, R., Errounda, F. Z., Glitho, R., Crespi, N., Morrow, M. & Polakos, P. (2015). A data annotation architecture for semantic applications in virtualized wireless sensor networks. *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pp. 27–35.

Khan, W. Z., Ahmed, E., Hakak, S., Yaqoob, I. & Ahmed, A. (2019). Edge computing: A survey. *Future Generation Computer Systems*, 97, 219–235.

Kotis, K. & Katasonov, A. (2012). Semantic interoperability on the web of things: The semantic smart gateway framework. *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, pp. 630–635.

Kozma, L. (2008). k Nearest Neighbors algorithm (kNN). *Helsinki University of Technology*.

Kromkowski, P., Li, S., Zhao, W., Abraham, B., Osborne, A. & Brown, D. E. (2019). Evaluating statistical models for network traffic anomaly detection. *2019 Systems and Information Engineering Design Symposium (SIEDS)*, pp. 1–6.

Kumar, P. M. & Gandhi, U. D. (2018). A novel three-tier Internet of Things architecture with machine learning algorithm for early detection of heart diseases. *Computers & Electrical*

Engineering, 65, 222–235.

Lavanya, G., Rani, C. & Ganeshkumar, P. (2019). An automated low cost IoT based Fertilizer Intimation System for smart agriculture. *Sustainable Computing: Informatics and Systems*.

Le-Tuan, A., Hayes, C., Hauswirth, M. & Le-Phuoc, D. (2020). Pushing the Scalability of RDF Engines on IoT Edge Devices. *Sensors*, 20(10), 2788.

Li, Y., Pang, W., Sun, C., Zhou, Q., Lin, Z., Chang, Y., Li, Q., Zhang, M. & Duan, X. (2019). Smartphone-enabled aerosol particle analysis device. *IEEE Access*, 7, 101117–101124.

Lu, H., He, X., Du, M., Ruan, X., Sun, Y. & Wang, K. (2020). Edge QoE: Computation offloading with deep reinforcement learning for Internet of Things. *IEEE Internet of Things Journal*.

Ma, M., Wang, P. & Chu, C.-H. (2013). Data management for internet of things: Challenges, approaches and opportunities. *2013 IEEE International conference on green computing and communications and IEEE Internet of Things and IEEE cyber, physical and social computing*, pp. 1144–1151.

Mad Saad, S., Andrew, A. M., Md Shakaff, A. Y., Mat Dzahir, M. A., Hussein, M., Mohamad, M. & Ahmad, Z. A. (2017). Pollutant recognition based on supervised machine learning for indoor air quality monitoring systems. *Applied Sciences*, 7(8), 823.

Maimon, O. & Rokach, L. (2005). Data mining and knowledge discovery handbook.

Maksimović, M., Vujović, V. & Milošević, V. (2014). Fuzzy logic and wireless sensor networks–a survey. *Journal of Intelligent & Fuzzy Systems*, 27(2), 877–890.

Manyika, J. (2015). *The Internet of Things: Mapping the value beyond the hype*.

Manyika, J., Chui, M., Bisson, P., Woetzel, J., Dobbs, R., Bughin, J. & Aharon, D. (2015). Unlocking the Potential of the Internet of Things. *McKinsey Global Institute*.

Marques, G. & Pitarma, R. (2018). Indoor air quality monitoring for enhanced healthy buildings. In *Indoor Environmental Quality*. IntechOpen.

Martínez, A., Cañibano, E. & Romo, J. (2020). Analysis of Low Cost Communication Technologies for V2I Applications. *Applied Sciences*, 10(4), 1249.

Mashal, I., Alsaryrah, O., Chung, T.-Y., Yang, C.-Z., Kuo, W.-H. & Agrawal, D. P. (2015). Choices for interaction with things on Internet and underlying issues. *Ad Hoc Networks*, 28, 68–90.

Mathew, S. S., Atif, Y., Sheng, Q. Z. & Maamar, Z. (2013). The web of things-challenges and enabling technologies. In *Internet of things and inter-cooperative computational technologies for collective intelligence* (pp. 1–23). Springer.

Mehdiyev, N., Enke, D., Fettke, P. & Loos, P. (2016). Evaluating forecasting methods by considering different accuracy measures. *Procedia Computer Science*, 95, 264–271.

Mehmood, K., Saifullah, M. I. & Abrar, M. M. (2020). Can exposure to PM2. 5 particles increase the incidence of coronavirus disease 2019 (COVID-19)? *The Science of the Total Environment*.

Merenda, M., Porcaro, C. & Iero, D. (2020). Edge Machine Learning for AI-Enabled IoT Devices: A Review. *Sensors*, 20(9), 2533.

Meyer, S., Ruppen, A. & Hilty, L. (2015). The things of the internet of things in BPMN. *International conference on advanced information systems engineering*, pp. 285–297.

Mittal, K., Jain, A., Vaisla, K. S., Castillo, O. & Kacprzyk, J. (2020). A comprehensive review on type 2 fuzzy logic applications: Past, present and future. *Engineering Applications of Artificial Intelligence*, 95, 103916.

Mliki, H., Kaceam, A. H. & Chaari, L. (2019). Intrusion Detection Study and Enhancement Using Machine Learning. *International Conference on Risks and Security of Internet and Systems*, pp. 263–278.

Molinara, M., Ferdinandi, M., Cerro, G., Ferrigno, L. & Massera, E. (2020). An end to end indoor air monitoring system based on machine learning and SENSIPLUS platform. *IEEE Access*, 8, 72204–72215.

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

Moon, J., Kum, S. & Lee, S. (2019). A heterogeneous IoT data analysis framework with collaboration of edge-cloud computing: Focusing on indoor PM10 and PM2. 5 status prediction. *Sensors*, 19(14), 3038.

Mora, H., Signes-Pont, M. T., Gil, D. & Johnsson, M. (2018). Collaborative working architecture for IoT-based applications. *Sensors*, 18(6), 1676.

Munir, J. (2016). State-of-the-art of Internet of Things ontologies. *Tech. Univ. Berlin*.

Munir, M., Siddiqui, S. A., Dengel, A. & Ahmed, S. (2018). DeepAnT: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7, 1991–2005.

Murshed, M., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G. & Hussain, F. (2019). Machine learning at the network edge: A survey. *arXiv preprint arXiv:1908.00080*.

Negash, B., Rahmani, A. M., Westerlund, T., Liljeberg, P. & Tenhunen, H. (2016). LISA 2.0: lightweight internet of things service bus architecture using node centric networking. *Journal of Ambient Intelligence and Humanized Computing*, 7(3), 305–319.

Palevi, B. R. P. D., Rivai, M. & Purwanto, D. (2019). Fuzzy Logic-Based Wet Scrubber to Control Air Pollutant. *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pp. 74–79.

Papcun, P., Kajati, E., Cupkova, D., Mocnej, J., Miskuf, M. & Zolotova, I. (2020). Edge-enabled IoT gateway criteria selection and evaluation. *Concurrency and Computation: Practice and Experience*, 32(13), e5219.

Parmezan, A. R. S., Souza, V. M. & Batista, G. E. (2019). Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. *Information sciences*, 484, 302–337.

Patel, P., Kattepur, A., Cassou, D. & Bouloukakis, G. (2013). Evaluating the Ease of Application Development for the Internet of Things.

Paul, D., Chakraborty, T., Datta, S. K. & Paul, D. (2018). IoT and machine learning based prediction of smart building indoor temperature. *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, pp. 1–6.

Poongodi, T., Rathee, A., Indrakumari, R. & Suresh, P. (2020). IoT Sensing Capabilities: Sensor Deployment and Node Discovery, Wearable Sensors, Wireless Body Area Network (WBAN), Data Acquisition. In *Principles of Internet of Things (IoT) Ecosystem: Insight Paradigm* (pp. 127–151). Springer.

Purich, P. (2011). Oracle Complex Event Processing EPL Language Reference, 11g Release 1 (11.1. 1.4. 0) E14304-02.

Rahman, H. & Hussain, M. I. (2019). A comprehensive survey on semantic interoperability for Internet of Things: State-of-the-art and research challenges. *Transactions on Emerging Telecommunications Technologies*, e3902.

Rahman, H. & Rahmani, R. (2018). Enabling distributed intelligence assisted future internet of things controller (fitc). *Applied computing and informatics*, 14(1), 73–87.

Rajith, A., Soki, S. & Hiroshi, M. (2018). Real-time optimized HVAC control system on top of an IoT framework. *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 181–186.

Ranjan, K. G., Tripathy, D. S., Prusty, B. R. & Jena, D. (2021). An improved sliding window prediction-based outlier detection and correction for volatile time-series. *International Journal of Numerical Modelling: Electronic Networks, Devices and Fields*, 34(1), e2816.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A. & Clarke, S. (2015). Middleware for internet of things: a survey. *IEEE Internet of things journal*, 3(1), 70–95.

Rose, K., Eldridge, S. & Chapin, L. (2015). The internet of things: An overview–Understanding the issues and challenges of a more connected world. The Internet Society (ISOC). October.

Ruiz-Sarmiento, J.-R., Monroy, J., Moreno, F.-A., Galindo, C., Bonelo, J.-M. & Gonzalez-Jimenez, J. (2020). A predictive model for the maintenance of industrial machinery in the context of industry 4.0. *Engineering Applications of Artificial Intelligence*, 87, 103289.

Ruta, M., Scioscia, F. & Di Sciascio, E. (2012). Enabling the Semantic Web of Things: framework and architecture. *2012 IEEE Sixth International Conference on Semantic Computing*, pp. 345–347.

Samie, F., Bauer, L. & Henkel, J. (2019). From cloud down to things: An overview of machine learning in internet of things. *IEEE Internet of Things Journal*, 6(3), 4921–4934.

Sarabia-Jácome, D., Lacalle, I., Palau, C. E. & Esteve, M. (2019). Efficient Deployment of Predictive Analytics in Edge Gateways: Fall Detection Scenario. *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pp. 41–46.

Sebastiani, P., Abad, M. M. & Ramoni, M. F. (2009). Bayesian networks. In *Data mining and knowledge discovery handbook* (pp. 175–208). Springer.

SECTOR, S. & ITU, O. (2012). Series y: Global information infrastructure, internet protocol aspects and next-generation networks next generation networks–frameworks and functional architecture models. *International Telecommunication Union, Geneva, Switzerland, Recommendation ITU-T Y*, 2060.

Seydoux, N., Drira, K., Hernandez, N. & Monteil, T. (2018). Towards cooperative semantic computing: a distributed reasoning approach for fog-enabled swot. *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pp. 407–425.

Sezer, O. B., Dogdu, E. & Ozbayoglu, A. M. (2017). Context-aware computing, learning, and big data in internet of things: a survey. *IEEE Internet of Things Journal*, 5(1), 1–27.

Sharma, S., Chandra, M. & Kota, S. H. (2020). Health Effects Associated with PM 2.5: a Systematic Review. *Current Pollution Reports*, 1–23.

Sheltami, T. R., Bala, A. & Shakshuki, E. M. (2016). Wireless sensor networks for leak detection in pipelines: a survey. *Journal of Ambient Intelligence and Humanized Computing*, 7(3), 347–356.

Shemshadi, A., Sheng, Q. Z., Zhang, W. E., Sun, A., Qin, Y. & Yao, L. (2016). Searching for the Internet of Things on the Web: Where it is and what it looks like. *arXiv preprint arXiv:1607.06884*.

Shi, W., Cao, J., Zhang, Q., Li, Y. & Xu, L. (2016). Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5), 637–646.

Shukla, P. (2017). Ml-ids: A machine learning approach to detect wormhole attacks in internet of things. *2017 Intelligent Systems Conference (IntelliSys)*, pp. 234–240.

Shukla, S., Hassan, M. F., Jung, L. T., Awang, A. & Khan, M. K. (2019). A 3-tier architecture for network latency reduction in healthcare internet-of-things using fog computing and machine learning. *Proceedings of the 2019 8th International Conference on Software and Computer Applications*, pp. 522–528.

Smrithy, G., Balakrishnan, R. & Sivakumar, N. (2019). Anomaly detection using dynamic sliding window in wireless body area networks. In *Data science and big data analytics* (pp. 99–108). Springer.

Steinwart, I. & Christmann, A. (2008). *Support vector machines*. Springer Science & Business Media.

Stojmenovic, I., Wen, S., Huang, X. & Luan, H. (2016). An overview of fog computing and its security issues. *Concurrency and Computation: Practice and Experience*, 28(10), 2991–3005.

Su, X., Li, P., Flores, H., Riekki, J., Liu, X., Li, Y. & Prehofer, C. (2017). Transferring remote ontologies to the edge of Internet of Things systems. *International Conference on Green, Pervasive, and Cloud Computing*, pp. 538–552.

Su, X., Li, P., Riekki, J., Liu, X., Kiljander, J., Soininen, J.-P., Prehofer, C., Flores, H. & Li, Y. (2018). Distribution of semantic reasoning on the edge of internet of things. *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–9.

Syafrudin, M., Fitriyani, N. L., Alfian, G. & Rhee, J. (2019). An affordable fast early warning system for edge computing in assembly line. *Applied Sciences*, 9(1), 84.

Szilagyi, I. & Wira, P. (2016). Ontologies and Semantic Web for the Internet of Things-a survey. *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*, pp. 6949–6954.

Taieb, S. B., Hyndman, R. J. et al. (2012). *Recursive and direct multi-step forecasting: the best of both worlds*. Citeseer.

Taneja, M., Jalodia, N. & Davy, A. (2019). Distributed decomposed data analytics in fog enabled IoT deployments. *IEEE Access*, 7, 40969–40981.

Taud, H. & Mas, J. (2018). Multilayer perceptron (MLP). In *Geomatic Approaches for Modeling Land Change Scenarios* (pp. 451–455). Springer.

Tschofenig, H., Arkko, J., Thaler, D. & McPherson, D. (2015). Architectural considerations in smart object networking. Tech. *Internet Architecture Board*.

Turgut, Z., Üstebay, S., Ali Aydın, M., Gürkaş Aydın, G. Z. & Sertbaş, A. (2019). Performance analysis of machine learning and deep learning classification methods for indoor localization in Internet of things environment. *Transactions on Emerging Telecommunications Technologies*, 30(9), e3705.

Vafaeipour, M., Rahbari, O., Rosen, M. A., Fazelpour, F. & Ansarirad, P. (2014). Application of sliding window technique for prediction of wind velocity time series. *International Journal of Energy and Environmental Engineering*, 5(2), 1–7.

Wang, C., Gill, C. & Lu, C. (2020a). Adaptive Data Replication in Real-Time Reliable Edge Computing for Internet of Things. *2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI)*, pp. 128–134.

Wang, X., Wang, C., Li, X., Leung, V. C. & Taleb, T. (2020b). Federated deep reinforcement learning for internet of things with decentralized cooperative edge caching. *IEEE Internet of Things Journal*.

World Health Organization, C. T. F. (2013). Health effects of particulate matter. 1–20.

Wu, M., Lu, T.-J., Ling, F.-Y., Sun, J. & Du, H.-Y. (2010). Research on the architecture of Internet of Things. *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, 5, V5–484.

Xu, T., Han, G., Qi, X., Du, J., Lin, C. & Shu, L. (2020). A Hybrid Machine Learning Model for Demand Prediction of Edge-Computing based Bike Sharing System Using Internet of Things. *IEEE Internet of Things Journal*.

Yao, H., Fu, D., Zhang, P., Li, M. & Liu, Y. (2018). MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system. *IEEE Internet of Things Journal*, 6(2), 1949–1959.

Ye, J., Dasiopoulou, S., Stevenson, G., Meditskos, G., Kontopoulos, E., Kompatsiaris, I. & Dobson, S. (2015). Semantic web technologies in pervasive computing: A survey and research roadmap. *Pervasive and Mobile Computing*, 23, 1–25.

Yu, W., Liang, F., He, X., Hatcher, W. G., Lu, C., Lin, J. & Yang, X. (2017). A survey on the edge computing for the Internet of Things. *IEEE access*, 6, 6900–6919.

Zantalis, F., Koulouras, G., Karabetsos, S. & Kandris, D. (2019). A review of machine learning and IoT in smart transportation. *Future Internet*, 11(4), 94.

Zeng, D., Guo, S. & Cheng, Z. (2011). The web of things: A survey. *JCM*, 6(6), 424–438.

Zhang, D. & Woo, S. S. (2020). Real Time Localized Air Quality Monitoring and Prediction Through Mobile and Fixed IoT Sensing Network. *IEEE Access*, 8, 89584–89594.

Zhou, H., Wang, H., Li, X. & Leung, V. C. (2018). A survey on mobile data offloading technologies. *IEEE Access*, 6, 5101–5111.