UNIVERSITY OF CALABRIA

DOCTORAL THESIS

Technologies and IoT Protocols applied to Energy Management in Smart Home Environment

---

Author:
Abdon SERIANNI

Supervisor:
Prof. Floriano DE RANGO

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Information and Communication technologies
DIMES

September 13, 2021

Declaration of Authorship

I, Abdon SERIANNI, declare that this thesis titled, "Technologies and IoT Protocols applied to Energy Management in Smart Home Environment" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

_____

Date:

_____

UNIVERSITY OF CALABRIA

Abstract

Faculty Name

DIMES

Doctor of Philosophy

Technologies and IoT Protocols applied to Energy Management in Smart Home Environment

by Abdon SERIANNI

This thesis presents the studies during this period of my PhD course. In the first research period, I focused the activities principally on the study and analysis of the protocols and technologies used for the IoT solutions in Smart Home environment. It was analyzed the MQTT protocol and its possible applications. The MQTT protocol uses the event-driven publish/subscribe pattern. In our tests, MQTT usage was compared with a classic HTTP request/response paradigm, used in REST and CoAP approaches.

A layered IoT communication architecture will be proposed and described. The usage of proposed IoT communication architecture was analyzed in Smart Home context and in other application contexts such as e-Health and Internet of Vehicles (IoV). After an analysis of Data Mining and Machine learning concepts, the focus of the activities was on Neural Networks. The use of LSTM networks was analyzed for time-series forecasting and prediction of consumption in two different environments (home and office).

In the smart home environment, smart objects are characterized by limited resources. Our proposal to increase the computational capabilities of these smart devices is a hidden cognitive object that uses pre-trained NN and continuous learning for anomaly detection and suggested action prediction tasks. The Cognitive Smart Object is the joining of a smart device and a hidden cognitive object. The Cognitive Smart Object was used in thermal comfort control application and manage better energy consumption. The concepts introduced have been used for an assisted comfort solution and the neural network results were used to suggest to the user conventional management of the climatic comfort levels. A Continuous Learning mechanism was been implemented with the usage of user feedback to shape the neural network and obtain a neural network that follows user behaviours that diverge from behaviour compliant with the ASHRAE standard. From the analysis of the results obtained, it was possible to highlight how NN has given results closer to the user's habits and at the same time the user has been educated to use the right levels of thermal comfort.

Acknowledgements

To my sons

Contents

List of Figures

List of Tables

Acronyms

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks. 6

ACK acknowledgement. 7

AI Artificial Intelligence. 43

ANN Artificial Neural Network. 79–81, 83, 85

ASHRAE American Society of Heating, Refrigerating and Air-Conditioning Engineers. 82

BLE Bluetooth Low Energy. 6, 9, 33, 36, 37

BSS Basic Service Set. 6

CIoT Cognitive Internet of Things. 77, 78

CoAP Constrained Application Protocol. 6, 25–27, 36

CoV Cloud of Vehicle. 43, 48

DCF Distributed Coordination Function. 6

DM Data Mining. 51–53

FCM Google Firebase Cloud Messaging. 38

FFD Full Function Devices. 7, 8

FL Fuzzy Logic. 39, 43

GFSK Gaussian-shaped Frequency Shift Keying. 8

HAR Human Activity Recognition. 37–39

HVAC Heating, Ventilation and Air Conditioning. 1, 36, 80–82, 91, 102

IaaS Infrastructure as a Service. 30

IoE Internet of Everything. 3, 4

IoT Internet of Things. 1–7, 9, 16, 22, 27–29, 32, 33, 35–37, 49, 78, 79

IoV Internet of Vehicles. 29, 34, 43, 44

KDD Knowledge Discovery in Databases. 51, 52

LAS Local Area Service. 43, 44

LSTM Long Short-Term Memory. 51, 60–63, 65, 73–75

LTE Long Term Evolution. 6

M2M Machine-to-Machine. 7, 15, 43

MD Membership Degree. 39, 40

ML Machine Learning. 51, 53

MQTT Message Queuing Telemetry Transport. 6, 9–11, 18–24, 26–28, 36–38, 43–45

NIST National Institute of Standards and Technology. 29

NN Neural Network. 90, 94, 96–98

PaaS Platform as a Service. 30

PAN Personal Area Network. 7

PCF Point Coordination Function. 6

PDF Probability Density Function. 46

PVM Predicted Mean Vote. 78

QoS Quality of Service. 13–15, 19–23

REST REpresentational State Transfer. 6, 21, 22, 26, 27

RFD Reduced Function Devices. 7, 8

RMSE Root Mean Squared Error. 67, 69, 74

SaaS Software as a Service. 30

SET Standard Effective Temperature. 78

TCP Transmission Control Protocol. 15, 27

TLS Transport Layer Security. 20, 21, 25

UDP User Datagram Protocol. 27

UI User Interface. 35

URIs Uniform Resource Identifiers. 16

VM Virtual Machine. 21

WLAN Wireless Local Area Network. 6

WPAN Wireless Personal Area Network. 8

XEP Extension Protocols. 15

XMPP Extensible Messaging and Presence Protocol. 6, 15

Summary

This thesis focuses on the study and analysis of the protocols and technologies used for the IoT solutions in the Smart Home environment.
In the first chapter, an overview of the Internet of Things will be presented with an analysis of a layered IoT architecture. The most used IoT network communication protocols and IoT application protocols will be presented. A performance evaluation of the analyzed IoT communication protocol will be provided.
The second chapter introduces a proposed layered IoT communication architecture and the main entities that constitute it will be reported. The proposed IoT communication architecture has been applied in e-Health and IoV contexts. These proposed solutions are based on the MQTT communication protocol and the use of a Fuzzy Logic classifier.
The third chapter provides an introduction to the notions of Data Mining and Machine Learning applied to the Smart Home with a focus on the use of neural networks for predicting energy consumption and supporting comfort management.
The fourth chapter will be described the use of the Cognitive Smart Object used to assist the management of thermal comfort in the home environment. A Cognitive Smart Object is characterised by the association of a smart device and a hidden cognitive object to improve the HW/SW resources of the smart device. Neural Networks are used for suggested action prediction and anomaly detection operations for HVAC control in the Smart Home context.
The fifth chapter presents the results of an assisted-comfort solution based on the use of Cognitive Smart Objects. The proposed solution is based on the use of feedback and the analysis of user interactions with the system for HVAC control. The research focuses on the research of the trade-off between comfort and energy savings. The concept of continuous learning is applied to allows Neural Networks to learn and adapt to the comfort habits that the user prefers.
The conclusions and future works are reported in the sixth chapter.

Chapter 1

Internet of Things

In this chapter an overview of Internet of Things (IoT) will be presented. We will analyze a layered IoT architecture and the most used protocols and technologies in IoT. A performance evaluation of the analyzed IoT communication protocol will be provided.

1.1 IoT overview

The recent term Internet of Things was first coined by Kevin Ashton in 1999 to name real objects connected to the Internet. The IoT is gaining consensus and represents an opportunity for development especially in modern wireless telecommunications. The increase in connected devices supports the use of innovative solutions, with real objects and places that are now able to communicate with each other, collaborate with other systems, and transfer data and information. There will be an increasing number of connected objects in places and in everyday life, some examples can be:

- Thermostats
- Heating, Ventilation and Air Conditioning (HVAC) control systems
- Cameras
- Smartphones
- Wearable objects
- Environmental sensors

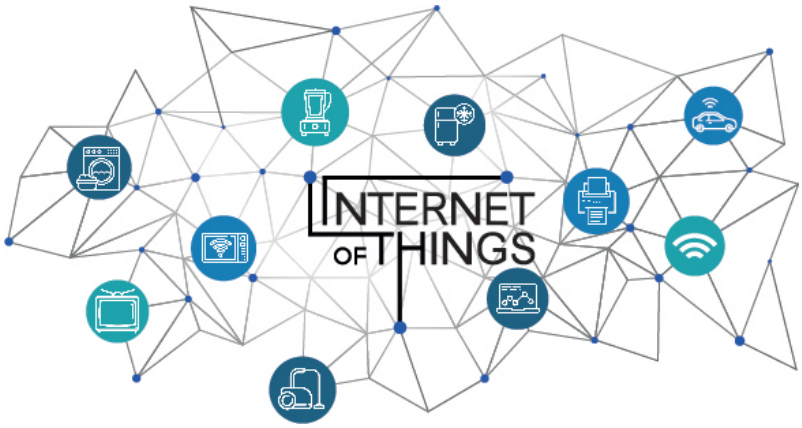

FIGURE 1.1: Internet of Things

The term IoT can indicate a set of technologies that allow us to connect any type of device to the Internet [1] [2].

The IoT can be considered as a network of physical elements composed by:

- *Sensors* used to collect data
- *Identifiers* used to identify the source of data
- *Software* used to analyze data
- *Internet connection* used to communicate and receive notifications.

An alternative re-definition of IoT could be *"IoT is the network of things, with clear element identification, embedded with software intelligence, sensors and ubiquitous connectivity to the Internet"*.

In its simplest form, the main goal of IoT is to physically connect anything/everything through the Internet for monitoring/control functionality.

Examples of IoT devices can be the refrigerator, the clock, the traffic light even if all objects can be considered examples of IoT devices. The important thing is that the objects are connected to the network and that they can to transmit and receive data. [3]

Applications of the Internet of Things can be identified in:

- Home Automation
- Robotics

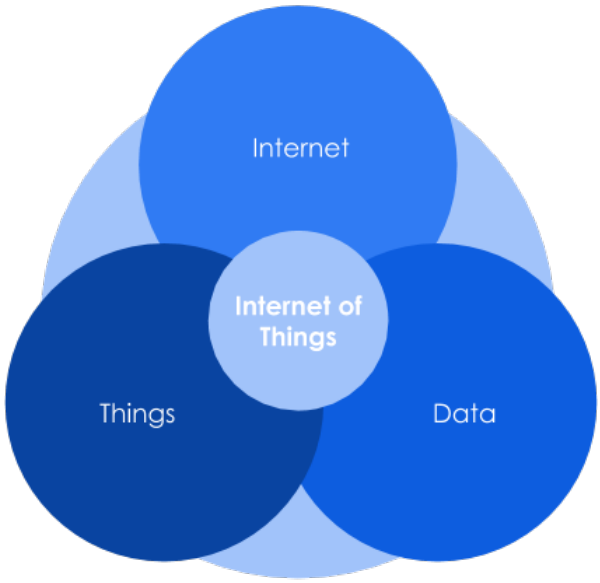

FIGURE 1.2: Internet of Things simplest form

- Automotive industry
- Biomedical Industry

There are also many areas of application for IoT such as:

- Smart City
- Smart Building and Smart Home
- Smart Mobility
- Smart Manufacturing and Industry 4.0
- Smart Agriculture

In these contexts is possible to provide a more complete definition of IoT as the Internet of Everything (IoE) with four main components:

- *People*: connecting people through the Internet
- *Data*: leveraging data for decision making
- *Process*: delivering the right information to the right person/machine at the right time

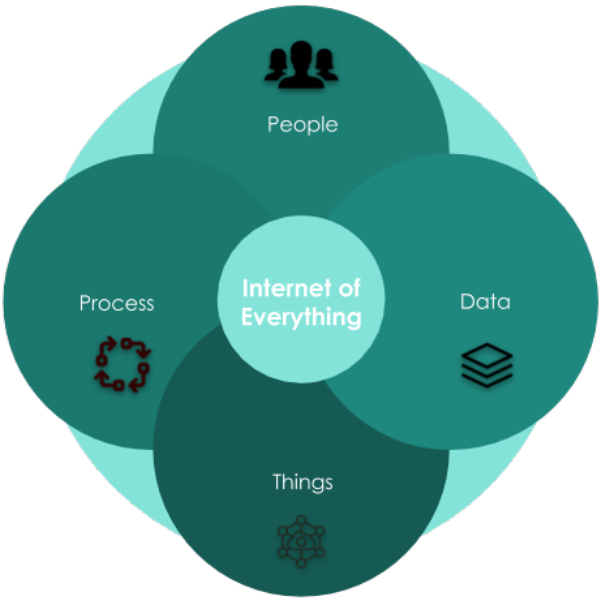

FIGURE 1.3: Internet of Everything

- *Things*: physical devices and object connected to the Internet

The IoE combine people, things, process and data to form a network over the Internet with distributed intelligence, sensing and acting capabilities.

1.2 IoT Architecture

It is possible to split IoT solutions into a layered architecture.There are several numbers of proposed architectures in literature [4] [5]. The basic IoT model consists of a three-layer architecture consisting of the Application, Network and Perception Layers. Some other IoT models are composed by five-layer model: Business, Application, Service Management, Object abstraction and Objects Layers.

A brief description of the five-layers architecture is provided below.

- *Objects Layer*: heterogeneous physical devices (sensors and actuators) of the IoT that collect and process information
- *Object Abstraction Layer*: transfers data produced by Object layer to the Service Management layer using various technologies such as RFID, WiFi, Bluetooth Low Energy, ZigBee, etc.

- *Service Management Layer*: process received data and enable IoT application to work with heterogeneous objects acting as middleware layer.
- *Application Layer*: provides the smart-services requested by the users.
- *Business Layer*: support decision-making processes based on Data analysis, manage system activities and services.

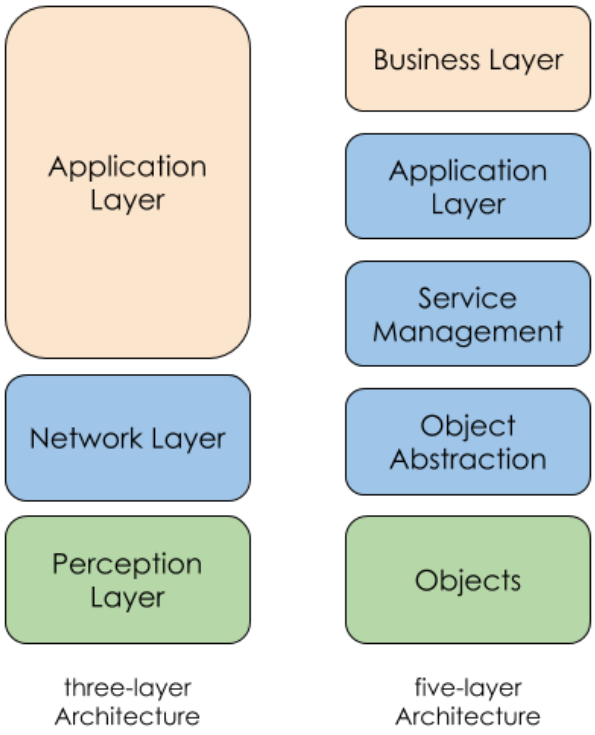

FIGURE 1.4: The IoT Architecture

The Application Layer acts as an interface between end-users and smart devices. Also this layer provide an interface to the Business Layer used to high-level analysis. The management of these functionalities typically require many computational resources and exploit resources made available by Cloud computing and Fog / Edge computing.

1.3 IoT protocols and technologies

IoT communication technologies connect heterogeneous devices to provide specific smart services. There are several network communication standards and protocols

that allow IoT nodes to communicate typically using low power and in the presence of lossy and noisy communication links. Examples of communication protocols used in IoT solutions are IEEE 802.11 i.e. WiFi, Bluetooth Low Energy (BLE), IEEE 802.15.4 i.e. ZigBee, Long Term Evolution (LTE), Z-Wave, IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN) [6]. Other standards and protocols used in the application IoT context are Message Queuing Telemetry Transport (MQTT), Constrained Application Protocol (CoAP), REpresentational State Transfer (REST) and Extensible Messaging and Presence Protocol (XMPP) [7] [8].
A brief introduction of most used protocols and technologies are presented below with a separation of *IoT network communication protocols* and *IoT application protocols*.

1.3.1 IoT network communication protocols

In this section a description of prominent network communication protocols in IoT context are provided.

IEEE 802.11

IEEE 802.11 (*Wi-Fi*) standards group are the most commonly used wireless standards in common networking. The IEEE 802.11 give wireless connectivity that requires quick installation inside a Wireless Local Area Network (WLAN). Wi-Fi standards have been generally adopted for digital devices, including laptops, smartphones, tablets and various smart devices. It defines the MAC methods for accessing the physical medium. Mobility is handled at the MAC layer, so handoff between adjacent cells is transparent to layers built on top of an IEEE 802.11 device. [9]
A Wi-Fi WLAN is based on a cellular architecture. Each cell is called a Basic Service Set (BSS) . A BSS is a set of mobile or fixed Wi-Fi stations. Access to the transmission medium is controlled using a set of rules called a coordination function. Wi-Fi defines a Distributed Coordination Function (DCF) and a Point Coordination Function (PCF).
IEEE 802.11ah is the version of IEEE 802.11 standards which is lightweight to satisfy IoT needs and low power consumption for devices. [10]
IEEE 802.11ah MAC layer features include:

- *Synchronization Frame*: Only valid stations with valid channel information can transmit by reserving the channel medium.
- *Efficient Bidirectional Packet Exchange*: with this feature, the sensors will go to sleep as soon as at the end of the communication and reduces power consumption.
- *Short MAC Frame*: IEEE 802.11ah reduces frame size from 30 bytes in traditional IEEE 802.11 to 12 bytes.
- *Null Data Packet*: Traditional IEEE 802.11 standards had acknowledgement (ACK) frames of 14 bytes. IEEE 802.11ah uses a preamble in place of ACKs and is much less in size.
- *Increased Sleep Time*: this standard is designed for power-constrained devices and it allows a long sleep period and waking up occasionally to exchange data only.

IEEE 802.15.4

The IEEE 802.15.4 protocol is the base of the ZigBee protocol. It is used for low-rate wireless private area networks with reliable communications based on low-power consumption, low data rate, low costs and a large number of nodes. Devices based on this protocol are used in IoT and Machine-to-Machine (M2M) communications with low data-rate services on power-constrained devices. IEEE 802.15.4 utilizes three different frequency channel bands (2.4 GHz, 915 MHz and 868 MHz) with different data-rate, distance coverage, throughput and latency. There are two different types of nodes in an IEEE 802.15.4 network:

- Full Function Devices (FFD) with capabilities of creation, control and management of network and functions of Personal Area Network (PAN) node.
- Reduced Function Devices (RFD) with reduced resources

The network topologies shown in 1.5 and used in IEEE 802.15.4 are:

- Star topology: all communications in the network are managed by the PAN coordinator.

- Mesh topology: any network nodes can communicate with any other devices with ad-hoc network management
- Cluster Tree topology: a special case of a mesh network with a large number of FFD nodes and the RFD nodes that represents a leaf of the tree structure.

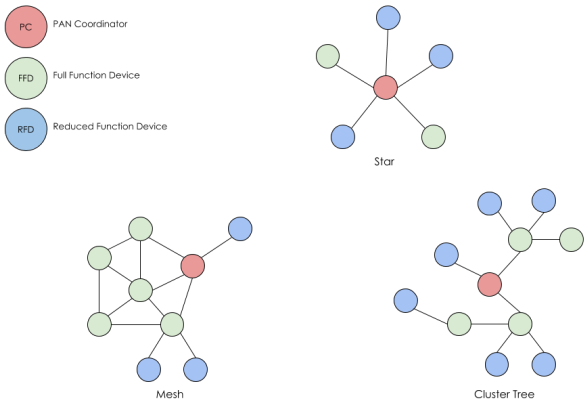

FIGURE 1.5: IEEE 802.15.4 network topologies

Bluetooth

Bluetooth is a standard for wireless communications based on a radio system designed for short-range low-cost communications devices. The devices can be used for single communications between portable devices, act as bridges between other networks, or serve as nodes of ad-hoc networks. This range of applications is known as a Wireless Personal Area Network (WPAN). [9] Bluetooth defines a full communication stack that enables the devices to find each other and advertise their offered services.

A Bluetooth device may operate in both master or slave mode; a maximum of eight devices working together.

Bluetooth devices use the 2.4 GHz band. The channels are accessed using an FHSS technique, using Gaussian-shaped Frequency Shift Keying (GFSK) modulation. Frequency hopping consists of accessing the various radio channels according to a long pseudo-random sequence generated from the address and clock of the master node of the network.

Bluetooth Low Energy

BLE technology is used in IoT context for short-range radio communications with very small power consumption. BLE modules are widely used in smartphones because it allows low-energy communications with other devices such as wearables, sensors and actuators. BLE as a classic Bluetooth uses adaptive frequency hopping spread spectrum to access the shared channel.
A star network topology is used by BLE devices that can operate as masters or slave. A master device can control multiple connections at the same time, but a slave can only be connected to a single master at a time.
In a BLE communication slaves use a discovery mechanism in witch sends advertisement over dedicated advertisement channels. These channels are scanned by the master periodically. To save energy a BLE device remains in sleep mode and exits this mode only to communicate with other devices [11].

Z-Wave

Z-Wave is a low-power wireless communication protocol designed for home automation and used IoT applications like smart homes [12].
Z-Wave communications covers about 30 meters point-to-point and uses small messages. It uses a master/slave architecture with controller and slave nodes. Controllers controls the whole network topology and it manages the slaves by sending commands to them. Z-Wave devices operate around 900 MHz band with a 40 kbps transmission rate. This protocol is specified for applications like smart home control, smart energy management, wearable health care control and fire detection.

1.3.2 IoT application protocols

A description of most used IoT application protocols is provided with detailed operating mechanisms and characteristics.

MQTT

MQTT protocol was developed for lightweight machine-to-machine communications by Andy Stanford-Clark (IBM) and Arlen Nipper (Eurotech) in 1999 and was

standardized in 2013. The intent was to have a bandwidth-efficient protocol characterized by low energy consumption [13], [14]. The protocol uses the publish/subscribe communication pattern and not the classic HTTP request/response paradigm. The publish/subscribe paradigm is event-driven and allows messages to be sent to clients. The data flow utilized by MQTT protocol is depicted below in 1.6.

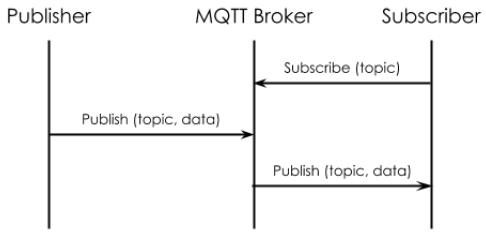

FIGURE 1.6: MQTT publish/subscribe pattern

MQTT consists of three components: *broker*, *publisher* and *subscriber*. The central node of communication is the MQTT broker that dispatch messages between senders and receivers. Each client who publishes a message on the broker includes a topic in the message. The topic is information that the broker uses to forward messages. Each client who wants to receive messages subscribes to a specific topic of interest and the broker forwards the messages relating to the topics of interest to the clients. This architecture allows you to create highly scalable solutions without a direct dependence between data producers and data consumers.

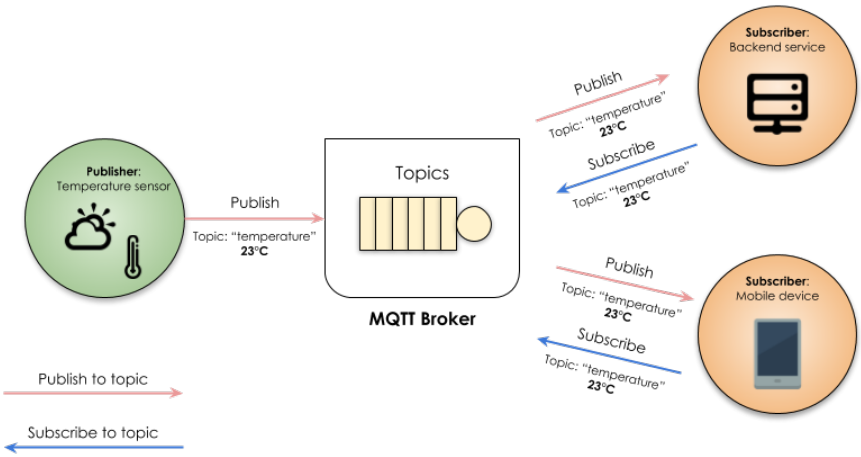

FIGURE 1.7: MQTT architecture

The publish/subscribe approach used with the MQTT protocol separates the clients who send a message (publisher) and the clients who receive messages (subscribers). An MQTT client can typically present publisher and subscriber functionality and use libraries that allow connection to an MQTT broker.

MQTT connections always take place between a client and the broker, there is never a direct connection between two clients. The connection between client and broker starts with a CONNECT message from the client to the broker that replies with a CONNACK message and a status code, once the connection with the broker is established, a client can decide to close the connection using the appropriate commands to disconnect.

The CONNECT message is sent by the client to the broker to establish the start of a new connection. If the message is malformed and does not comply with the specifications of the MQTT standard, the broker closes the connection, this allows you to prevent any attacks from "malicious" clients that can cause overloads and slowdowns and malfunctions of the broker.

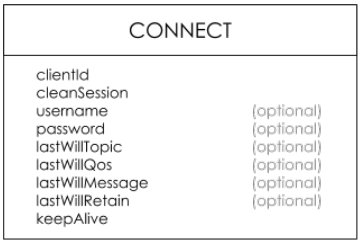

FIGURE 1.8: MQTT CONNECT packet structure

The typical structure of a CONNECT packet is shown in 1.8.
The fields that form up a CONNECT package are:

- *ClientId* represents the unique identifier assigned to each MQTT Client connected to an MQTT broker.
- *Clean Session* is a flag indicating whether the client wants to establish a persistent session or not. Using a persistent session the broker stores all the subscribe operations and the client's lost messages, in a non-persistent session no information about the client is stored by the broker.

- *Username & Password* are the client's authentication and authorization parameters.
- *Will Messages* are parameters that allow the sending of any notification messages to other clients when certain events arise, such as the disconnection of a client.
- *Keep Alive* is the time interval between the PINGs that clients and brokers carry out to check if both are online.

Upon receipt of a CONNECT packet the broker replies with a CONNACK packet which contains two fields: Session Present flag is a flag indicating whether there is already a persistent session on the broker of the client that sent the CONNECT message. Connect acknowledge flag is a signaling flag that specifies whether the connection between client and broker has been successful and possibly what connection problems have arisen. The format of a CONNACK package is shown below.

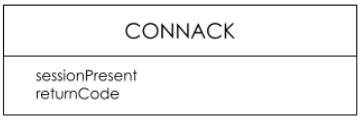

FIGURE 1.9: MQTT CONNACK packet structure

The following table shows the status codes and their respective brief description.

Status code	Description
0	Connection Accepted
1	Connection Refused, unacceptable protocol version
2	Connection Refused, identifier rejected
3	Connection Refused, Server unavailable
4	Connection Refused, bad username or password
5	Connection Refused, not authorized

TABLE 1.1: MQTT CONNACK status code

Once connected to the broker, an MQTT client can publish a message. The MQTT protocol is based on topic-based filtering operations of messages on the broker. Each message must contain a topic that is used by the broker to send messages to the clients "interested" in the topic. In addition to the topic, each message typically has

a payload which contains the data to be transmitted in byte format. The structure of a PUBLISH package is shown below.

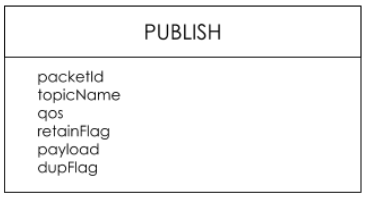

FIGURE 1.10: MQTT PUBLISH packet structure

The fields that make up a PUBLISH package are:

- Topic Name is a string, typically hierarchical
- Quality of Service (QoS) represents the Quality of Service level of the message, it can take values 0, 1 and 2.
- Retain-Flag is a flag that determines whether the message can be saved by the topic for the specific topic in order to provide the latest messages of the topic of interest to new clients who subscribe to the topic.
- Payload is the content of the MQTT message
- Packet Identifier is the unique identifier used by clients and brokers to identify the message.
- DUP-flag represents the duplicate flag and indicates whether the message has been re-sent.

The client who wants to publish information on a specific topic, only deals with sending the PUBLISH package to the broker, it is then the responsibility of the broker to forward the information to the clients who have subscribed to the correct topic. After a publish operation, the client does not receive any feedback and has no information on how many or which clients received the information. The management of the exchange of information of the MQTT protocol provides, in addition to sending PUBLISH messages, the consideration during the reception of messages, SUBSCRIBE messages or packets that have the following structure.

The fields of a SUBSCRIBE package are:

FIGURE 1.11: MQTT SUBSCRIBE packet structure

- Packet Identifier is the unique identifier used by clients and brokers to identify the package.
- List of Subscriptions may contain an arbitrary number of topic / QoS Level pairs representing the client's topics of interest and the respective required QoS levels.

The message that can be defined as the opposite of that of SUBSCRIBE is the message of UNSUBSCRIBE which is used to remove a client's subscribe to a topic of interest. The fields of the UNSUBSCRIBE message are completely equivalent to the SUBSCRIBE message and the package format is shown below.

FIGURE 1.12: MQTT UNSUBSCRIBE packet structure

The concept that links MQTT's publish and subscribe operations is that of topic, a topic is a UTF-8 string that is used by the broker to filter the messages of the various connected clients. A topic consists of one or more topic levels, each topic level is separated by a topic level separator. A typical example of a topic is shown below.

A client does not need to create a topic before performing publish and/or subscribe operations, in fact the broker accepts every valid topic, without the need for initialization. A valid topic contains at least one character, can contain spaces and is case-sensitive, for example the topics *smathome/smartDevice* and *smartHome/smartDevice* will be two separate topics.

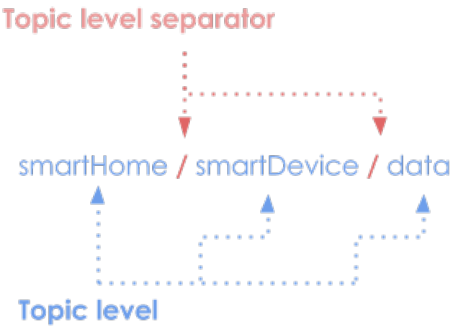

FIGURE 1.13: MQTT Topic structure

XMPP

The XMPP is a protocol designed for chats and messages exchange applications. It is based on XML language and was standardized by IETF more than a decade ago. XMPP is extensible and allows the specification of Extension Protocols (XEP) that increase its functionality. [15] XMPP operates over Transmission Control Protocol (TCP) and provides publish/subscribe and also request/response messaging communication patterns. It is designed for near real-time communications and thus, efficiently supports low-latency small messages.
XMPP has TLS/SSL security built in the core of the specification. However, it does not provide any QoS options that make it impractical for M2M communications. XMPP uses XML messages (eXtensible Markup Language) that create additional overhead due to unnecessary tags and require XML parsing that needs additional computational ability which increases power consumption.

RESTful API over HTTP

The use of web-based services based on REST architecture has now become commonplace. The REST architecture, denied in 2000 in Roy Fielding's doctoral thesis [16], collects a set of principles on how network architecture should be composed, the main characteristics of which are:

- *Client-Server*: network architecture to be used in which an entity (client) uses a series of services made available to an entity (server).

- *Stateless*: requirement of the REST architecture is that the communication is of the stateless type (CSS - Client Stateless Server) or every request made by the client to the server must contain all the information necessary for the server to understand the request, without referring to any further stored server-side information.

- *Cache*: clients have cached a response obtained following a request must be marked as cacheable or non-cacheable. If a response is cacheable, a client can take advantage of that cached response for subsequent equivalent requests. This allows you to improve scalability, decrease network traffic and have less latency.

- *Resources*: resources that servers make available to clients are indicated by a resource identifier; through it, you can access the resource and/or change its status.

CoAP

CoAP is a network-oriented protocol for IoT applications that uses features similar to the HTTP protocol with low overhead.[17], [18] Unlike HTTP-based protocols, CoAP is based on the UDP protocol and does not use the classic congestion control techniques provided by the TCP protocol. CoAP provides Uniform Resource Identifiers (URIs) and REST methods such as GET, POST, PUT and DELETE, allows group communication for the IoT using multicast. Improvements on the reliability of the UDP protocol are given by mechanisms for retransmission and discovery of the resources made available by the CoAP protocol.

The interaction model of the CoAP protocol is comparable to the client/server model on which the HTTP protocol is based. As illustrated in picture 1.14 CoAP can be logically divided into two distinct sub-layers:

- *Messages*: designed to manage the lower UDP layer and to provide reliable communication with message duplication detection and recovery mechanism

- *Request/Response*: manages REST communications and the exchange of information between requests and responses

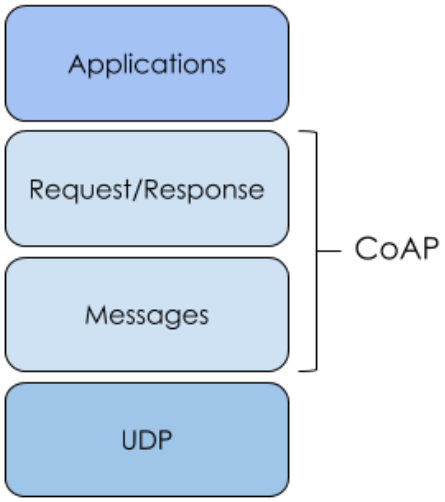

FIGURE 1.14: CoAP Stack

The exchange of messages in the CoAP protocol is asynchronous and the Messages level supports 4 different types of messages:

- *Confirmable*: messages that requires an acknowledgement (ACK)
- *Non-confirmable*: messages that do not require the sending of a response (ACK)
- *Acknowledgement*: response messages to confirmable messages
- *Reset*: message sent after a request that the server could not be process

Reliability of CoAP communication is achieved by a mix of confirmable and non-confirmable messages.

CoAP is based on the exchange of compact messages, the structure of a CoAP package is shown in 1.15 and includes a 4-byte header followed by a variable-length part which can include the Token, Options and Payload fields. A Token field is used to correlating requests and responses

The header of the CoAP package consists of the following fields:

- Version (Ver): version of CoAP used
- Type (T): type of message

    0. confirmable (request)
    1. non-confirmable (request)

2. acknowledgement (response)
3. reset (response)

- Token Length (OC): 4 bits that indicate the length in bytes of the token field that can take values from 0 to 8 bytes.
- Code: 8 bits in which the first 3 bits indicate the class (c) and the following 5 bits indicate the detail (d) forming a code field of type *class.detail* like HTTP status codes.
- Message-ID: 16 bits that identify the messages to discriminate any duplicates and associate ACK / Reset response messages with Confirmable / Non-confirmable request messages

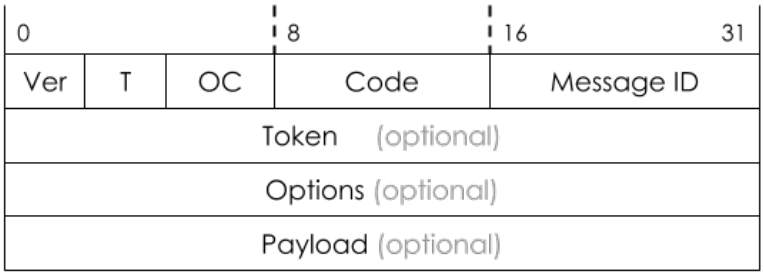

FIGURE 1.15: CoAP packet structure

1.3.3 Evaluation of IoT communication protocols

In this subsection, a performance evaluation of different IoT communication protocols is provided [19].
In [20], studies on the performance comparison of MQTT and CoAP are introduced using a common middleware that supports MQTT and CoAP. Experiments are used to analyse the performance of MQTT and CoAP in terms of end-to-end delay and bandwidth consumption.
In [21], the authors present and analyze the efficiency, usage, and requirements of MQTT and CoAP using a Raspberry-Pi and a temperature sensor.

The analysis of the communication times for MQTT protocol is linked to the event-driven publish/subscribe pattern. In our tests, the communication time is given by *Publish to Broker time + Publish to Subscriber time* in milliseconds.

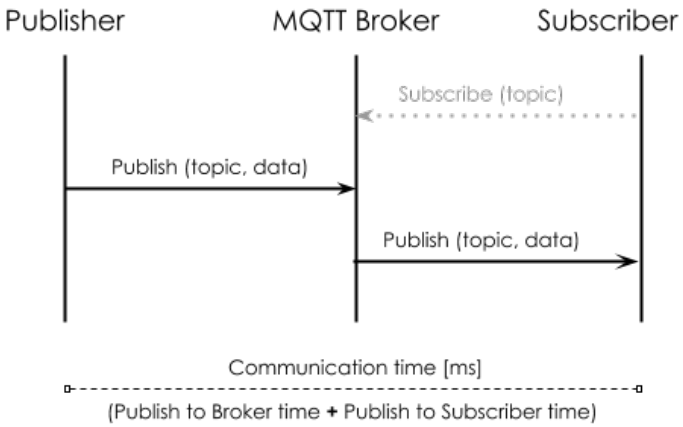

FIGURE 1.16: MQTT Publish/Subscribe timing

In classic HTTP request/response paradigm, used in REST/CoAP approach, the communication time is given by *Request time* + *Response time* in milliseconds. For all communication tests we have used the same JSON message that contains:

- *MessageID*
- *Timestamp*
- *Payload* of variable length (10, 50 or 100 bytes)

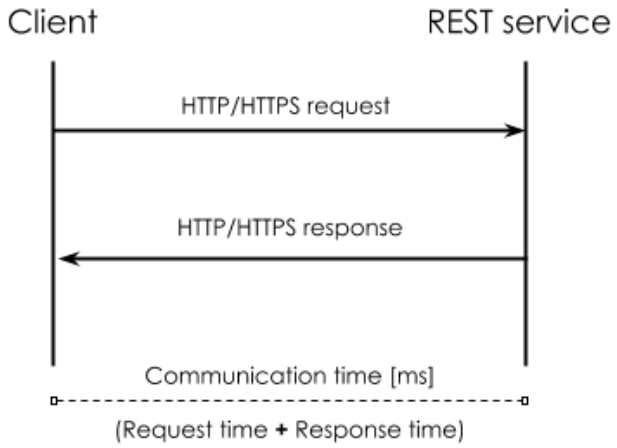

FIGURE 1.17: REST request/response timing

The communication time analysis of MQTT protocol was carried out for each MQTT QoS level. The QoS level defines the guarantee of delivery for MQTT messages:

- At most once (0) - best effort delivery without guarantee of delivery
- At least once (1) - a message is delivered at least one time to the receivers
- Exactly once (2) - a message is delivered exactly once to the receivers

Different QoS levels produce different communication times as shown in fig. 1.18.

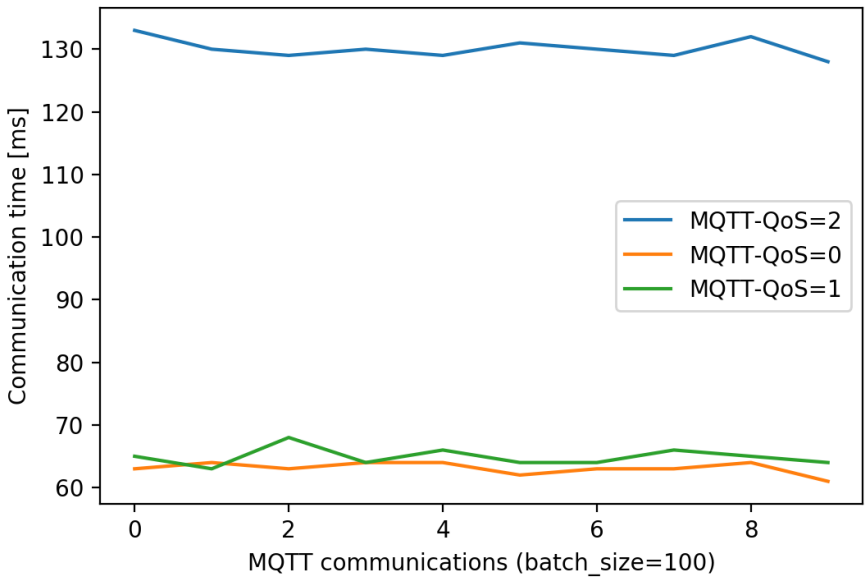

FIGURE 1.18: MQTT communication time for different QoS

1000 MQTT communications were used for this test for each level of QoS. The data were grouped into 10-batches and each batch represents the average communication time of 100 MQTT delivered messages. For QoS level 0 and 1, there are similar communication times about 63 milliseconds, for the QoS 2 level, the average communications time is about 130 milliseconds.

The previous results differing in the value of QoS were evaluated using Transport Layer Security (TLS). TLS is a cryptographic protocol which enables a secure and encrypted communication at the transport layer between client and server.
For QoS level 0 and 1, there are similar communication times about 67 milliseconds, for the QoS 2 level, the average communications time is about 175 milliseconds.
The results of the communication time of MQTT with TLS security are higher than

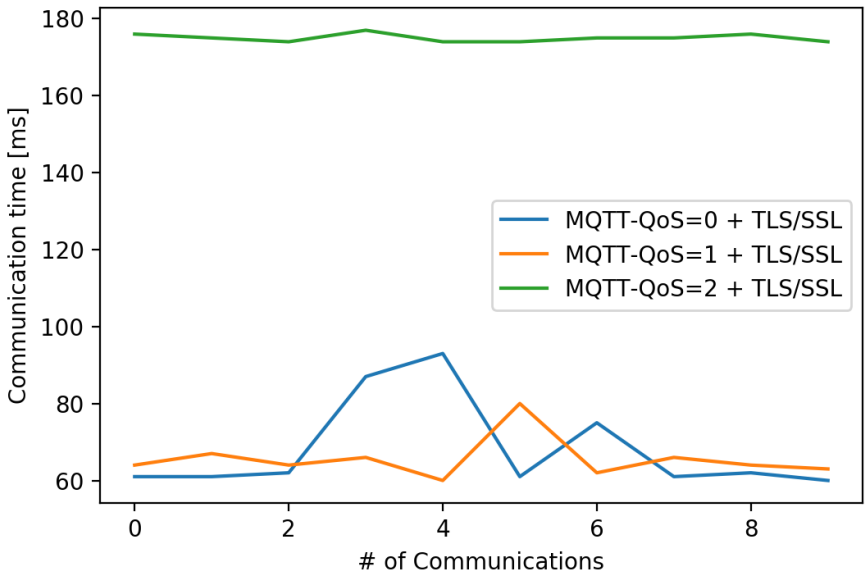

FIGURE 1.19: MQTT communication time for different QoS with TLS Security

the results shown in fig. 1.18. This is due to the construction of an encrypted channel with the exchange of keys and security certificates between client and server that allows increasing the security levels of the MQTT protocol.
The result is influenced by the caching of server responses that use the same established TLS connection.

The results of MQTT communication times are compared with REST request in fig. 1.20 and fig 1.21. In particular, tests have been done with 100 REST HTTP requests and 100 REST HTTPS requests grouped into 10-batches of 10 rest requests each one.

As depicted in the figures 1.20 and 1.21 the communication times of REST requests are higher than the MQTT communication times. In particular, the average communication time of REST HTTP requests are about 156 milliseconds and the average of REST HTTPS requests is about 295 milliseconds.
These results have shown that MQTT usage is better than rest usage in term of communication time with or without TLS security for all QoS level.

The test environment was the same for all tests, MQTT Broker and REST services were hosted into the same Cloud Virtual Machine (VM) and clients (MQTT

publisher, MQTT subscriber and REST client) runs over the same test workstation.

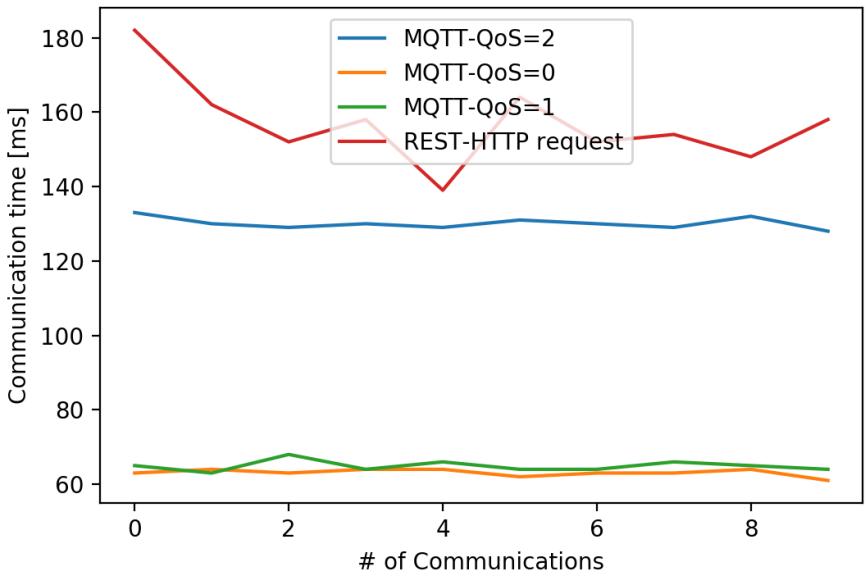

FIGURE 1.20: MQTT vs REST communication time

The analysis of the messages exchanged in the communications for different used IoT protocols is presented in terms of data exchanged and protocol overhead.

As depicted in the next figures, the message exchange analysis for MQTT protocol was given out for each QoS level.

In MQTT communications with QoS=0, a best-effort delivery is guaranteed, the communication can be divided into:

- *Publisher* sends the message to *MQTT Broker* with a PUBLISH
- *MQTT Broker* sends the message to *Subscriber* with a PUBLISH

For a message of 166 bytes of payload, the total exchanged data was 492 bytes but there is no guaranteed of delivery.

For MQTT communications with QoS = 1, a message is delivered at least one time to the receiver. As shown in fig. 1.23 for every published message an ack is generated, if a sender not receives this acknowledgement re-send the message, this can generate multiple delivered messages.

The communication with QoS=1 can be divided into:

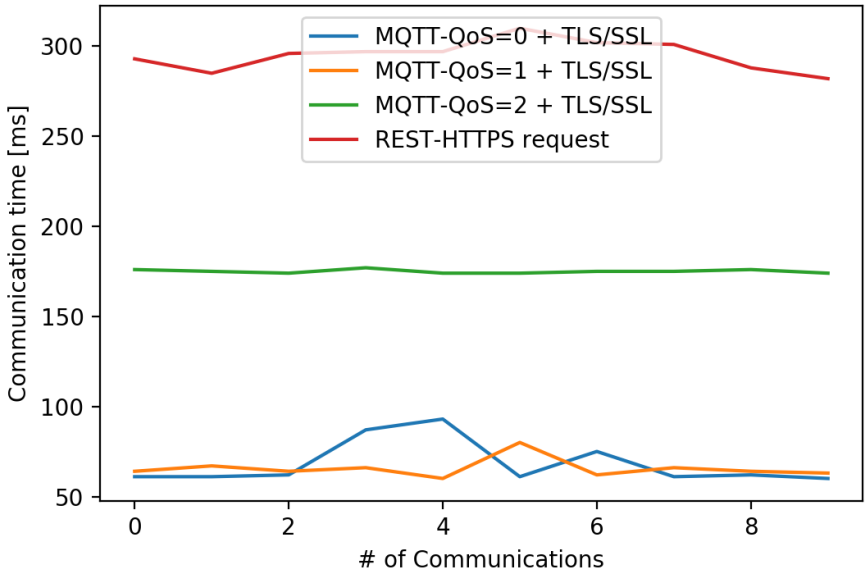

FIGURE 1.21: MQTT vs REST communication time with security

- *Publisher* sends the message to *MQTT Broker* with a PUBLISH
- *MQTT Broker* sends the ACK message to the *Publisher*
- *MQTT Broker* sends the message to *Subscriber* with a PUBLISH
- *Subscriber* sends the ACK message to the *MQTT Broker*

The 166 bytes of payload lead 636 bytes of total exchanged data (without multiple delivered messages).

MQTT communications with QoS=2 is the safest and slowest QoS level with a handshake of four messages between the sender and the receiver that confirm that the message has been sent and that the acknowledgement has been received. Each message is received only once by the receivers.

The communication with QoS=2 can be divided into:

- *Publisher* sends the message to *MQTT Broker* with a PUBLISH
- *MQTT Broker* sends the RECEIVED message to the *Publisher*
- *Publisher* sends the RELEASE message to *MQTT Broker*

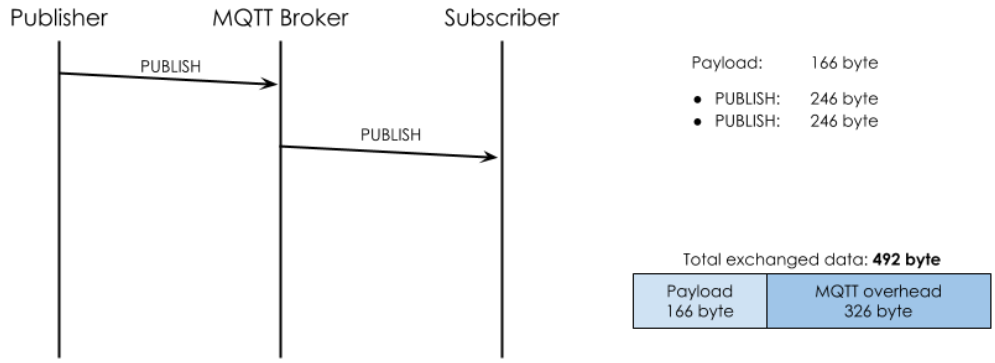

FIGURE 1.22: MQTT with QoS=0 data exchange

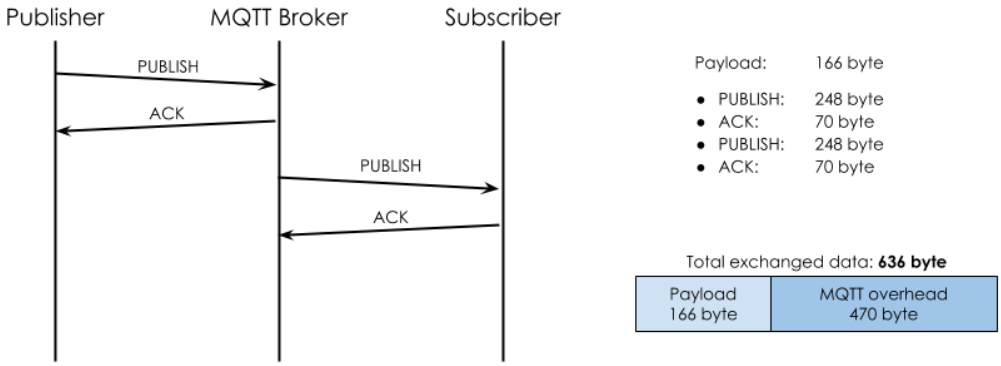

FIGURE 1.23: MQTT with QoS=1 data exchange

- *MQTT Broker* sends the COMPLETE message to the *Publisher*
- *MQTT Broker* sends the message to *Subscriber* with a PUBLISH
- *Subscriber* sends the ACK message to the *MQTT Broker*

For a message with 166 bytes of payload, the total exchanged data was 776 bytes.

The data-exchange analysis for classic HTTP/HTTPS request is presented below. The request/response paradigm of an HTTP request with a payload of 166 bytes can be divided into:

- HTTP request size equal to 636 bytes
- HTTP response size equal to 535 bytes

For a request/response with an exchange of 166 bytes of payload data, there are 1171 bytes of total exchanged data with a protocol overhead of 1005 bytes.

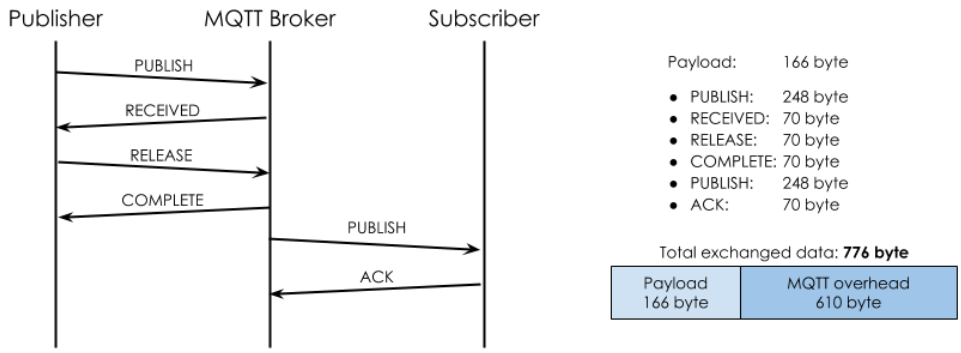

FIGURE 1.24: MQTT with QoS=2 data exchange

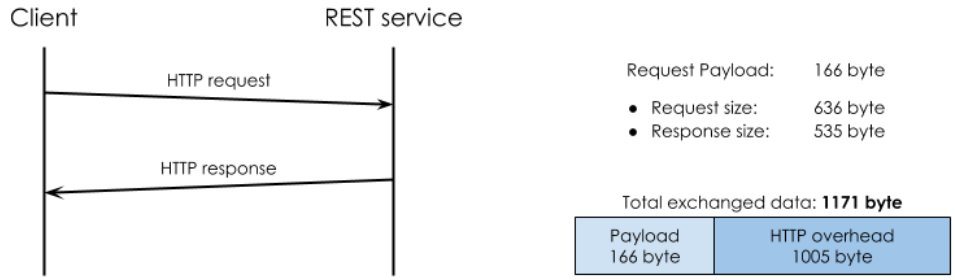

FIGURE 1.25: HTTP request data exchange

Considering an HTTPS request, there are 1011 bytes of protocol overhead to which the data exchanged for the TLS handshake must be added.
During a TLS handshake, both communicating sides exchange messages to establish encrypted communication with asymmetric encryption and use of a pair of a public and private key. The analyzed TLS overhead is equal to 4749 bytes, in fig.1.26 is depicted the complete HTTPS request data exchange.

The data-exchange analysis for CoAP request/response is presented following for *Confirmable* (CON) and *Non-Confirmable* (NON) CoAP message.

In Confirmable messages an acknowledgement (ACK) is required, for a request of 166 bytes there are 306 bytes of total exchanged data with a CoAP protocol overhead of 140 bytes. The CoAP request size is equal to 237 bytes and the CoAP ACK size is equal to 69 bytes and contains the response or the error code.

No acknowledgement is required in Non-Confirmable messages, for a request of 166 bytes there are 237 bytes of total exchanged data.

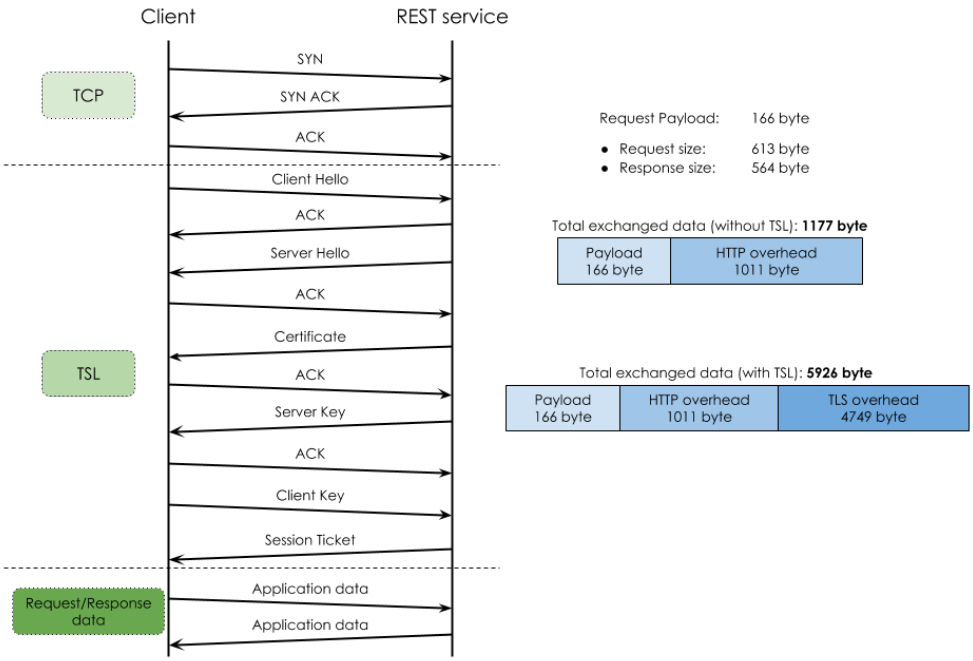

FIGURE 1.26: HTTPS request data exchange

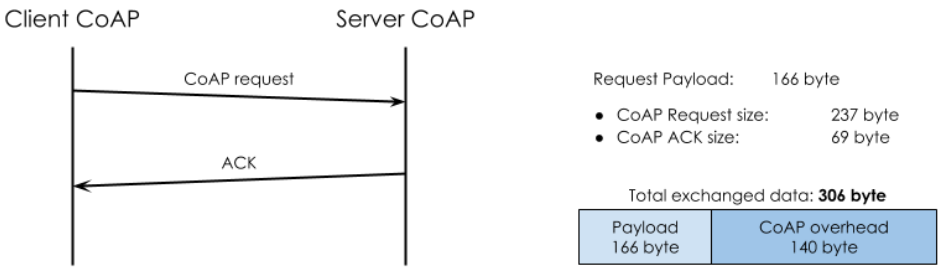

FIGURE 1.27: CoAP CON request data exchange

The main differences between MQTT, CoAP and REST are:

- MQTT uses a *publish/subscriber* paradigm while CoAP and REST uses a *request/response* paradigm
- MQTT is a many-to-many protocol that uses a central broker to dispatch messages coming from the publisher to the subscribers instead in CoAP and REST there is a one-to-one client-server communication
- MQTT is an event-oriented protocol while CoAP and REST are more suitable for state transfer
- CoAP and REST require updates on status changes with periodic requests

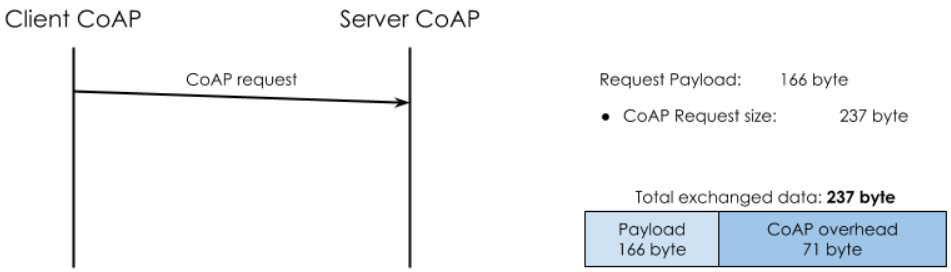

FIGURE 1.28: CoAP NON request data exchange

- CoAP runs on top of the User Datagram Protocol (UDP) while MQTT and REST run on top of TCP

The IoT communication protocols analyzed use different approaches and are characterized by different performances and ways of use.

MQTT is the protocol with the best performances in terms of communication times, this has been clearly observed in the previous evaluations.
In terms of ease of use and client availability, the REST approach is the most used, MQTT is the easiest protocol to use with the availability of clients for various environments (Java, Android, Python, C, C ++, JavaScript).
In terms of protocol overhead, CoAP is the analyzed protocol with the best performance. CoAP needs the use of an HTTP-CoAP proxy for remote integration and this was a drawback for the protocol.

1.4 Chapter summary

In this chapter, an overview of the Internet of Things has been presented with the definition of the application areas and a layered IoT architecture. The most utilised protocols and technologies in the IoT domain were analyzed with the subdivision into IoT network communication protocols and IoT application protocols. A performance evaluation of different application protocols was provided in terms of the communication time, data exchanged and protocol overhead. The analyzed protocols use different methods and are characterized by different performances and techniques of use. MQTT is the protocol with the best performances in terms of communication time and is the easiest protocol to use with the availability of clients for several environments.

Chapter 2

IoT Communication architecture

This chapter will briefly introduce the concepts of Cloud, Edge and Fog computing. A layered IoT communication architecture will be proposed and the main entities that constitute it will be described. The application of IoT communication architecture will also be shown in other application contexts such as e-Health and Internet of Vehicles (IoV).

2.1 Cloud, Edge and Fog computing

The National Institute of Standards and Technology (NIST) in 2011 defines Cloud computing as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". [22]

Cloud computing allows complete virtualization and scalability of IT infrastructures and services. The term Cloud Computing, in particular, indicates a series of technologies that allow data to be processed and stored thanks to the use of hardware and software resources distributed over the network.

The basic features of Cloud computing are:

- *On-demand usage*: computing capabilities can be provided automatically when needed, without requiring any human interaction between consumer and service provider;

- *Ubiquitous network access*: computing capabilities are available over the network and accessible using a wide range of client platforms (e.g., workstations and mobile devices);
- *Dynamic Resource management*: computing resources are location independent and pooled to serve multiple consumers, dynamically allocating and deallocating them according to consumer demand;
- *Scalability*: computing capabilities can flexibly be provided and released to scale in and out according to demand. The consumer has the perception of unlimited, and always adequate, computing capabilities;
- *Usage monitor*: resource usage can be monitored and reported according to the type of service offered. This is particularly relevant in charge-per-use, or pay-per-user, services because it grants great transparency between the provider and the consumer of the service.

There are different types of Cloud solutions and they can be:

- *Public Cloud*, provided by a provider that makes resources and virtual machines, storage and applications available through the network;
- *Private Cloud*, a Cloud system consisting of servers owned by those who use the service;
- *Hybrid Cloud*, an intermediate solution between the private Cloud and Public Cloud.

As shown in 2.1, Cloud computing services can be differentiated into three different types:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

The strong interaction between Cloud computing and IoT introduces several new challenges that cannot be fully solved only with Cloud computing solutions.

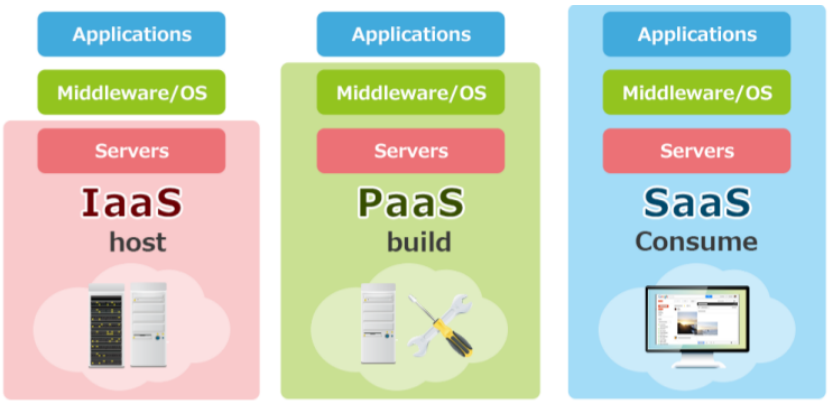

FIGURE 2.1: Types of Cloud services

Also, there has been an increasing number and variety of smart clients and powerful edge devices, such as smartphones, tablets, smart IoT devices, smart vehicles, etc.. In this context, Edge and Fog computing principles have become extremely interesting.

Edge computing is an emerging paradigm born of necessity to move the computation at the edge of the network. The increasing interest for Edge computing starts with the emerging of IoT. As defined in [23], "Edge computing refers to the enabling technologies allowing computation to be achieved at the edge of the network, on downstream data on behalf of Cloud services and upstream data on behalf of IoT services". The basic idea of Edge Computing is to extend Cloud computing to the network edge moving the computation near the data sources represented by the IoT devices.

Fog computing provides distributed computing, storage, control, and networking capabilities closer to the user and this is often considered as an implementation of Edge Computing. CISCO in 2012 defined Fog computing as "a highly virtualized platform that provides compute, storage, and networking services between end devices and traditional Cloud computing Data Centers, typically, but not exclusively located at the edge of network". [24]

Fog computing includes the Edge computing concept, providing a structured intermediate layer that connects IoT and Cloud computing. [25]

Fog nodes can be placed anywhere between IoT devices and the Cloud and not always directly connected to the data sources. Fog computing can be considered a

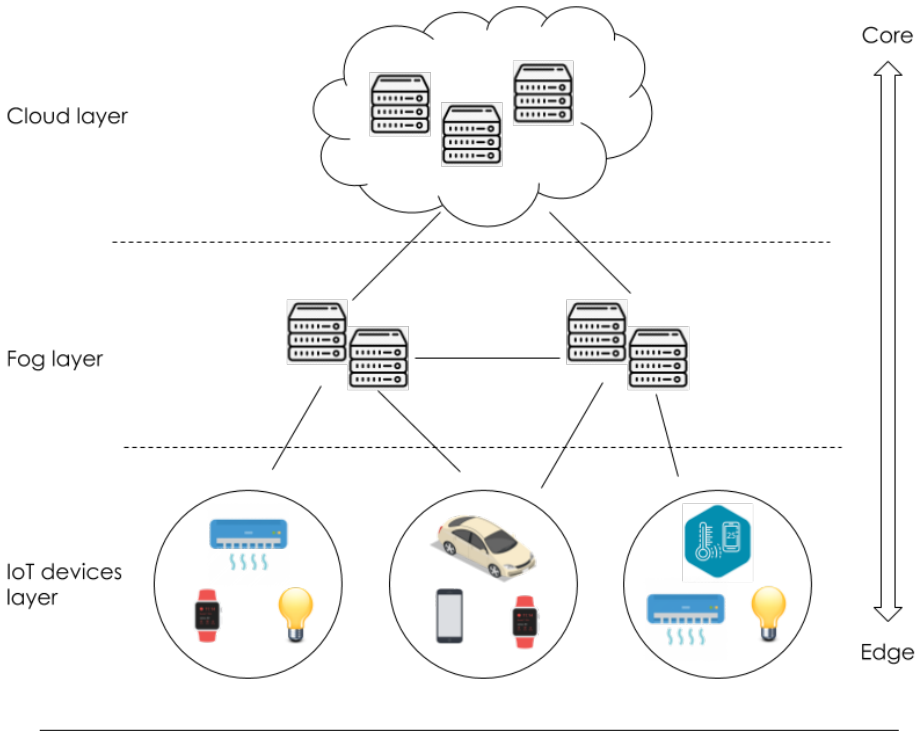

FIGURE 2.2: Layered architecture of Fog computing

new entity between Cloud and IoT that support and improve their interaction, integrating IoT, Edge and Cloud computing.

2.2 Definition of IoT communication architecture

The focus of research is data analysis and energy efficiency of devices in the Smart Home environment using IoT communication protocol and technologies.[26]
In this work an IoT communication architecture has been defined to improve energy management and devices power consumption in Home Automation.
A general IoT communication architecture is depicted in fig. 2.3.
The proposed architecture can be utilized in different application contexts of IoT. The core-entity is a Cloud-based Platform used to:

- IoT device management
- Data gathering and analysis
- User interaction management

Starting from the study on IoT network communication protocols analyzed in the previous chapter and exploiting the advantages of the technologies analyzed, it is possible to use an IoT Gateway device equipped with multiple communication interfaces (Wi-Fi, ZigBee e BLE).

Smart IoT devices (e.g. smart thermostat, smart meter, smart lighting system, smart plugs, wearable smart devices, smart vehicles) can communicate directly with the Cloud-based platform or use the IoT smart gateway. Users of the smart IoT environment who use the mobile APP or the Web APP can use smart devices to obtain, for example, energy waste reduction and life quality improvement.

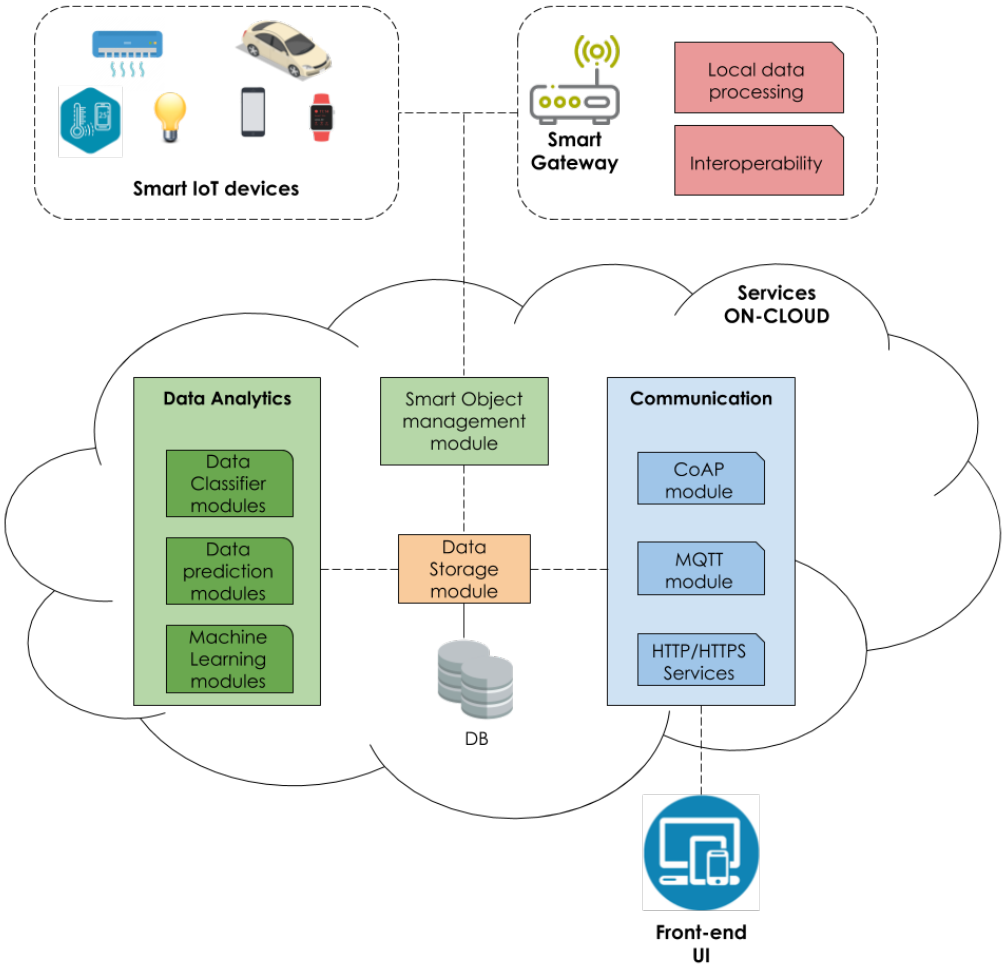

FIGURE 2.3: General IoT communication architecture

2.2.1 The role of Cloud-based platform

The Cloud-based platform implements data storage and data analytics. It provides a smart device and user interaction management. The main entities of the core-layer of the proposed architecture are:

- Data Analytics modules
- Communication modules
- Data Storage module
- Smart Object management module

The general IoT architecture proposed can be customized and applied in several application areas of the IoT. The SW modules of the on-cloud services are specialized according to the application context and the problem encountered. This chapter presents the proposed IoT architecture usage in the Smart Home context and solutions related to e-Health and IoV.
The data analytics techniques used differ according to the problem encountered and generally may require the use of the following techniques:

- *Data classification*
- *Data forecasting*
- *Machine learning*

The SW communication modules are based on different IoT communication technologies to allow the connection of heterogeneous smart devices. As shown in the previous chapter, several network communication standards and protocols provide IoT nodes to communicate. Standards and protocols related to the proposed IoT architecture are MQTT, CoAP and REST services on HTTP/HTTPS.
Smart Device Management and Data Storage modules define the operations and the data used and stored associated with smart devices. Examples of operations and data that are shaped and are re-defined according to the type of IoT context addressed are, for example, configuration operations, the status of the device's data and the interactions of the different devices with the proposed system and log information.

The Client User Interface (UI) can be seen as the set of mechanisms provided to the user for interaction with the platform created and includes a web-app and mobile APP that the user uses for the use and management of smart devices.

2.2.2 The role of proposed IoT Gateway

The main role of the smart IoT gateway is to support various protocols and take local pre-processing data. The tasks carried out by the gateway are here briefly described.

Local data processing

Local data processing is implemented to provide intelligence at the IoT gateway by which the device's data is analyzed locally. The fog layer handles a huge amount of data and response appropriately concerning various conditions of the smart home environment. Figure 2.4 describes the block diagram of the local pre-processing functions of the IoT gateway device.

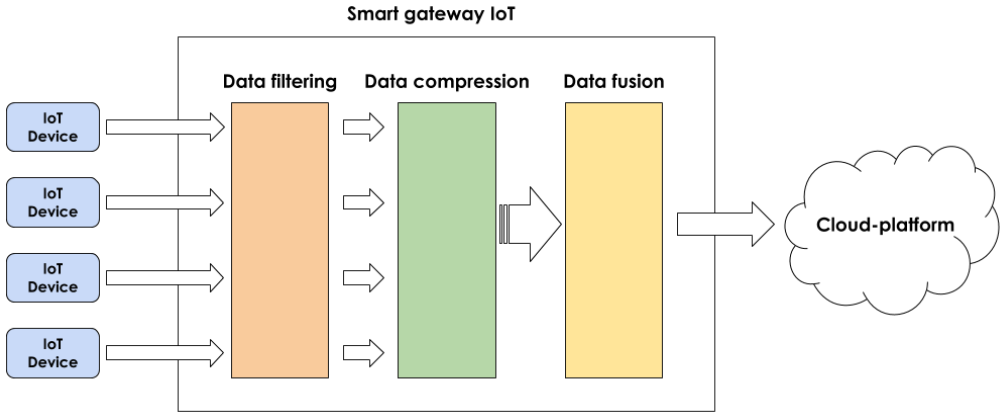

FIGURE 2.4: IoT Gateway local data processing schema

The *Data Filtering* in pre-processing task at the edge is required before data analysis is performed on Cloud-side of system. The data collected from smart IoT devices in the home environment contains a snapshot of the current status of the user devices and the environmental parameters of the house. The fog layer receives data via various communication protocols and implements light-weight filtering to remove some corrupt or incomplete data.

Data Compression techniques are applied for reducing communication latency and energy consumed. For devices that acquire huge amounts of data such as smart

meter devices, it is important to use lossless compression techniques, to be able to reconstruct the entire sequence of data on the cloud platform.

Data Fusion allows to decrease the volume of devices data, and consequently reduce the energy needed for data transmission. Data fusion is classified into three categories: complementary, competitive, and cooperative [27]. Complementary data fusion can be performed at the fog layer to perform more reliable global knowledge. The average temperature of the home environment is an instance provided from different devices such as HVAC controller and intelligent thermostat. Competitive data fusion can be utilized when a single environment parameter is gathered from different devices to increase the accuracy and consistency of results. Cooperative data fusion can also contribute advantages at the Fog layer with new information can be extracted from the heterogeneous data collected from diverse sources. For example, cooperative data fusion can produce general information about the home environment.

Interoperability

The smart IoT gateway provides technical interoperability for the heterogeneous IoT devices connected via different network interfaces. As depicted in figure 2.5, the IoT devices are connected to the Smart gateway using various standards (e.g., ZigBee, BLE, Wi-Fi). Semantic interoperability is granted integrating different IoT communication protocols like MQTT and CoAP in smart IoT gateway. [28]

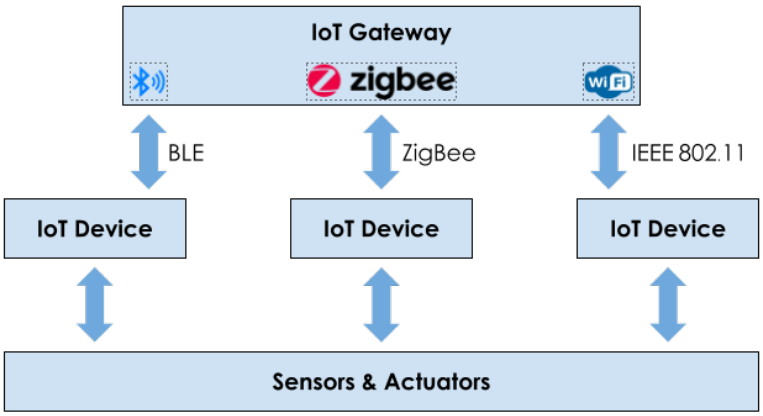

FIGURE 2.5: IoT Gateway interoperability

These IoT communication protocols are different characteristics and messaging architecture helpful for different types of IoT devices and applications.

2.3 Application of proposed architecture in other application fields

The proposed IoT communication architecture was used in the context of the Smart Home and in two other different application contexts which are e-Health and IoV. These alternative two solutions will be described in the following sections.

2.3.1 e-Health application

In [29] our goal is to help people to monitor their activities and increase knowledge about their habits and health status. Health status recognition is achieved by a co-operative platform that uses data coming from IoT devices to keep reference values always updated.

IoT Architecture overview

The communication is based on MQTT protocol, the data are transmitted and received by using the publish/subscribe paradigm proposed by the MQTT standard. The presented software architecture can be divided into two main modules:

- Smartphone application
- Cloud assisted application

Regarding the wearable device, it is based on an Arduino board and it can periodically collect data from the sensors. Data are collected and sent to the smart-phone through BLE interface. The smartphone application collects and sends data to the external services exploiting the lightweight protocol. On Cloud assisted platform human activity classification process is triggered by the Human Activity Recognition (HAR) routine. Moreover, data mining routines analyze received data to discover anomalies.
Software layer represents an important component of the whole IoT ecosystem. In

the server-side block, a server based on MQTT specifications has been deployed by using Eclipse Paho Java Client [30], an MQTT client library written in Java. Smartphones are used to send acquired data from the wearable smart device through 3G/4G connection towards the MQTT broker.

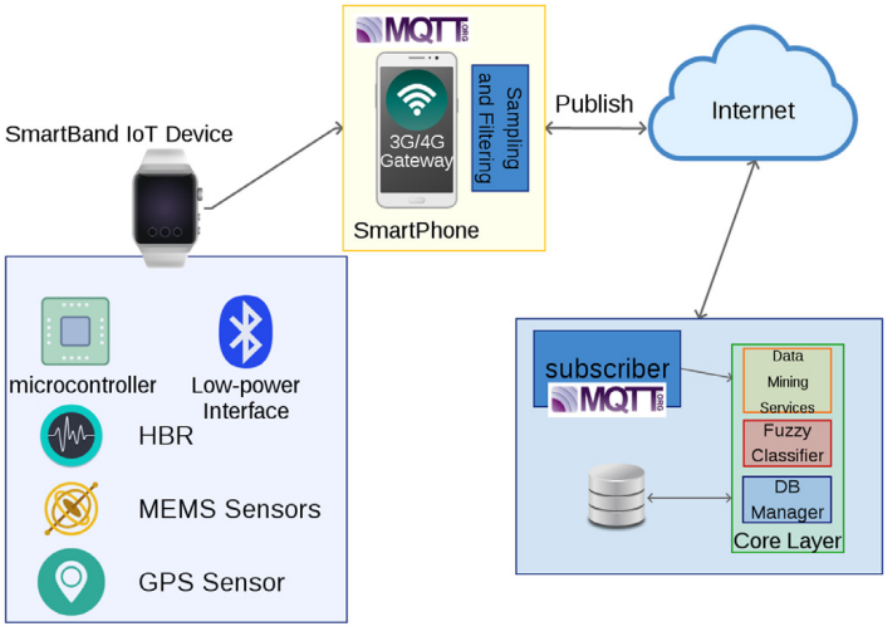

FIGURE 2.6: e-Health proposed architecture

The proposed architecture uses HiveMQ [31], an highly scalable Enterprise MQTT Broker designed for lowest latency and highest throughput. The core layer is composed of three sub-modules:

- DB Manager used to gather data from MQTT broker and store it in a DB.
- HAR Classifier used to classify Human Activity from user data using Fuzzy Logic Rules.
- Data Mining Services are used to make a post-analysis to discover pattern and anomalies.

To store collected data we choose MongoDB as a database management system. Google Firebase Cloud Messaging (FCM) services are used to send notification messages and post-analysis result to the client application via HTTP protocol.

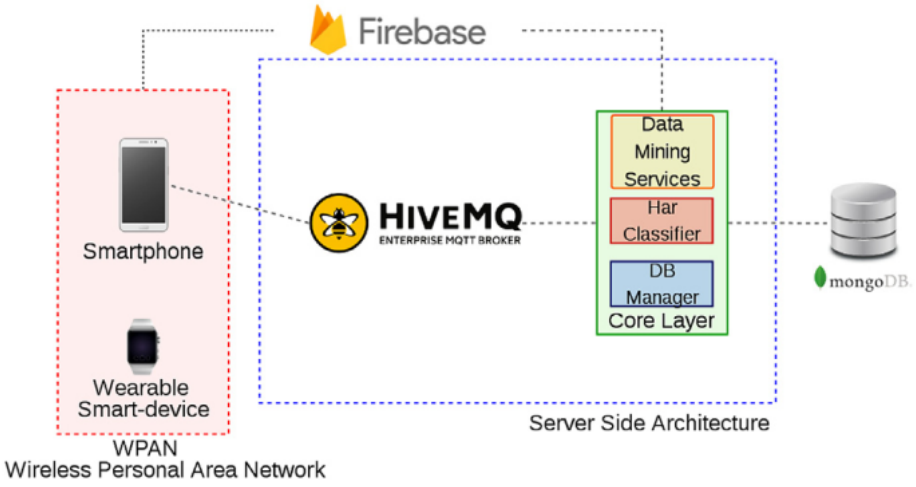

FIGURE 2.7: e-Health communication modules

Human Activity Recognition through a Fuzzy Logic-based classifier

Using the Fuzzy Logic (FL), it is possible to evaluate a wider set of possibilities than binary logic operators. To make this possible the Membership Degree (MD) of an object referred to a fuzzy set can realize the value in the range $[0, 1]$, instead, using binary logic the same value is restricted to the values 0 and 1 (false and true). In FL, the MD is to be intended as indicating "how much" property is true. Making relationship rules between input and output it is possible to describe systems or functions. One of the most usual inference methods is the Mamdani approach [32], divided into four main steps: input fuzzification, inference rule evaluation, aggregation, and defuzzification. Other methods exist such as Sugeno method [33], but for the case of study, Mamdani methods better fit the application.

Using several datasets and samples it has been verified that to build the HAR it is possible to consider only acceleration along X and Z axes.

In the first step of the classification process, we have to produce the training set, which is achieved exploiting the same sensors that we will use for the full application. In the learning phase, each data point in the cluster is marked with a membership degree. This value represents the importance of the point in the cluster. To accomplish this task we use an algorithm based on the Fuzzy C-Means algorithm described in [34]. The main goal of this task is to group data in clusters that identify the chosen Activities dataset.

The algorithm starts with some constants that are the number of clusters, a weighted component (*Fuzzifier*) called m and it is commonly chosen to be 2 and an initial membership matrix with some threshold values.

The Fuzzifier controls the class overlapping and helps us to assign a data point to a certain cluster. The MD represents the relevance of a given data point to a cluster. Initially, the MD is chosen randomly and these values are stored in the U matrix. A threshold is used to evaluate the convergence of the algorithm in the iterations. Let us explain how the algorithm works. We have a direct relationship between a class and cluster, in fact, for each class exists a dedicated cluster. In this work we consider the following classes: *Resting*, *Walking* and *Running*. Therefore, we start the learning phase with a preset to make faster the learning process.

In particular, during iteration 0 we introduce some data points for each cluster to make easier the process of finding the cluster center that it is given by the following equation.

$$C_j = \frac{\sum_{i=1}^{n} U_{i,j}^{m*} x_i}{\sum_{i=1}^{n} U_{i,j}}$$

where:

- n: number of data points
- j: number of current cluster
- x_i: is the i-th data point
- $U_{i,j}$: is the degree of membership of x_i in the j-th cluster and it be longs to the range $[0, 1]$

Value C_j is important to assign further values of $MD(U_{i,j})$. Since a data point may belong to different clusters the membership degree brings up the importance of data point in the considered cluster. Therefore, the formula to achieve value is herein reported:

$$U_{i,j} = \frac{1}{\sum_{k=1}^{n} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

Where the variables c_j is the center of j-th cluster and the c_k is the center of k-th cluster. As previously stated the variable m represents the fuzzification terms and it typically assumes the value of 2.

Data filtering

The classifier works on sampled data coming from IoT devices to recognize user activity. Lots of data may be collected and transferred from devices to the cloud-assisted system. These data may waste the performance of devices that have to spend a great amount of energy in the communications step. In order to reduce the number of transfer sessions, it is possible to aggregate data adding a filtering step on the sampled data. The filter is arranged into the devices and works real-time. Three filters are proposed to accomplish this task and are also compared with a raw transmission (C-data). Filter activities have to be tuned up to keep classifier goodness. In particular, the following filter options are considered:

- *C-Data (Continuous data)*: this means that the device sends continuous data to the remote service and no filter is applied before data submission.
- *S-Data (Sample data)*: this means that data are aggregated and periodically sent on the basis of a temporal window observation.
- *N-Filter(Normal Filter)*: in this case filter is applied to a data point and it acts taking into account the average of samples in the last temporal window and last recognized activity.
- *R-Filter (Restricted Filter)*: It acts on the data in a more effective way taking into account last recognized activity, data sample from last temporal window and historical data.

Results

The device performs continuous monitoring of the user by acquiring data coming from sensors, these data are used to perform real-time monitoring.
In fig.2.8 clusters of activities are depicted. It is interesting to observe that increasing the number of data points, cluster edges are overlapping with higher frequency. This

is a common issue in the HAR classifier and this happens because user behaviours are quite similar when performing certain activities for example walking and light running. The closer are data points to the edge of a cluster the harder is the work of the classifier. The goodness of the classifier is to put these points in the correct cluster assigning their membership degree to the clusters.

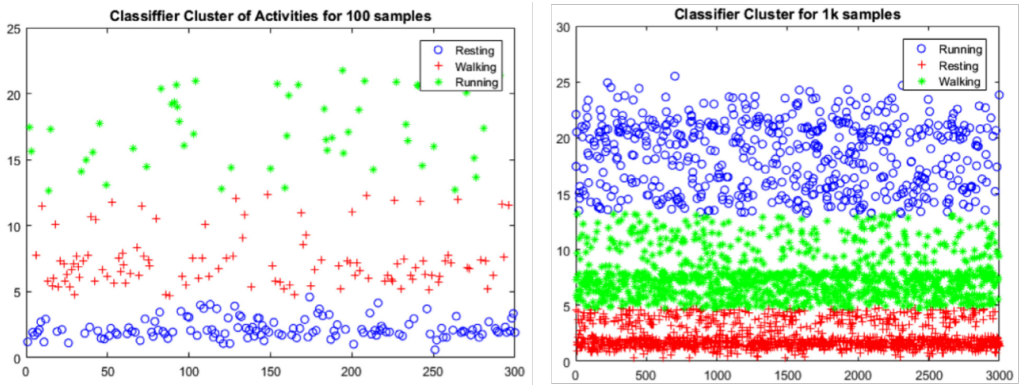

FIGURE 2.8: Clusters achieved with 100 and 1000 samples for each activity

The filter effects on system performances are analysed utilising a simple scenario where several e-health devices are employed to bring up data to a server node that has the main task to aggregate and make the first level of elaboration.
Filters are tested in the same environment and a comparison changing the number of nodes and type of filter has been carried out. We used several set of data produced by an IoT smart device through multiple connections to the server. Our goal is to verify how the system responds by changing filters type and number of sources.
The average serving time measured by the edge node is shown in fig. 2.9. Here the maximum serving time has been set to five seconds and the maximum buffer size has been set to 350 messages; this is made because old data may produce wrong results on e-health devices in terms of HAR. Moreover, it is possible to note that the stronger filter actions are the higher is the number of devices served by the system in a reasonable time. This can be measured because the number of messages related to the activity recognition generated by the devices decreased with the filtering action on the samples.

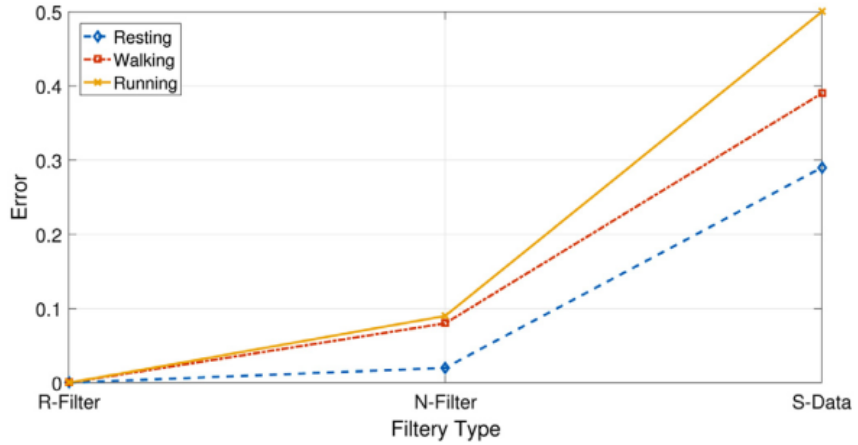

FIGURE 2.9: Errors for activity recognition made by the classifier with filtering

2.3.2 IoV application

In [35], an embedded device that uses a cooperative M2M system to analyze driving style in the IoV domain was designed. Moreover, a Cloud of Vehicle (CoV) layer is used to improve system resources in terms of computational power. Data are analyzed in the cloud domain to better fit driver dynamics that may change drastically following road condition. Data are collected in a real-time way from vehicles by the proposed on-board device. A classifier based on FL is proposed to implement an Artificial Intelligence (AI) module able to recognize driver behaviours and to suggest corrective actions to better fit road conditions.

IoT Architecture overview

To support the proposed architecture, we use a novel protocol based on lightweight M2M protocol to improve reliability and scalability.

A reference architecture is shown in fig.2.10.

Cloud service instances can exchange data with the Local Area Service (LAS), which is in charge of managing the area in which the vehicle is moving. Moreover, each LAS implements a transparent gateway for MQTT goals. Moreover, it instantiates mechanisms for congestion and flows control in the managed area. The CoV is composed of Cloud and LAS instances, and MQTT broker. To allow communication between vehicles and CoV, we design our protocol with the main goal to optimize

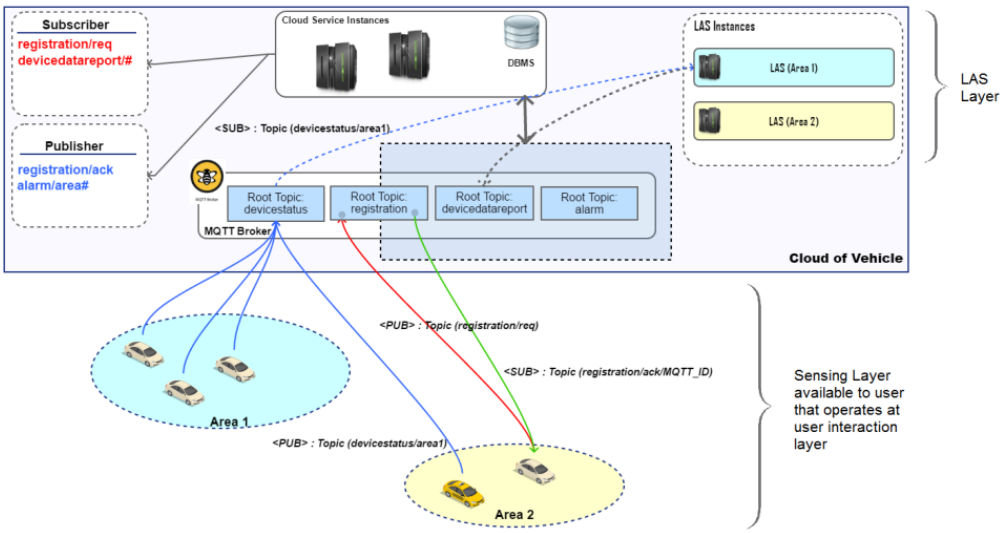

FIGURE 2.10: IoV communication architecture

network resources. The device continuously collects data regarding the vehicle and the environment. It is connected through an ad-hoc connection to a local internet gateway represented by a mobile node, which supplies a 4G data connection to reach LAS instances. LAS nodes are connected to cloud services. Regarding LAS, the main tasks are to collect and to elaborate data by preparing messages to be sent in the cloud layer. The remote services on the cloud, instead, elaborate data coming from LASs. They extract information about drivers to better fit AI classifier to the driver.

Software layer represents an important component of the whole IoV ecosystem. Cloud services exploit MQTT specifications for communicating with LAS instances. This module is deployed by using Eclipse Paho Java Client [30], an MQTT client library written in Java. Mobile devices are used to send collected environments data through 3G/4G connection towards the MQTT broker. Embedded devices connected to the vehicle gather data regarding their status and driving behaviour. These data are captured by two inner components called Danger Manager (DM) and Update Manager (UM). The proposed architecture uses two different types of messages: standard and alert message. These kinds of messages are collected by two software modules with a different computational priority, alert messages have a high priority due to their safety content used in case of dangerous situations. These managers store data in a database, which may be instantiated in a remote location to better

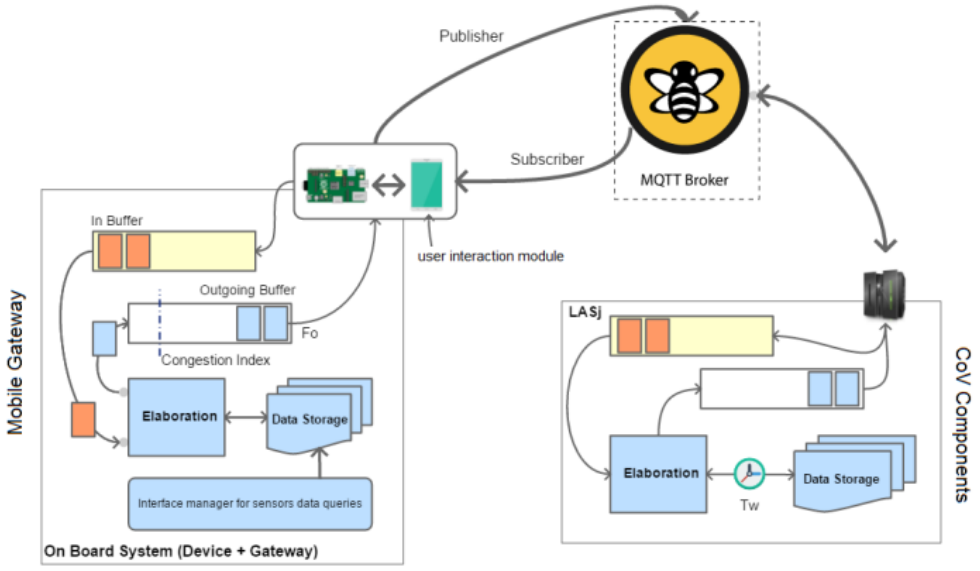

FIGURE 2.11: Protocol data management of proposed architecture

exploit distributed infrastructure and to achieve a more robust and flexible system. Moreover, the computational load can be better distributed along with the platform in order to obtain better performances and a reduced response time to the incoming queries. Each IoT device sends data to the MQTT broker as shown in fig.2.11. This kind of communication is performed exploiting smart-phone connection to the Internet.

Environment & Driver Classification

The whole classification process is made through two sub-processes that we can see as a two-step classifier. The first one is responsible to understand the context in which the driver is. The second step has the main task to understand the driver behaviour starting from the knowledge of the context obtained as the output of the first step. To gather needed data we have to use several sources. The most important is the vehicle that will supply several data such as speed, acceleration, consumption and so on. These data can give us important information about driver habits and dangerous behaviours on the road in an indirect manner.

The two-step classifier uses data acquired the sensing layer. In the sensing layer, the sensors are connected with the Raspberry Pi. This layer is very important because it allows acquiring a set of data needed to understand drivers behaviour and vehicle

performances in terms of fuel consumption, average speed and so on. Exploiting Raspberry Bluetooth we can directly connect the OBD-II interface that will allow us to gather data from CAN-BUS of the vehicle. Moreover, the Raspberry device will be equipped with accelerometer, gyro and GPS sensors. The inertial sensors are important, and they can be used to identify different behaviours of different users [36]. The environment classification represents an important step because it is strictly related to the driving style. To implement the classifier we used an approach based on Fuzzy Logic model. The models used in this work based on the method proposed in [32]. It implements the inference model in fuzzyfication step and centroid based algorithm in the de-fuzzification process. A global scheme of the classifier is shown in fig. 2.12.

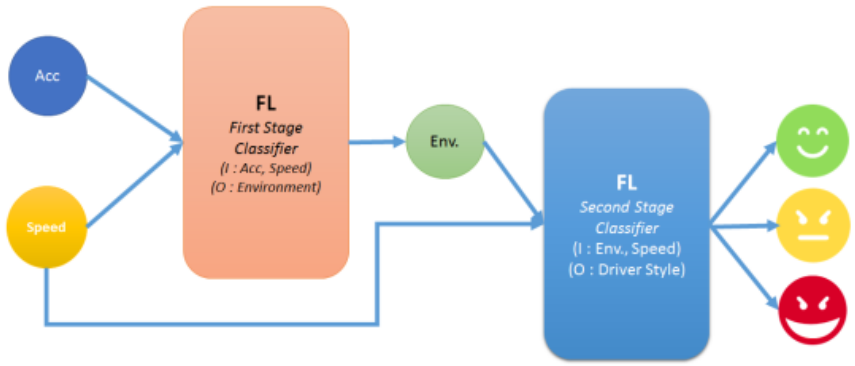

FIGURE 2.12: Fuzzy Model used for classification of the driver behavior

To characterize the different driving styles it is necessary to identify in-advance the environment in which the driver is moving. A human driver immediately recognizes the context that surrounds him but, to automate the detection of possible aggressive driving behaviour, it is necessary to identify the characteristics of the current environment. In this work, we choose as main characteristic parameters to recognize an environment the average speed and accelerations. Many observations have been made taking into account data coming from several users and vehicles. To gather data we directly interfaced an OBD module acquiring vehicle dynamics such as fuel consumption, acceleration/deceleration, torque, etc. Using this information we built Probability Density Function (PDF) for urban, sub-urban and extra-urban

environments to automatically recognize the MD in each case. The model has been realized by using Matlab software on pre-acquired data samples.

Results

To perform classification tasks, we tested the platform by sampling several data coming from different trips and tracks. In the environment classification, we consider both speed and acceleration contribute. Clusters of this classifier are depicted in fig. 2.13.

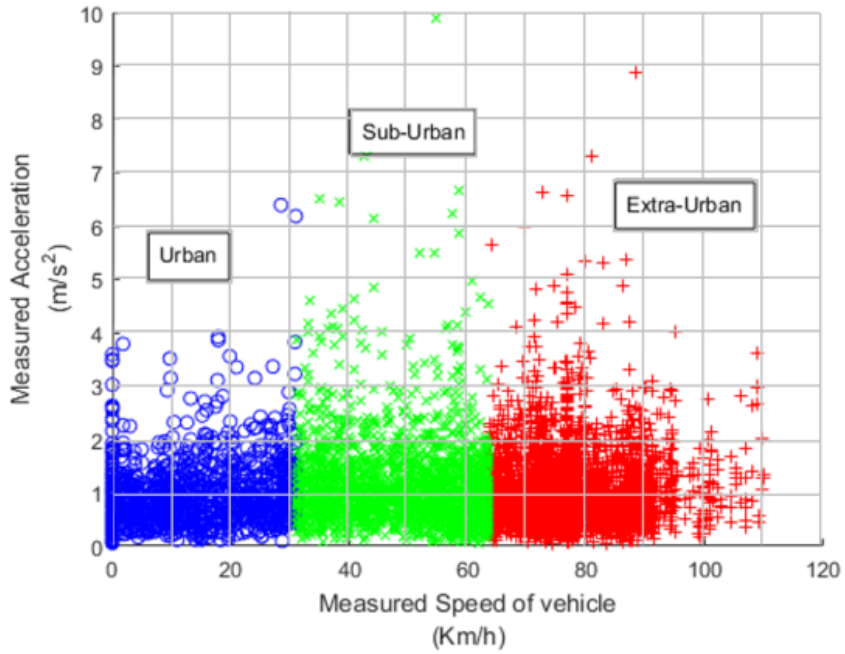

FIGURE 2.13: Environment classifier achieved by using speeds and accelerations

We use full information to recognize a driver's behaviours and investigate how our system can affect drivers. One of the main goals is to achieve a more eco-friendly driving style reducing fuel consumption and CO_2 emissions. Moreover, observing drivers actions and behaviours, it may be possible to notice alerts to the users. From the acquired dataset, we compared the consumption trends of some drivers classes. We consider aggressive and normal drivers. Both classes of drivers present some characteristics on measured speed and accelerations in all kinds of environment that help us to define classes. We reported the consumption trend measured along an

observation period of one month to verify if a more eco-friendly driving style is reached by the most aggressive drivers. It is possible to see as the drivers under the feedback and alerting triggered by the classifier on CoV can affect the driver behaviour reducing the fuel consumption and increasing the number of km per litres. In this case, aggressive users can improve more the performance because of their aggressive style, when mitigated, produces immediate energy and CO_2 reduction. The improvement is less significant for the normal driver where the fuel consumption is not so high and it is not so affected by a bad driving style.

In fig. 2.14 the aggressive occurrences trend along one month of observations is shown.

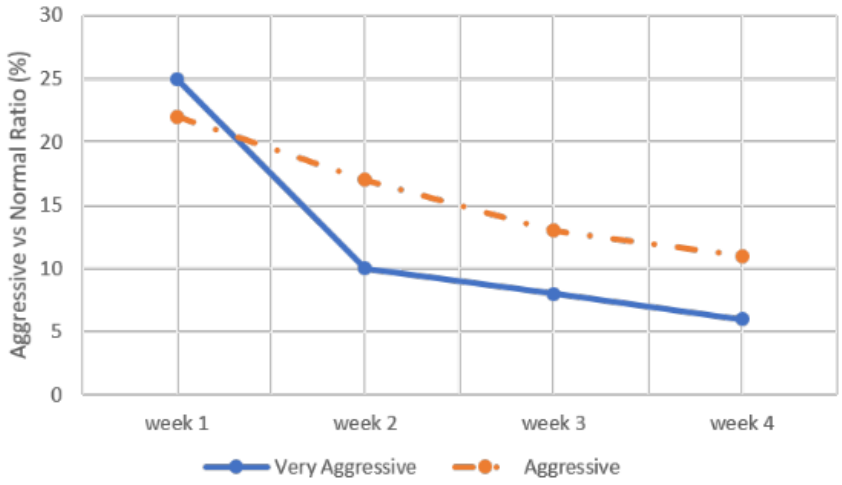

FIGURE 2.14: Aggressive occurrences trend along one month of observations

The drivers that carried out these results have made the measurements along the same tracks changing their behaviour when an alarm was raised by the proposed system. Results are very impressive in terms of average speed and accelerations of the aggressive drivers. A safer and careful driving style is reported by the observations that were conducted along a medium period. Our tests have been carried out using similar tracks on different users on a period of one month. In particular, for our tests, we have considered three different types of vehicles.

2.4 Chapter summary

In this chapter, the concepts of Cloud, Edge and Fog computing was described. A general IoT communication architecture was proposed. The core-entity of the architecture is a Cloud-based Platform. Using the study on IoT network communication protocols analyzed in the first chapter it is possible to use an IoT Gateway multi-interfaces. In this way, smart IoT devices can communicate directly with the Cloud-based platform or use the IoT smart gateway. The proposed IoT communication architecture has been specialized and used in e-Health and IoV cotexts. Both proposed solutions are based on the MQTT protocol and the use of a Fuzzy Logic classifier.

The next chapters will introduce the concepts of Data Mining and Machine Learning applied to the Smart Home with a particular focus on the use of neural networks for predicting energy consumption and supporting comfort management.

Chapter 3

Data analysis and Machine Learning techniques in IoT

In this chapter, the concepts of Data Mining (DM) and Machine Learning (ML) will be introduced with a focus on neural networks. After a description of the basic architecture of a neural network, the difference between Single-layer Neural Networks and Multi-layer Neural Networks will be shown. Among the various applications of neural networks, the use of neural networks in time-series forecasting and prediction problems will be analyzed in detail. The use of Recurrent Neural Networks and in particular of Long Short-Term Memory (LSTM) was the natural choice for the problem of time-series forecasting and prediction. The LSTM networks usage to predict electric consumption in a different environment will be provided. Finally, a comparison of the LSTM prediction results with other data mining techniques will be given.

3.1 Data analysis, Machine Learning techniques

Knowledge Discovery in Databases (KDD) is the process that provides the "*non-trivial extraction of implicit, previously unknown and potentially useful information from data*". [37]

Data Mining is an appropriate step in this process with the use of specific algorithms for extracting patterns from data. Data mining is the study of collecting, cleaning, processing, analyzing, and obtaining useful insights from data. A wide variation

exists in terms of the problem domains, applications, formulations, and data representations that are encountered in real applications. Data mining is a term that is used to describe these different aspects of data processing. The data mining process is a pipeline containing many phases such as data cleaning, feature extraction, and algorithmic design. [38]

The workflow of a common data mining application contains the following phases and is depicted in fig. 3.1:

- Data collection
- Feature extraction and data cleaning
- Analytical processing and algorithms

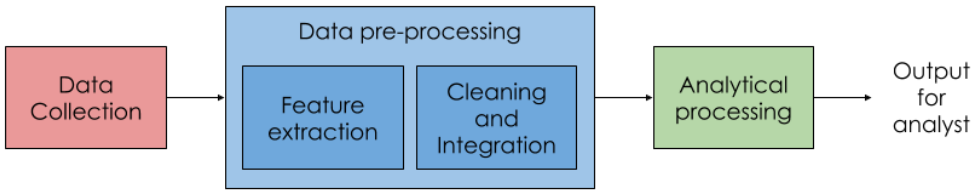

FIGURE 3.1: Data processing pipeline

There is a significant overlap between Machine Learning and Data Mining techniques. These two terms are generally confused because they often employ the same methods and consequently overlap significantly. Arthur Samuel defined Machine Learning as a "*field of study that gives computers the ability to learn without being explicitly programmed*".

ML focuses on classification and prediction, based on known properties previously learned from the training data. DM focuses on the discovery of previously unknown properties in the data. The term Data Mining was introduced in the late 1980s (the first KDD conference took place in 1989), whereas the term Machine Learning has been in use since the 1960s. An ML approach usually consists of two phases: training and testing. Often, the following steps are performed:

- Identify class attributes (features) and classes from training data
- Identify a subset of the attributes necessary for classification

- Learn the model using training data
- Use the trained model to classify the unknown data

For most ML approaches, there should be three phases: training, validation, and testing. ML and DM methods often have parameters such as the number of layers and nodes for an ANN. After the training phase is complete, there are usually several models available. To determine which one to use and have a good evaluation of the error it will achieve on a test-set, there should be a part separate dataset, the validation dataset. The model that performs the best on the validation data should be the model used. The accuracy reported is optimistic and might not reflect the accuracy that would be obtained on another test-set.
There are three main types of DM/ML approaches: unsupervised, semi-supervised, and supervised. In unsupervised learning problems, the main task is to find patterns, structures, or knowledge in unlabeled data. When a portion of the data is labelled during the acquisition or by human experts, the problem is called semi-supervised learning.
The addition of the labelled data greatly helps to solve the problem. If the data are completely labelled, the problem is called supervised learning and generally, the task is to find a function or model that explains the data. [39] [40]

There are several applications and methods of Data Mining and Machine Learning tasks as:

- *Classification*: many DM/ML problems are focused on a specialized goal that is represented by the value of a particular feature in the data. This particular feature is referred to as the class label. In these supervised problems, the relationships of the remaining features in the data concerning this special feature are learned. The data used to learn these relationships is related to the training data. The learned model is used to establish the estimated class labels for the data where the class-label is missing.

- *Outlier detection*: An outlier is a data point that is significantly different from the remaining data. A possible formal definition of the concept of an outlier is "An outlier is an observation that deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism".

- *Association rules*: the goal of Association Rules is to discover previously unknown association rules from the data. An association rule describes a relationship among different attributes: IF (A AND B) THEN C. This rule describes the relationship that when A and B are present, C is present as well. Association rules have metrics that tell how often a given relationship occurs in the data. The support is the prior probability (of A, B, and C), and the confidence is the conditional probability of C given A and B.

- *Clustering*: a set of techniques for finding patterns in high-dimensional unlabeled data [41]. It is an unsupervised pattern discovery approach where the data are grouped based on a similarity measure. The main advantage of clustering for intrusion detection is that it can learn from audit data without requiring the system administrator to provide explicit descriptions of various attack classes.

3.2 Neural networks introduction

Artificial neural networks are common machine learning techniques that simulate the mechanism of learning in biological organisms. [42]
The biological neurons connection mechanism is simulated in artificial neural networks with computation units that are referred to as neurons. The computational units are linked together through weights. Each input to a neuron is computed with a weight, which affects the function calculated at that unit. An artificial neural network computes a function of the inputs by propagating the computed values from the input neurons to the output neurons and using the weights as intermediate parameters. Learning occurs by changing the weights connecting the neurons. The external impulse needed for learning in artificial neural networks is provided by the training data containing examples of input-output pairs of the function to be learned.

3.3 Basic Architecture of Neural Networks

The basic architecture of Neural Networks can be divided into Single-layer Layer Neural Networks and Multi-layer Neural Networks. In the single-layer neural network also called perceptron, a set of inputs is directly mapped to output by using a generalized variation of a linear function. In the multi-layer neural network also called feed-forward network, there are input and output layers separated by a group of hidden layers.
A description of single-layer neural networks and multi-layer neural networks are provided below.

3.3.1 Single-layer Neural Networks and Multi-layer Neural Networks

The neural network that contains a single input layer and an output layer is typically called perceptron.
The input layer contains n nodes that transmit the input features $\overline{X} = [x_1...x_n]$ with weight $\overline{W} = [w_1...w_n]$ to the output node.
The linear function $\overline{W} \cdot \overline{X} = \sum_{j=1}^{n} w_j x_j$ is computed at the output node to make a prediction $\widehat{y} \in \{-1, +1\}$ as follow:

$$\widehat{y} = f\left\{\overline{W} \cdot \overline{X}\right\} = f\left\{\sum_{j=1}^{n} w_j x_j\right\} \tag{3.1}$$

The prediction error is $E\left(\overline{X}\right) = y - \widehat{y}$. In the architecture of the perceptron the single input layer transmits the features to the output node through the weighted edges $w_1...w_n$ with which the features are multiplied and added at the output node. The activation function f is applied in order to convert the aggregated value into a class label. Different activation function $\varphi(z)$ can be used to simulate different types of models like:

- Sign function : $\varphi(z) = sign(z)$
- Sigmoid function : $\varphi(z) = \frac{1}{1+e^{-z}}$
- Hyperbolic tangent function : $\varphi(z) = \frac{e^{2z}-1}{e^{2z}+1}$

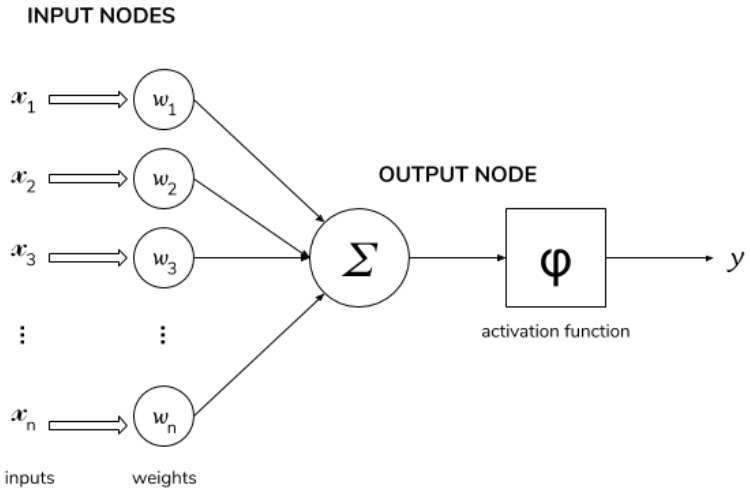

FIGURE 3.2: Single-layer Neural Networks

In many cases there is an invariant part of the prediction called bias. The bias can be added using a *bias neuron*, the weight of the bias edge provides the bias variable.

$$\widehat{y} = f\left\{\overline{W} \cdot \overline{X} + b\right\} = f\left\{\sum_{j=1}^{n} w_j x_j + b\right\} \tag{3.2}$$

The perceptron was designed to minimize the number of classification errors. The training phase of the neural network works by feeding each input data instance X into the network one by one or in batch to create the prediction $\widehat{y}$. The weights update is based on the error value $E\left(\overline{X}\right) = y - \widehat{y}$, in particular when the data point $\overline{X}$ is fed into the network the weight vector $\overline{W}$ is updated as follows:

$$\overline{W} \Leftarrow \overline{W} + \alpha\left(y - \widehat{y}\right)\overline{X} \tag{3.3}$$

The parameter α regulates the learning rate of the neural network on the learning cycles. Each training cycle can be called an *epoch*. The neural network cycles through the points in random order during the training phase and changes the weights to minimize the prediction error on that point.

The perceptron algorithm has good performance with *linearly separable* data sets 3.3 (a). To cope *non-linearly separable* data sets problems 3.3 (b), more complex neural networks architectures are needed.

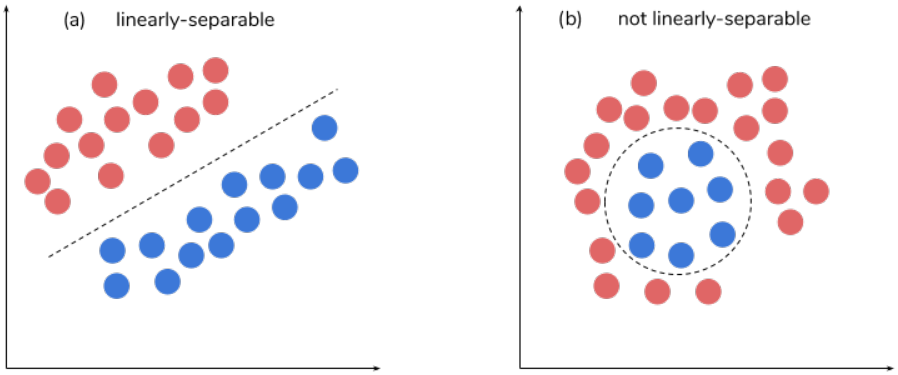

FIGURE 3.3: Linearly-separable and not linearly-separable data

In Multi-layer Neural Networks there are more than one computational layer. Unlike Single-layer Neural Networks, between input and output layer there are *hidden layers* with computations not visible to the user. The addition of the hidden layers allow to extends single-layer capabilities to solve non-linear classification problems. The hidden nodes process the information received from the input nodes and pass them to the output layer. The learning phase in Multi-layer Neural Network is an extension of Single-layer Neural Networks.

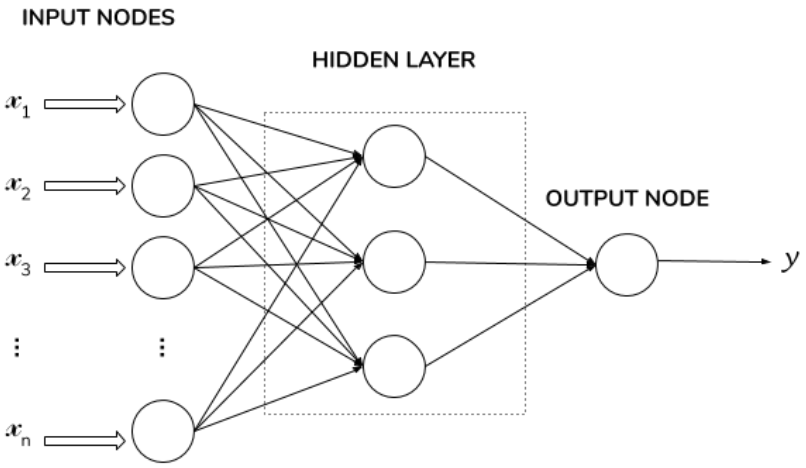

FIGURE 3.4: Multi-layer Neural Networks

3.4 Neural networks applications

Neural Networks are frequently more robust of other computational systems in solving different problems as:

- *Pattern classification*: use supervised learning to assign an unknown input pattern to one of several pre-specified classes based on properties that characterize a given class.
- *Clustering*: use unsupervised learning to form the clusters by exploring the similarities/dissimilarities between the input patterns based on their inter-correlations. The neural network assign "similar" item to the same cluster.
- *Function approximation*: training a neural network on input-output data to approximate the rules relating the input to the outputs. This function approximation is applied to problems where no theoretical model is available using data obtained from experiments or observations.
- *Forecasting*: training a neural network on samples from time-series data in a given scenario and use these data to make a forecast and predict future data. The neural network predict $\varphi(t+1)$ from previous historical observation $\varphi(t-2)$,$\varphi(t-1)$, $\varphi(t)$ where t represents the time-step of the observations.
- *Optimization*: find a solution that maximizes or minimizes an objective function with a set of constraints. Neural networks are efficient in solving complex and non-linear optimization problems.
- *Association*: training a neural network on ideal noise-free data to classify noise-corrupted data. The associative network can be used to reconstruct corrupted or missing data.

3.5 Common Neural Networks Architectures

There are a huge number of Neural Networks architectures used to solve several machine learning problems in different application domains. [43] A brief description of the most frequently used Neural Networks architectures is given below.

- **Back-propagation Networks**: a multi-layer neural networks with an input layer, output layer and one or more hidden layers. These networks can learn the mapping from one data space to another using example and supervised learning. The error computed to the output layer is propagated backward to the hidden layer. Back-propagation networks are used for classification, forecasting, data modeling, data and image compression and pattern recognition.

- **Recurrent Neural Networks**: designed for sequential data like time-series with the input $\overline{x}_1 \cdots \overline{x}_n$ where $\overline{x}_t$ is a d-dimensional data at time t. Given an input $\overline{x}_t$ at each timestamp, an hidden state $\overline{h}_t$ changes at each timestamp when new data points need to be analyzed. Each timestamp can have an output value $\overline{y}_t$ that can be the prediction value of $\overline{x}_{t+1}$. The hidden state at time t is given by a function of the input vector at time t and the hidden vector at time $t-1$ as follows:

$$\overline{h}_t = f\left(\overline{h}_{t-1}, \overline{x}_t\right) \tag{3.4}$$

  A function $\overline{y}_t = g\left(\overline{h}_t\right)$ is used to learn the output probabilities from hidden states.

- **Restricted Boltzmann Machines**: neural network architecture for modelling data in an unsupervised way that uses the concept of energy minimization. Restricted Boltzmann machines are particularly useful for creating generative models of the data and dimensional reduction in supervised/unsupervised modelling. The supervised training is often preceded by an unsupervised phase called the pre-training phase. These neural networks were the first model used for deep learning and the pre-training approach was adopted by other types of models. The training process of a restricted Boltzmann machine requires Monte Carlo sampling.

- **Convolutional Neural Networks**: are biological inspired networks typically used in computer vision for image classification and object detection.
  In the convolutional neural network architecture, each layer of the network is 3-dimensional with two types of layers called respectively *convolution* and *subsampling* layers. These neural networks are used in image recognition, object

detection with a large amount of available training data for complex visual tasks. The convolutional neural networks are used also in voice recognition and natural language processing.

3.6 Neural Networks application in time-series forecasting and prediction

A time series is a sequence of numbers in chronological order. Traditional time series analysis uses mathematical-statistical methods to analyze this sequence and predict the future development of things. Time series analysis is commonly used in macroeconomic control of a national economy, enterprise management and management, market potential forecast, meteorological forecast, astronomy and oceanography. In the application field of this thesis work, the use of neural networks for the analysis of energy consumption data was investigated. In particular, the use of Recurrent Neural Networks is the natural choice for time-series forecasting and prediction. [44] The Recurrent Neural Networks are networks with loops in them that allowing information to persist and pass information from one step of the network to the next steps. These neural networks connect previous information to the present task.

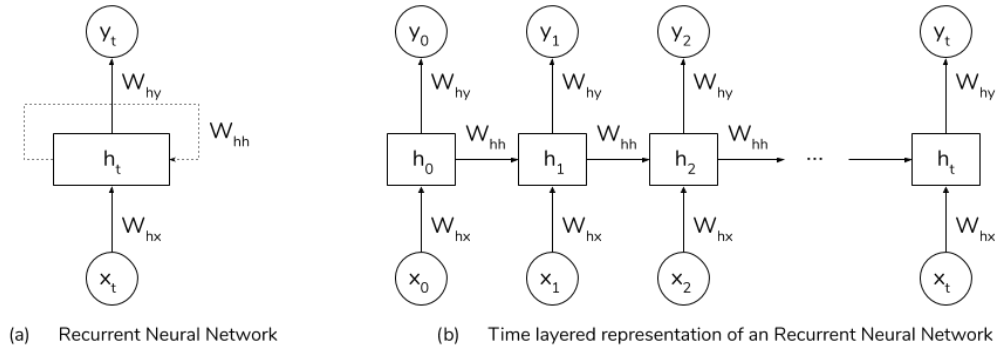

FIGURE 3.5: Recurrent Neural Networks representations

LSTM networks introduced in [45] are a special kind of Recurrent Neural Networks designed to avoid long-term dependency. In LSTM networks there are memory blocks that are connected through successive layers. Each block contains gates

that handle the state of the block and the output. [46]
There are three different types of gates in LSTM:

- *Input Gate*: sets the values from the input to update the memory state based on defined conditions
- *Forget Gate*: sets the values to throw away from the block based on defined conditions
- *Output Gate*: sets the output value based on the input and the state of the block based on defined conditions

An LSTM block receives an input sequence and then each gate uses activation units to decide the operation to do. These operations make the change of state of the block and the addition of information that through the block. The gates have weights that can be learned during the training phase. This makes the LSTM blocks smarter than classical neurons and enables them to memorize recent sequences.
Each LSTM block can be considered a memory unit and contains a cell with state c_t at time t. The inputs of an LSTM block are:

- the current input x_t
- the previous hidden state of all LSTM blocks in the same layer h_{t-1}

Given an input time series $\overline{x} = \{x_1, x_2, \cdots, x_n\}$ the LSTM network maps these input values in two time sequences $\overline{h} = \{h_1, h_2, \cdots, h_n\}$ and $\overline{y} = \{y_1, y_2, \cdots, y_n\}$.

The forget gate f_t is used to decide what information throw away from the cell state, the activation of forgetting gate uses a sigmoid function $\sigma(\cdot)$ as

$$f_t = \sigma\left(W_{xf}\, X_t + W_{hf}\, h_{t-1} + W_{cf}\, c_{t-1} + b_f\right) \tag{3.5}$$

The output of f_t function is a value between 0 and 1, the value 0 means forgetting the last state C_{t-1} completely and the value 1 means to keep the last state C_{t-1} completely.

The input gate i_t is used to decide what new information is going to be stored in the cell state, the activation of input gate uses a sigmoid function as

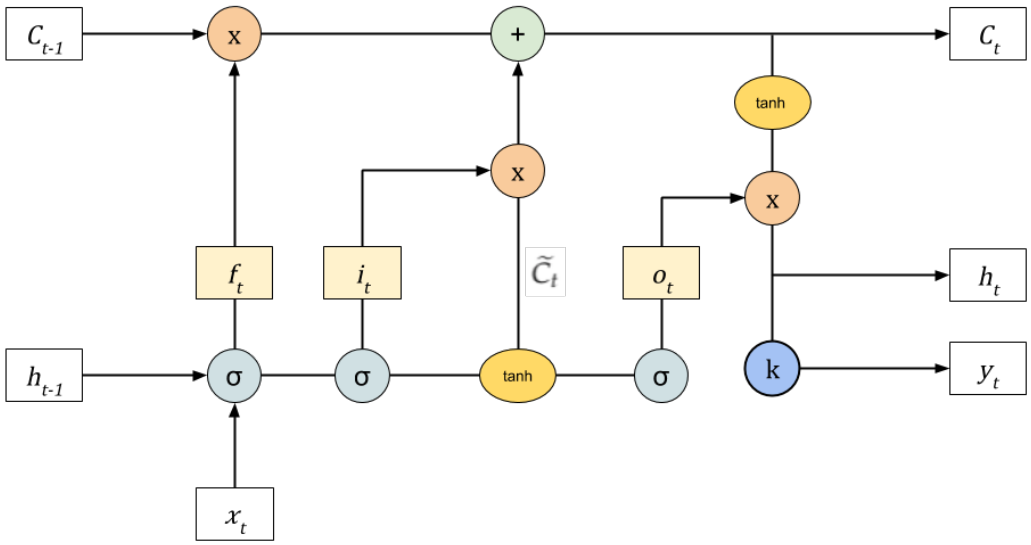

FIGURE 3.6: Architecture of LSTM block

$$i_t = \sigma\left(W_{xi}\, X_t + W_{hi}\, h_{t-1} + W_{ci}\, c_{t-1} + b_i\right) \tag{3.6}$$

To update the old cell state C_{t-1} into the new cell state C_t, the old state is multiplied by the output of the forget gate f_t to forget the information from the last state and added to $i_t \cdot \widetilde{C}_t$ to decide how much new information to be updated to the new cell state.

$$\widetilde{C}_t = tanh\left(W_{xc}\, X_t + W_{hc}\, h_{t-1} + b_c\right) \tag{3.7}$$

$$C_t = f_t\, c_{t-1} + i_t\, \widetilde{C}_t \tag{3.8}$$

The output gate o_t uses a sigmoid function to filter and output the cell state as

$$o_t = \sigma\left(W_{xo}\, X_t + W_{ho}\, h_{t-1} + W_{co}\, c_t + b_o\right) \tag{3.9}$$

The h_t output of LSTM cell uses the output gate value o_t and multiply it by the result of cell state C_t passed through a *tanh* function.

$$h_t = o_t\, tanh(c_t) \tag{3.10}$$

The y_t output of the LSTM block uses an output activation function $k\left(\cdot\right)$ as

$$y_t = k\left(W_{yh}\, h_t + b_y\right) \tag{3.11}$$

The W matrices used in the previous analytical formulations represent respectively:

- *input weight matrices*: $W_{ix}, W_{fx}, W_{ox}, W_{cx}$
- *recurrent weight matrices*: $W_{ih}, W_{fh}, W_{oh}, W_{ch}$
- *output weight matrices*: W_{yh}
- *connections weight matrices*: W_{ic}, W_{fc}, W_{oc}

The vectors b_i, b_f, b_o, b_c, b_y are the different bias vectors used.

3.6.1 Long Short-Term Memory for time-series forecasting in Home and Office environments

In the next subsections, the use of LSTM networks to predict electric consumption in a different environment is provided. Two different datasets are used to evaluate LSTM model in two different contexts. The results presented in the following sections are related to the best configuration of the neural network models obtained in the experimental tests. [47]
In [48] the authors demonstrate that LSTM-based RNN is capable of forecasting accurately the complex electric load time series with a long forecasting horizon.
In [49] the authors propose a long short-term memory neural network-based framework for the task of short-term load forecasting for individual residential households.
The LSTM networks can be used to accurately forecast the output power of photovoltaic systems. LSTM usage offers a further reduction in the forecasting error compared with the other methods.[50]

LSTM-based electric consumption forecasting in the home environment

For the electric consumption forecasting in the home environment, it was used the dataset "Individual household electric power consumption" downloaded from the

UCI Machine Learning Repository (`https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption`) [51].

This archive contains over 2 million measurements gathered in a house located in France between December 2006 and November 2010 (47 months).

Data Set Characteristics:	Multivariate, Time-Series	**Number of Instances:**	2075259	**Area:**	Physical
Attribute Characteristics:	Real	**Number of Attributes:**	9	**Date Donated**	2012-08-30
Associated Tasks:	Regression, Clustering	**Missing Values?**	Yes	**Number of Web Hits:**	373171

FIGURE 3.7: Individual household electric power consumption dataset info

The attribute information of the dataset are:

- *date*: Date in format dd/mm/yyyy
- *time*: time in format hh:mm:ss
- *global active power*: household global minute-averaged active power [kW]
- *global reactive power*: household global minute-averaged reactive power [kW]
- *voltage*: minute-averaged voltage [V]
- *global intensity*: household global minute-averaged current intensity [A]
- *sub metering 1*: energy sub-metering 1 (in watt-hour of active energy). It corresponds to the kitchen, containing mainly a dishwasher, an oven and a microwave (hot plates are not electric but gas-powered).
- *sub metering 2*: energy sub-metering 2 (in watt-hour of active energy). It corresponds to the laundry room, containing a washing machine, a tumble-drier, a refrigerator and a light.
- *sub metering 3*: energy sub-metering 3 (in watt-hour of active energy). It corresponds to an electric water heater and an air-conditioner.

After removing the rows with empty values and reshaping the columns of the original dataset, the resulting dataset has 2042187 measures with the following attributes:

- timestamp

- Voltage [V]
- Global intensity [A]
- Power [kWh]

From these attributes, the following features are extracted:

- Power [kWh]
- Year
- Month
- Day
- Hour

The data has been aggregated into "hourly" values and the entire dataset was further partitioned to obtain training-set and test-set to be used for the training and testing phases of the neural network.
The evaluation of the prediction model was made using Keras [52], an open-source deep-learning library written in Python. In particular, a Sequential model was used with 50 LSTM units and a single output layer. The used model uses the following parameters:

- *Optimizer*: Adam algorithm with a stochastic gradient descent method that is based on the adaptive estimation of first-order and second-order moments.
- *Loss function*: Mean Absolute Error that computes the mean of the absolute difference between target values and predicted values.

The training phase of the model uses two further parameters:

- *batch size*: number of samples per gradient update
- *epochs*: number of iteration to train the model

Simulations with different time horizons were carried out. In particular, the following scenarios were analyzed:

- *Month by month* forecast with training on 742 samples, validation on 658 samples and batch-size equal to 24 samples
- *Year by year* forecast with training on 8696 samples, validation on 8782 samples and batch-size equal to 168 samples (24*7)

The dataset used for month-by-month forecast split into training-set and test-set is shown in the figures 3.8 and 3.9.

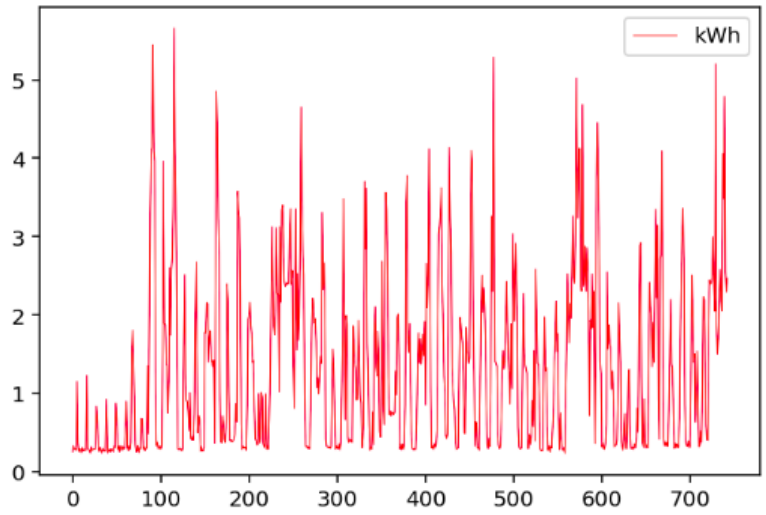

FIGURE 3.8: Training-set for home environment forecast

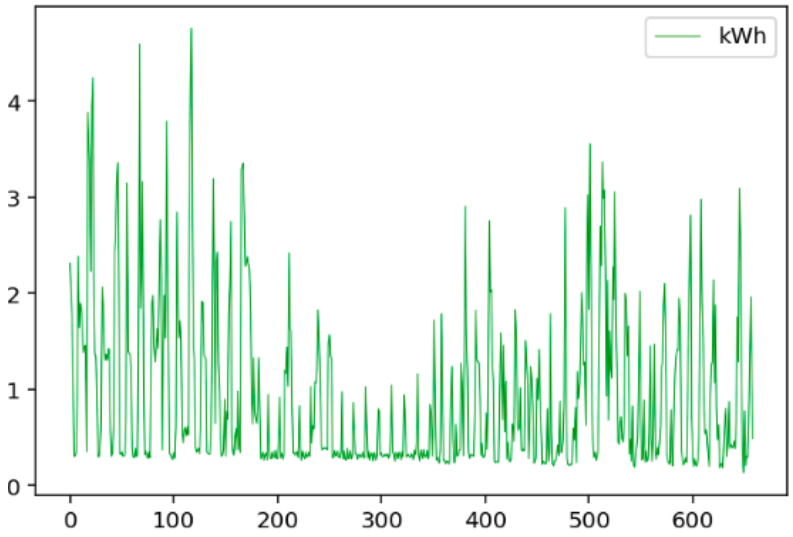

FIGURE 3.9: Test-set for home environment forecast

Figure 3.10 shows the result of the training phases over the 50 epochs iterations on a dataset with the corresponding training loss values and validation loss values.

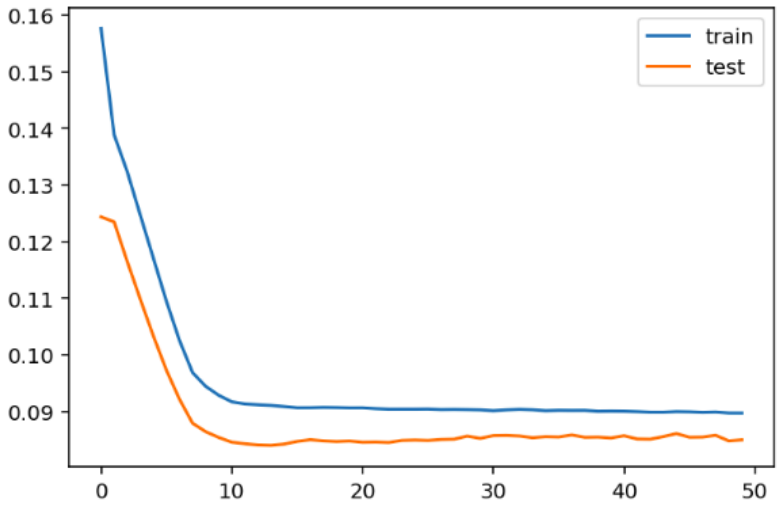

FIGURE 3.10: Training-model result in home environment

The output of the model prediction phase is represented in figure 3.11. The red scatters portion of the chart depict the real energy consumption and the blue line represents the consumption forecast.
The Root Mean Squared Error (RMSE) that computes the mean of squares of errors between real values and predictions is equal to **0.744**.

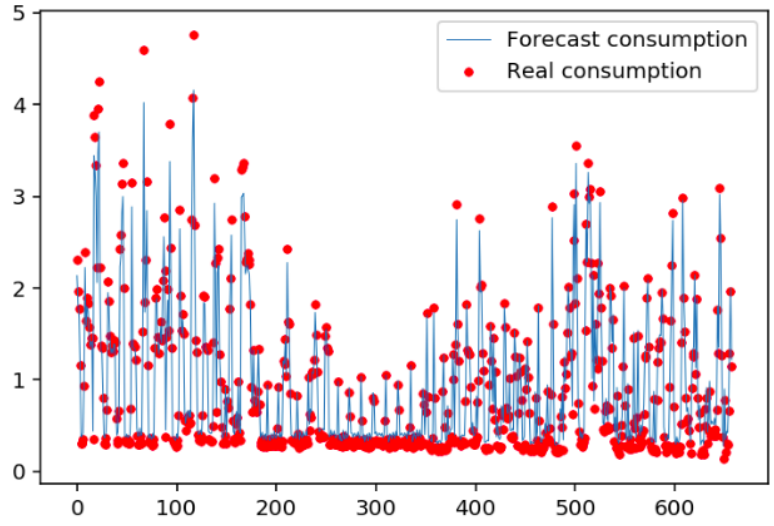

FIGURE 3.11: Monthly consumption forecast in home environment

The dataset used for year-by-year forecast split into training-set and test-set is shown in the figures 3.12 and 3.13.

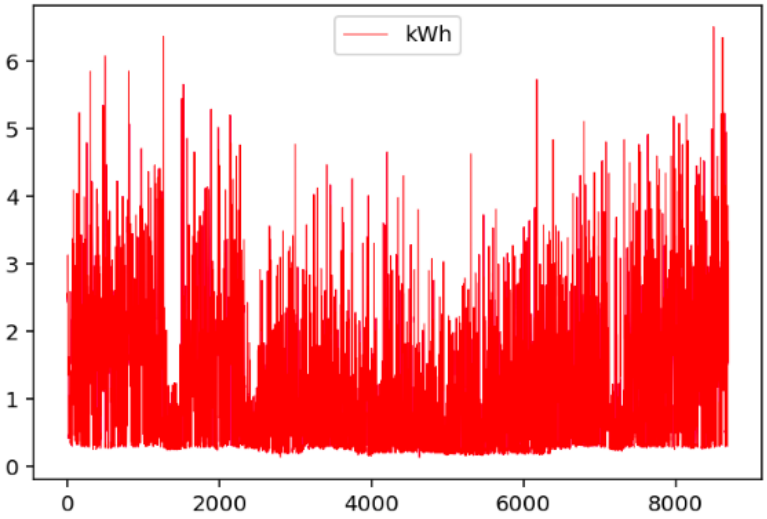

FIGURE 3.12: Training-set for home environment forecast

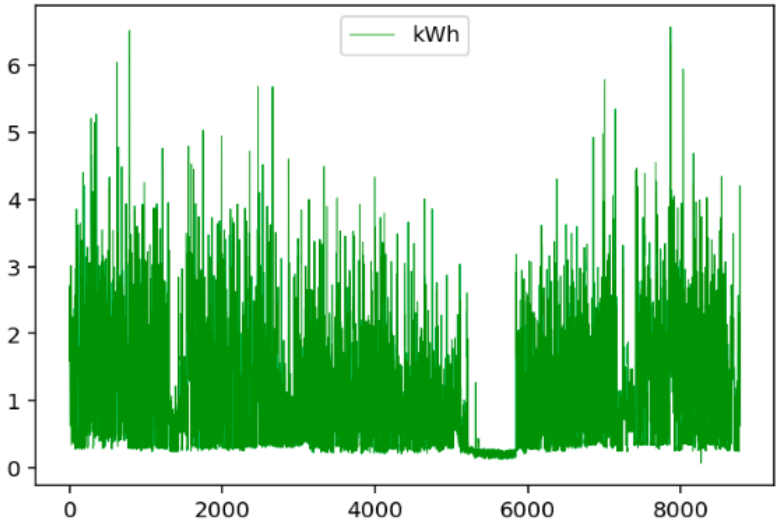

FIGURE 3.13: Test-set for home environment forecast

The result of the training phases for year-by-year prediction over the 50 epochs iterations on a dataset with the corresponding training loss values and validation loss values is shown in figure 3.14. The output of the model prediction phase for

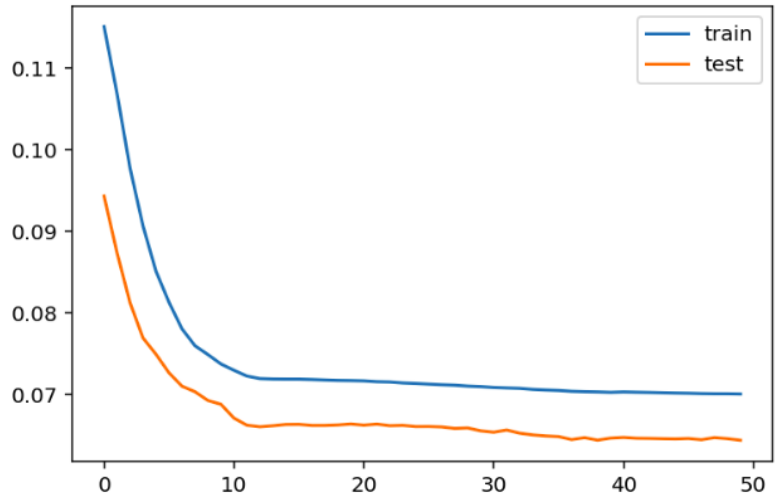

FIGURE 3.14: Training-model result in home environment for annual forecast

year-by-year prediction is represented in figure 3.15. The red scatters portion of the chart depict the real energy consumption and the blue line represents the consumption forecast. The RMSE for this simulation is equal to **0.640**.

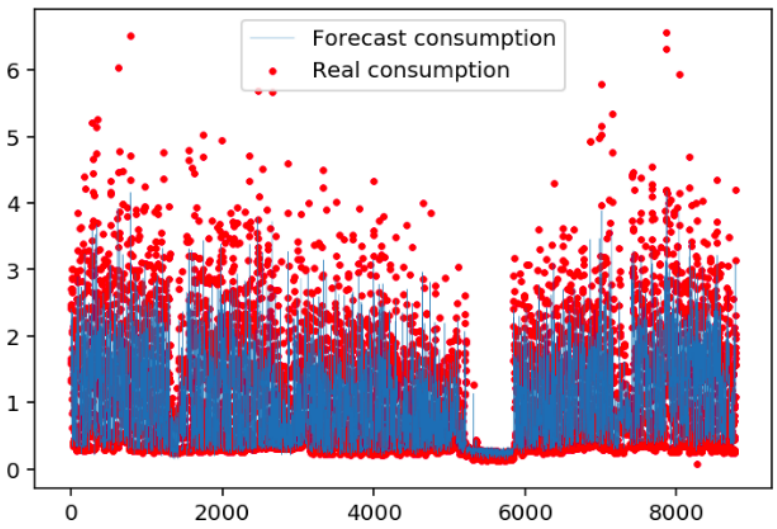

FIGURE 3.15: Annual consumption forecast in home environment

LSTM-based electric consumption forecasting in the office environment

The data used to analyze energy consumption in an office environment are gathered used a smart meter device that collects electrical consumption every minute. This dataset contains over 500.000 measurements collected in an office located in Calabria between January 2019 and February 2020.

The attribute information of the dataset are:

- *device ID*: unique device identifier
- *timestamp*: date in format dd/mm/yyyy hh:mm:ss
- *total consumption*: cumulative energy consumption
- *real-time consumption*: energy consumption measured in the last minute

The data has been aggregated in "hourly" and "daily" values, also the entire dataset was partitioned into training-set and test-set.

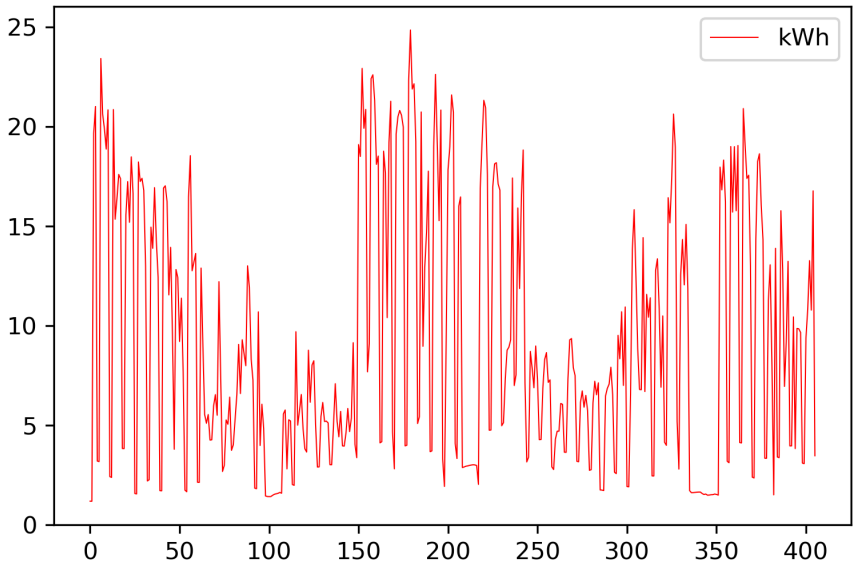

FIGURE 3.16: Daily energy consumption in office environment

The simulations have been carried out with training on 345 samples, validation on 60 samples and batch-size equal to 7 samples.

The result of the training phase over the 50 epochs iterations on a dataset is shown in figure 3.17.

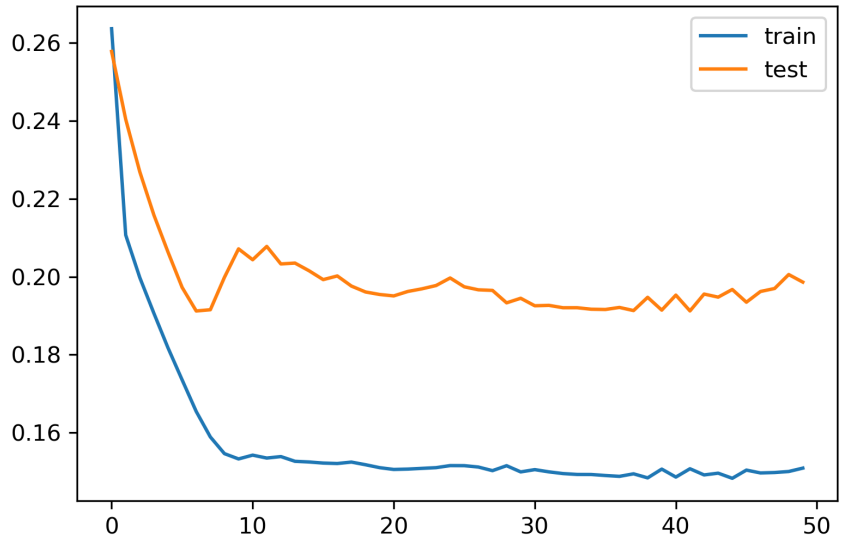

FIGURE 3.17: Training-model result in home environment

The output of the model prediction phase in office environment is depicted in figure 3.18. The blue line represents the real consumption and the orange line represents the energy consumption forecast.
The RMSE for this simulation is equal to **3.214**.

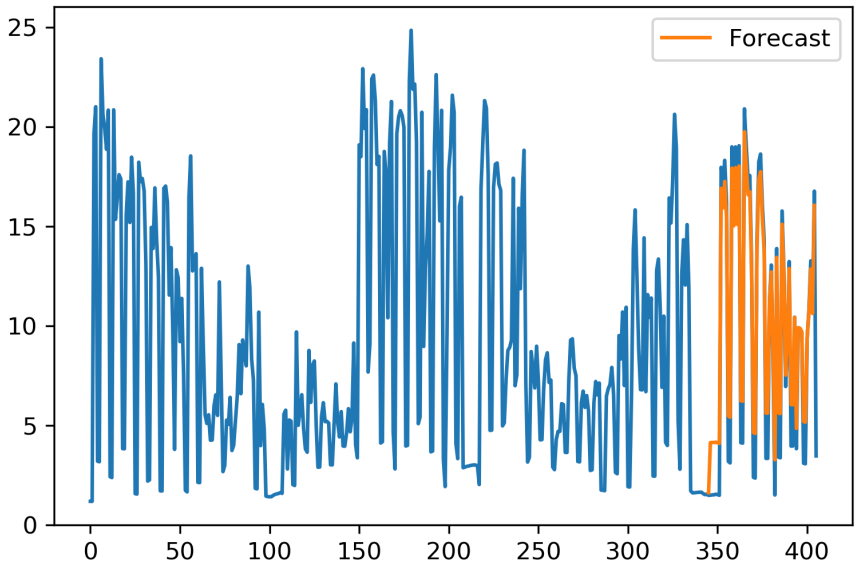

FIGURE 3.18: Daily consumption forecast in office environment

Better visualization of the prediction model results is shown in the figure 3.19. The consumption patterns in the office environment predict very different consumption between working days and weekends.

From figure 3.19 it is possible to see how the forecast model can predict with good results the alternation between working days with higher consumption and non-working days with reduced consumption.

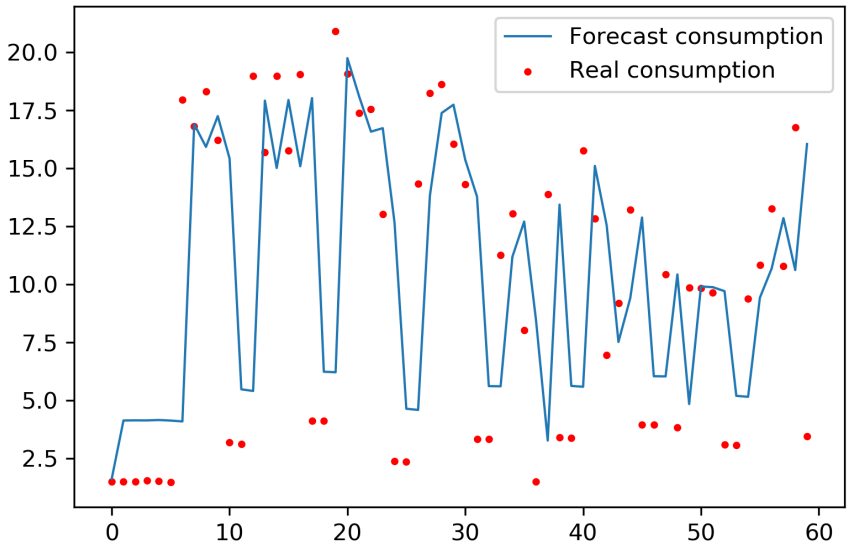

FIGURE 3.19: Detailed daily consumption forecast in office environment

3.6.2 Comparison of results with additional Data Mining methods

The results obtained with the LSTM neural networks were compared with other data mining techniques. These comparative tests were carried out using WEKA, an open-source machine learning software. WEKA is widely used for teaching, research, and industrial applications and it contains a plenty of built-in tools for standard machine learning tasks. [53]

A performance evaluation of different data mining techniques was carried out. In particular, the following functions were used:

- *Random Forest*: Class for constructing a forest of random trees. Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. [54]

- *Random Commettee*: Class for building an ensemble of randomizable base classifiers. Each base classifiers is built using a different random number seed (but based one the same data). The final prediction is a straight average of the predictions generated by the individual base classifiers.

- *M5P*: Class that implements base routines for generating M5 Model trees and rules. [55]

- *Linear Regression*: Class for using linear regression for prediction. Uses the Akaike criterion for model selection, and is able to deal with weighted instances.

The WEKA test-schema used for the performance evaluation is depicted in fig. 3.20. Training-set and Test-set used for comparative tests are the same used in the experiments of LSTM networks.

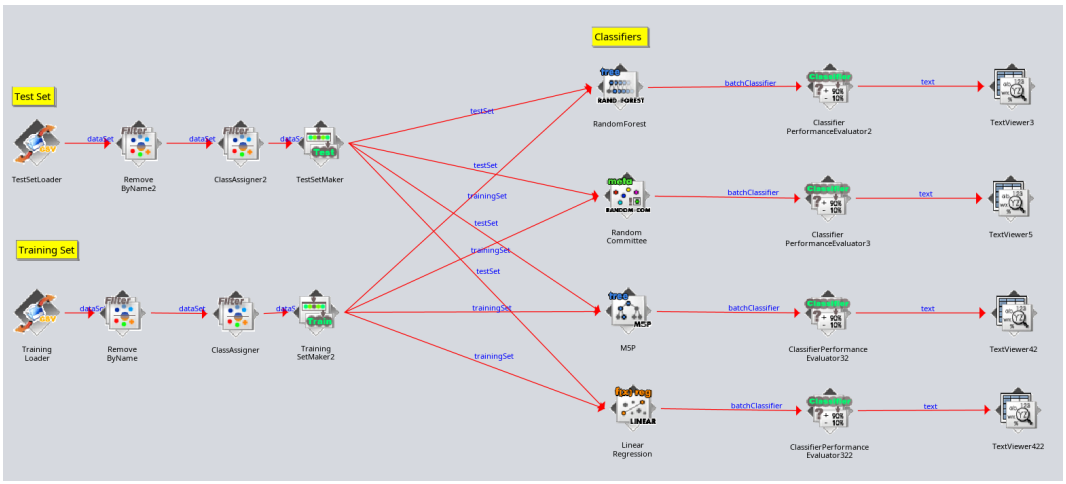

FIGURE 3.20: Data Mining comparison with other techniques

For the evaluation of the used machine learning algorithms, several different performance measures are available. To evaluate performances of analyzed methods we used Root Mean Squared Error value.

The RMSE is the average amount of error given on the test-set in the units of the output variable. This measure is useful to get an idea of the amount a specified prediction may be wrong on average.
Table 3.1 shows the comparative results for both the datasets used and the different time horizons examined in the previous sections.

Method	**Environment and perdiod**		
	Home Montly cons.	**Home Annual cons.**	**Office Daily cons.**
LSTM	*0.744*	*0.640*	*3.214*
RandomForest	*1.483*	*1.056*	*6.159*
RandomCommittee	*1.791*	*1.179*	*6.678*
M5P	*0.974*	*1.093*	*5.174*
Linear Regression	*1.137*	*1.019*	*4.720*

TABLE 3.1: Forecast results comparison of RMSE value

The RMSE values using LSTM networks are lower than the RMSE values obtained using the other methods. The forecast values of energy consumption using the LSTM networks are therefore on average closer to the real consumption values. Better results are achieved using the dataset relating to the home environment, the errors are on average lower than those obtained using the dataset relating to the office environment.

3.7 Chapter summary

In this chapter, after an introduction about Data Mining and Machine Learning techniques, the work was focused on Neural Networks. The basic architecture of a Neural Network was analyzed, which can be divided into Single-layer and Multi-layer. Both architectures have been studied and described. The main applications of neural networks and the different types of neural networks used in the different application domains of machine learning were then introduced.
The use of Recurrent Neural Networks and in particular of Long Short-Term Memory networks applied to the problem of time-series forecasting and the prediction was analyzed in detail. The use of LSTM networks was described in detail to predict electric consumption in home and office environments is provided.

Chapter 4

Cognitive IoT

In this chapter, the proposed Cognitive Smart Object will be described. The application of the Cognitive Smart Object implemented using a smart device for the control of air conditioners used to acquire information on the home environment and manage the thermal comfort of the home environment will be described and analyzed. A dataset compliant with the ASHRAE-55 standard (American Society of Heating, Refrigerating and Air-Conditioning Engineers - Thermal Environmental Conditions for Human Occupancy) was used, which provides the operating ranges to ensure the right levels of comfort.
The result of the use of the neural network was used to suggest to the user a set point based on the user's habits and the management of the climatic comfort typically used.
The Continuous Learning mechanism will be described, which uses user feedback to shape the neural network and obtain a neural network that follows user behaviours that deviate from behaviour compliant with the ASHRAE standard. Finally, an analysis of scalability will be shown as the number of models used varies with results in terms of loading time and RAM occupation.

4.1 Cognitive IoT definition

Cognitive Internet of Things (CIoT) is a novel paradigm, where physical and virtual things are interconnected and act as agents, with minimum human intervention. [56] [57]
The things interact following a context-aware perception-action cycle, use the methodology of understanding-by-building to learn from the physical environment, store

the learned semantic and/or knowledge in kinds of databases, and adapt themselves to changes or uncertainties via resource-efficient decision-making mechanisms, with two main goals:

- bridging the physical world (with objects, resources, etc) and the social world (with human demand, human behaviour, etc) together to create an intelligent physical-cyber-social system;
- enabling smart resource allocation, automatic network operation, and intelligent service provisioning.

A CIoT is an IoT with cognitive and cooperative mechanisms which are integrated to improve performance and achieve intelligence. [58]
A cognitive object can learn the context of use, analyze the recognized knowledge, make intelligent decisions, and perform adaptive actions, which aim to maximize system performance. In the cognitive process, the multi-domain cooperation can increase system capacity and machine learning can enhance the intelligence for future. In [59] the authors present a framework for the virtualization of real-world objects and the cognitive management of their virtual part. The framework consists of three levels of functionality and each level includes cognitive entities that provide the means for self-management and learning, allowing for smart, flexible applications and objects.

4.2 Thermal comfort definition

Thermal comfort is a situation in which a person is satisfied with the indoor conditions of the building. The levels of thermal comfort determine the heat exchanges between the body and the environment. Thermal comfort can differ from person to person and depends on the type of clothes and multiple parameters such as temperature or CO_2 levels.
A range of indices has been developed, tested and implemented to assess the quality of the indoor thermal environment. Among these models, the Predicted Mean Vote (PVM), the adaptive thermal comfort models and the Standard Effective Temperature (SET) are the most widely used. These models are now included in many

international standards such as ASHRAE 55–2017, EN 16798–1: 2019 and ISO 7730: 2005.

PMV and adaptive thermal comfort are aggregate models that predict how a group of people would perceive their thermal environment in terms of given environmental and personal parameters.

The goal of heating and cooling systems in buildings is to provide comfortable indoor environmental conditions for occupants.[60] Establishing the right levels of comfort in the environment is complex. It is possible to use qualitative indices to establish if an environment is thermally uncomfortable. However, the calculation of these indices turns out to be a complex task even for engineers and architects with extensive work experience in the construction sector.

4.3 Cognitive IoT smart objects

The use of smart objects in a Smart Home Environment is typically characterized by devices with limited HW/SW resources. The basic idea for increasing the computational capabilities of smart home devices is to associate a "hidden cognitive object" to the various real devices.

The cognitive IoT device proposed can be considered as a standard IoT device with communication capabilities and a computational core capable of performing simple analysis through an IoT. A Cognitive Smart Object can be seen as the union of a *smart device* and a *hidden cognitive object*.

As depicted in fig.4.1, a real object can be characterized by logical modules:

- *Command listener*
- *Command sender*
- *Communicator*

A hidden cognitive object uses an Artificial Neural Network (ANN) for:

- Suggested action prediction
- Anomaly detection

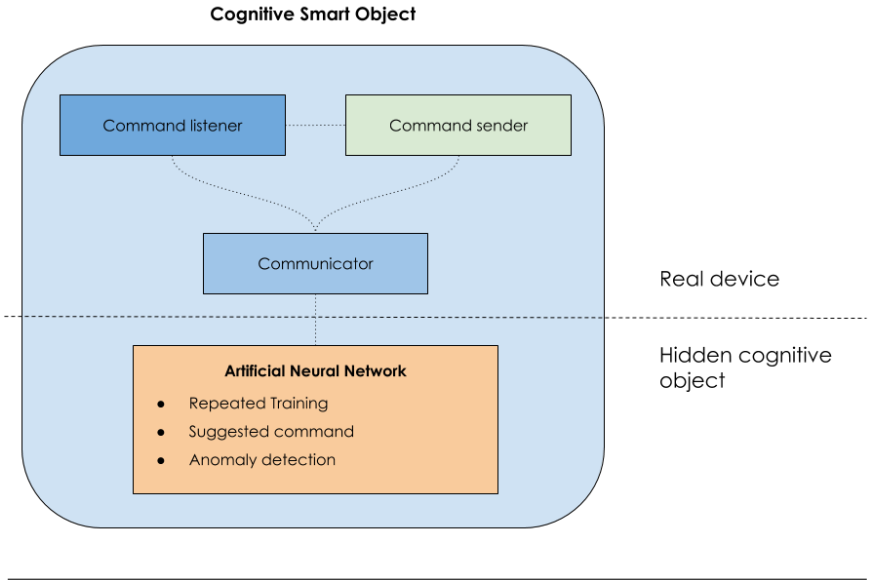

FIGURE 4.1: Cognitive Smart Object schema

Results of the ANN model is used as a suggested command to the user or anomaly detection in user-interaction with the system.

4.3.1 Cognitive IoT smart object integration for HVAC control

The use of devices by users is the basis of the learning process of the neural network used in the hidden cognitive object. The training and validation phases of the model and the consequent accuracy of the results are incremental and is driven by user interactions with the system.

Whenever the user interacts with a smart object, its hidden cognitive object repeats the learning process. The results of the learning process are used to provide a suggested action to the user or notify the user an anomaly detection.

Scheduled activities of ANN learning process can be used to analyze new incoming smart devices data, in this case, the anomaly detection and suggested action system is automated.

Thermal comfort is one of the basic aspects of indoor environmental quality and it is related to occupant satisfaction and energy use in buildings. [61]

The proposed solution has been implemented and tested using a smart device for HVAC control and the associated cognitive smart object. We are using a smart device to get information about the home environment and manage thermal comfort. The HVAC learning process with a restricted number of an analyzed item performs

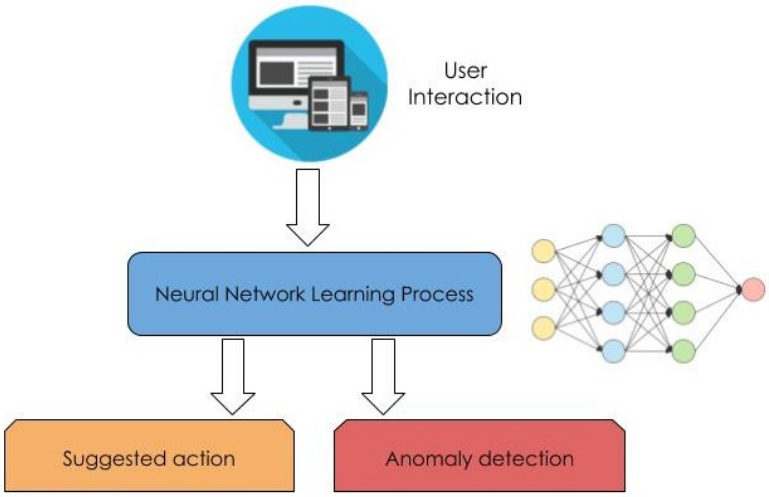

FIGURE 4.2: Cognitive Smart Object data flow

with low accuracy. For the low accuracy of the model for the beginning iterations, we decide to use a pre-trained ANN with a basic dataset. This dataset define the behaviour of the model in the early stages, user interactions and repeated learning stages customize the model with respect to the user's habits. In our testbed of comfort temperature control, the model learns from the temperature setpoints chosen by the user. This information is used to present to the user suggested actions or highlight any unusual situations.

In [62] the authors proposed a non-intrusive data-driven assisted method to evaluate the energy efficiency of the use of Residential HVAC system.

In [63] the authors present three data-driven control algorithms based on machine learning techniques and the use of a multi-objective optimization problem where energy efficiency and comfort parameters are maximized simultaneously.

A deep reinforcement learning technique is used in [64] to learn the effective strategy for operating the building HVAC systems. The authors' experiments demonstrate the energy consumption reduction with maintaining the room temperature in the user's desired range.

Neural Network model for HVAC control

The dataset used for the learning process contains over 2600 items related to data gathered using a smart device for HVAC control in the home environment. The

attribute information of the dataset are:

- *timestamp*: date in format dd/mm/yyyy hh:mm:ss
- *season*: period of the year (winter, spring, summer, autumn)
- *mode*: HVAC mode (hot/cold)
- *thermal comfort*: comfort rate from 1 (very uncomfortable) to 6 (very comfortable)
- *relative humidity*: current relative humidity
- *air temperature*: current room temperature
- *operative temperature*: chosen temperature setpoint

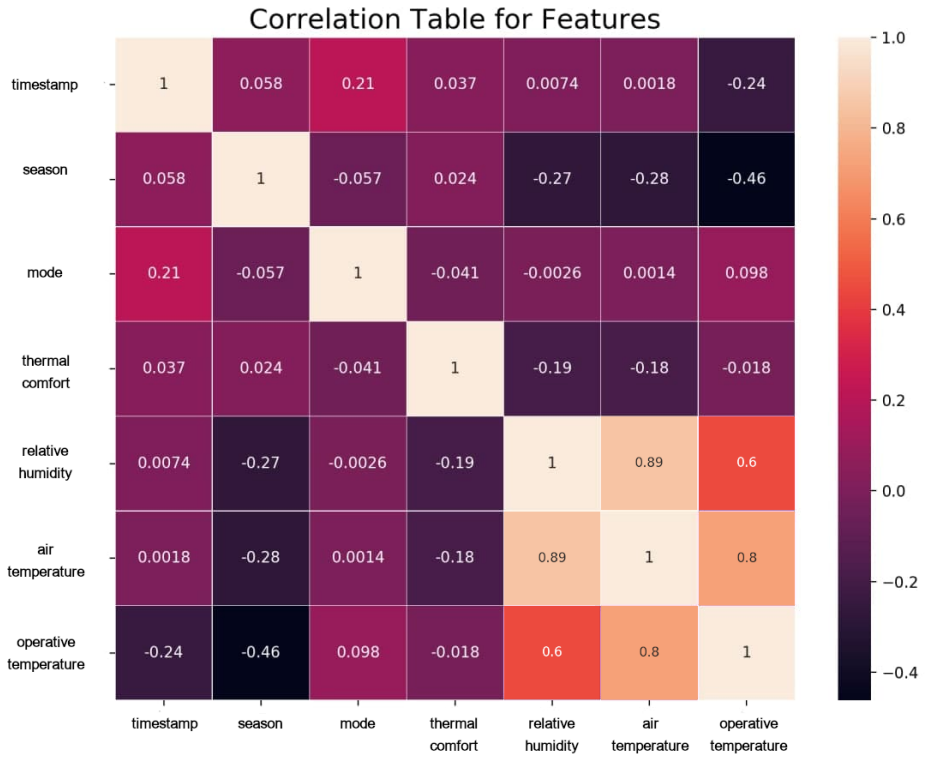

FIGURE 4.3: Correlation table for the dataset features

The dataset is compliant with the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) Standard-55 (*Thermal Environmental Conditions for Human Occupancy*) [65] [66]. This standard provides the ranges of indoor

environmental conditions to achieve acceptable thermal comfort for occupants of buildings. In figure 4.3 it is depicted the correlation table for the dataset features, each cell in the table shows the correlation between the two features. The maximum correlation coefficient value is between temperature and mode values. Other high correlation values are between setpoint and temperature and between setpoint and mode. The information extracted from the timestamp (month, day, hour) has a low correlation coefficient with the other features.

A Sequential model was used with an input layer, 2 densely-connected ANN hidden layer and a single output layer is shown in fig.4.4.

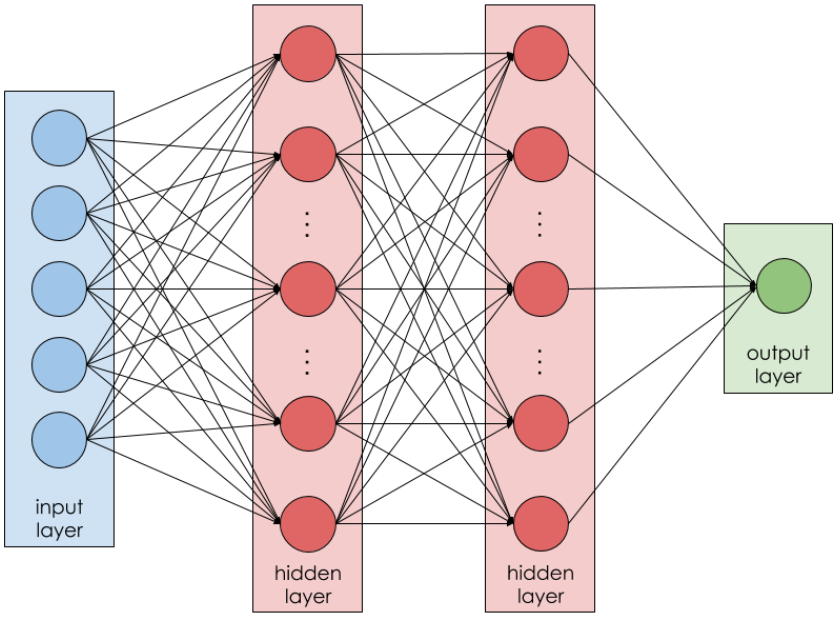

FIGURE 4.4: ANN structure

The used model uses the following parameters:

- *Optimizer*: *Root Mean Square Propagation*, an optimizer that implements the RMSprop algorithm
- *Loss function*: *Mean Absolute Error* that computes the mean of the absolute difference between target values and predicted values

The training phase of the model uses two further parameters:

- *batch size*: number of samples per gradient update equal to 50
- *epochs*: number of iteration to train the model equal to 32

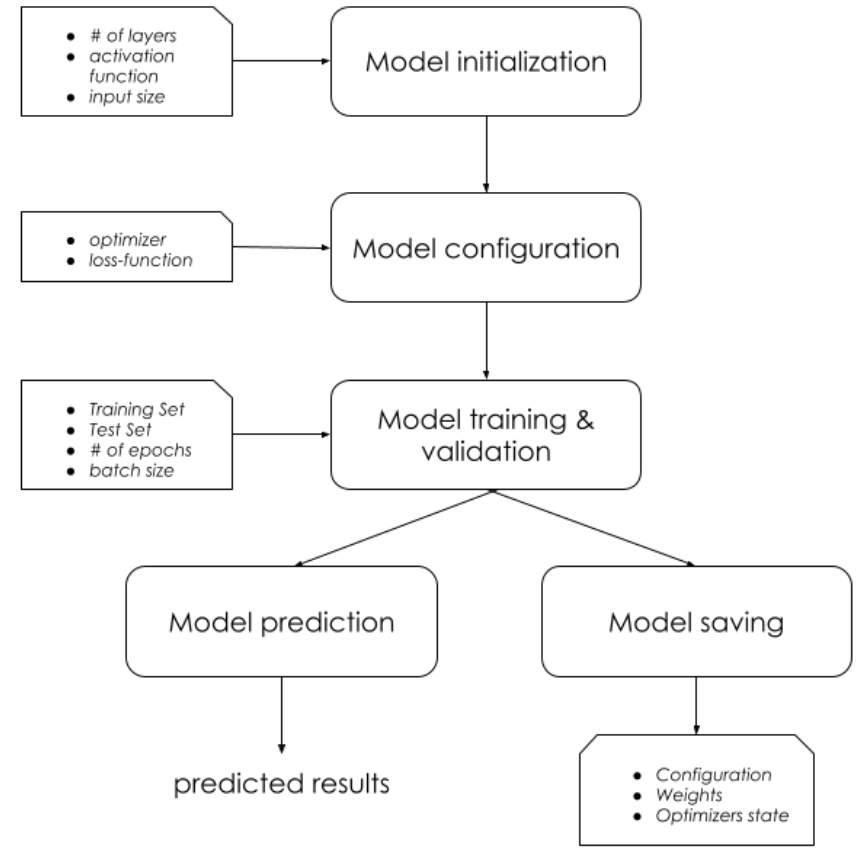

FIGURE 4.5: Neural Network functional steps

In fig. 4.5 Neural Network functional steps is shown. The description of every single phase is given below:

- *Model Initialization*: definition of model structure with layers specification and parameters as activation function and input size
- *Model Configuration*: configure the model for training with optimizer and loss-function
- *Model Training & Validation*: trains and evaluate the model for a fixed number of epochs (iterations on a dataset)
- *Model Prediction*: generated output prediction for the input samples
- *Model Saving*: save the model on file with the model configuration (topology), model weights and model optimizers state. A saved model can be loaded and re-instantiated without of the code used for model definition or training

The neural network model results is used for anomaly detection and suggested action. An example of a suggested action is described below, starting from a command the result of the neural network analysis is shown. An user command is characterized as follow

$$timestamp \mid mode \mid setpoint$$

The result of ANN analysis is an estimated setpoint and this result is highly connected with the mode and current temperature acquired from the smart device.

$$command + current\ temp \rightarrow suggested\ setpoint$$

Continuous Learning model

The cognitive part associated with the smart object can be used to suggest the user the recommended setpoint learned from his habits. A command characterized by a setpoint temperature outside the common operating ranges or not corresponding to the detected temperature can generate an alert message to the user.
As depicted in fig. 4.6 starting from a user command there is a validation check on the command. If the system identifies an anomaly send a suggested setpoint to the user otherwise the command is executed. The negative feedbacks are stored for continuous learning of user habits. This indicates a neural network shaping with adjustments of the initial structure of the neural network for learning user habits that deviate from behaviours that comply with the ASHRAE standard. There are some operations that the system carries out once for each user, specifically the construction and loading of the pre-trained model (ASHRAE standard-compliant). These initial operations are followed by actions performed at runtime at each user interaction with the prediction and validation steps that are performed before the execution of a command. The user feedback storage is used for triggering the model re-training operation whenever 50 user feedbacks are collected. In this way, the model works with continuous learning and will adapt autonomously to the new data. [67] The accuracy of the model will automatically improve and overall the model will perform better and more tailored to the user's habits.
Fig. 4.7 describe the data flow used to obtain a continuous learning model.

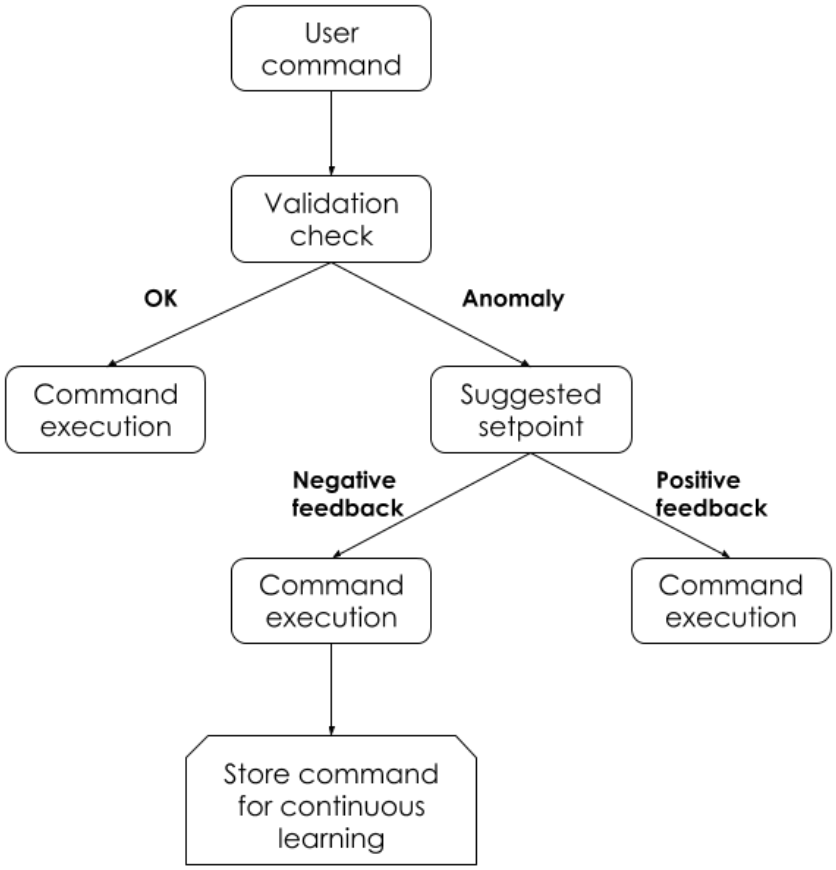

FIGURE 4.6: Cognitive object data flow

Analysis of results

To obtain a qualitative analysis of the results, user feedback was evaluated for the suggestions provided by the neural network.

In fig. 4.8 is depicted the feedback of the user regarding the suggested action predicted of the model. The highest percentage of feedback, corresponding to approximately 55%, is given by the "no-change" action, the percentage of "cool" and "warm" corrective actions are about 20%.

Analyzing fig. 4.9 it is possible to see how as the user interactions with the system increase the learning of the neural network improves and the percentage of feedback for the "no-change" action increases compared to the "cool" and "warm" corrective actions.

The pre-trained ASHRAE model and the successive learning phases generates a neural network that follows the user's habits. Corrective actions enables the model

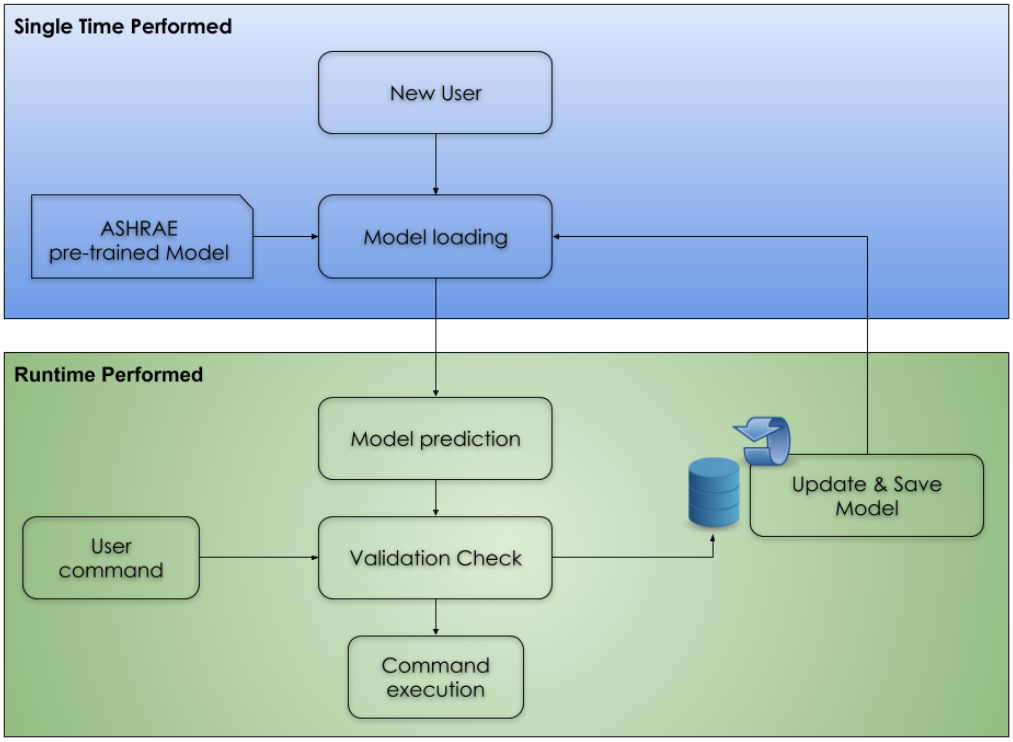

FIGURE 4.7: Continuous learning schema

to adapt to user habits that differs from a typical behaviour defined by the initial dataset compliant with the ASHRAE standard.

The phases of construction, training and prediction of the model were analyzed in terms of execution time and RAM occupation.
Fig. 4.10 shows the training and configuration times of the model. In particular, it is possible to remark how the model initialization and configuration time is constant and equal to about 0.05 seconds while the model training and validation time is on average equal to 32 seconds.

In Table 4.1 a model data summary is provided. These pieces of information are about an initial model trained and validated with a dataset that contains ASHRAE-compliant items. The average training time of the model on a dataset of approximately 2600 item is equal to 32 seconds, the file size on disk after model saving is 25.8 kB and the RAM occupancy of the single model is equal to 4.96MB. The prediction phase duration is about 0.07 seconds and the RAM occupancy in this step is 0.26 MB.

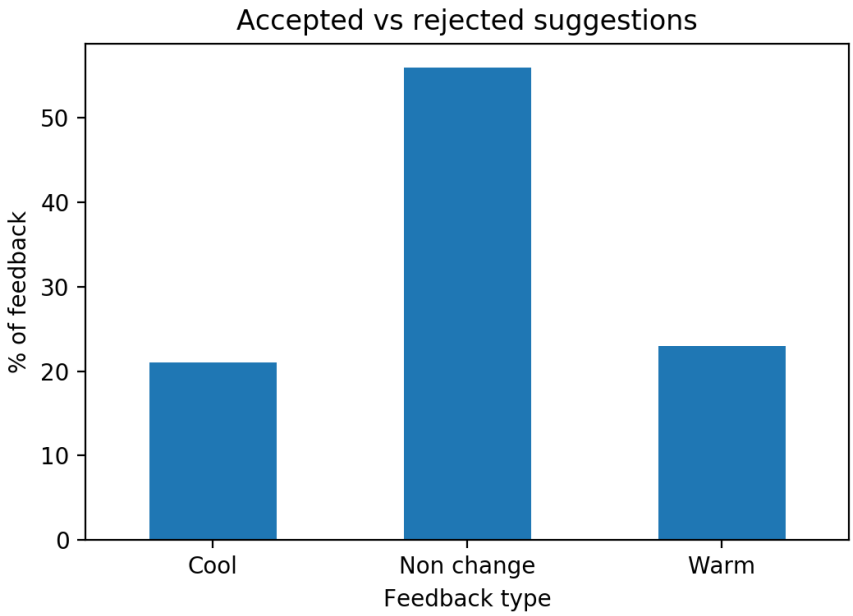

FIGURE 4.8: User feedback to the suggested action

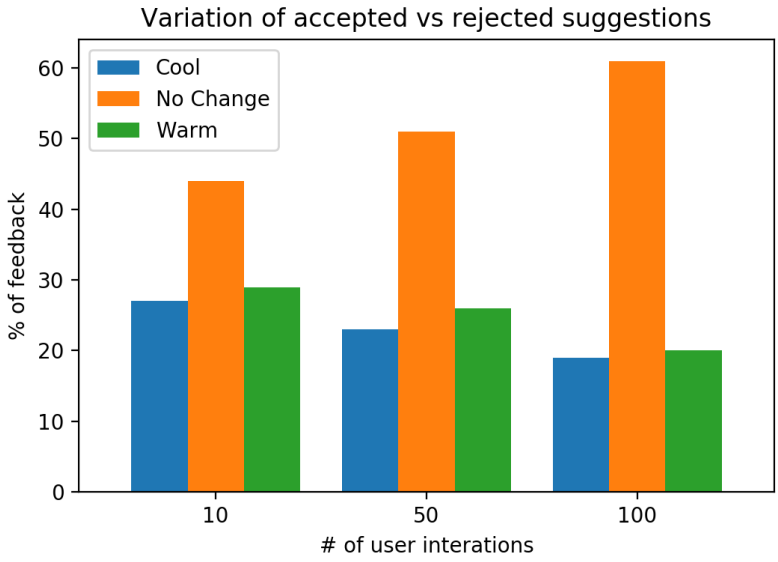

FIGURE 4.9: Accepted and Rejected suggestions for the number of user interaction

Ad-hoc tests were carried out to analyze the loading times and memory occupation in case of the use of multiple models (associated with different users). The RAM memory occupation is compliant with a cloud-based platform and the execution times are not high for a typically high computation rate task.

Table 4.2 shows test results for an upload of 1,10,30,50,100 pre-trained models with

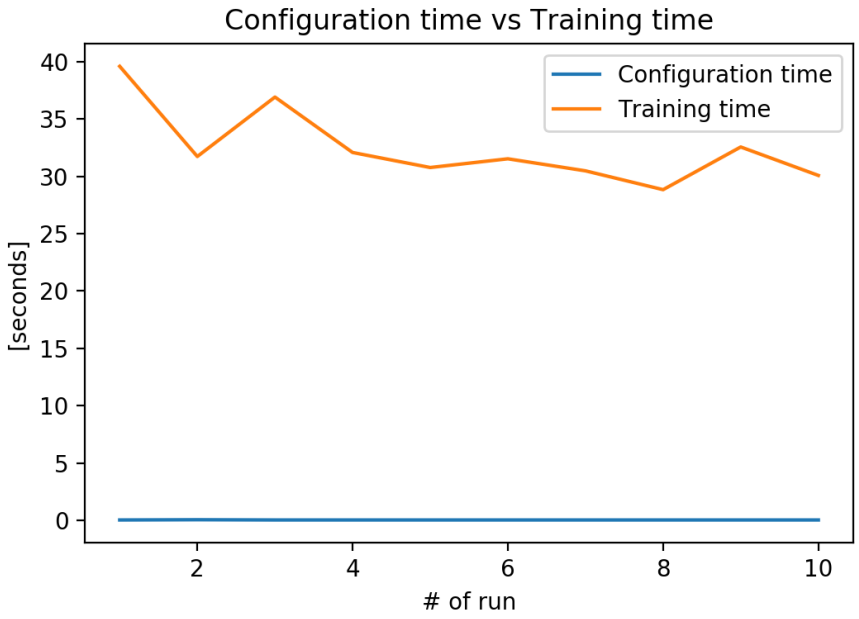

FIGURE 4.10: Compiling and Training time of the NN model

Model analysis	
Configuration time	0.05 s
Avg Training time	32.44 s
Loading time	0.35 s
Model file size	25.8 kB
Single model RAM occupancy	4.96 MB
Model prediction time	0.07 s
Model prediction RAM occupancy	0.26 MB

TABLE 4.1: Model data summary

ASHRAE-compliant dataset.

# of models	loading time	occupied RAM
1	0.05 s	4.96 MB
10	3.43 s	16.30 MB
30	10.18 s	86.42 MB
50	18.30 s	158.48 MB
100	33.27 s	496.46 MB

TABLE 4.2: Loading time and occupied RAM for multiple models

The analysis of the scalability with increasing users and Neural Network (NN) models used, in terms of loading time and RAM memory usage is shown in fig. 4.11 and fig. 4.12. Loading time and RAM occupation are linear as the number of models used changes.

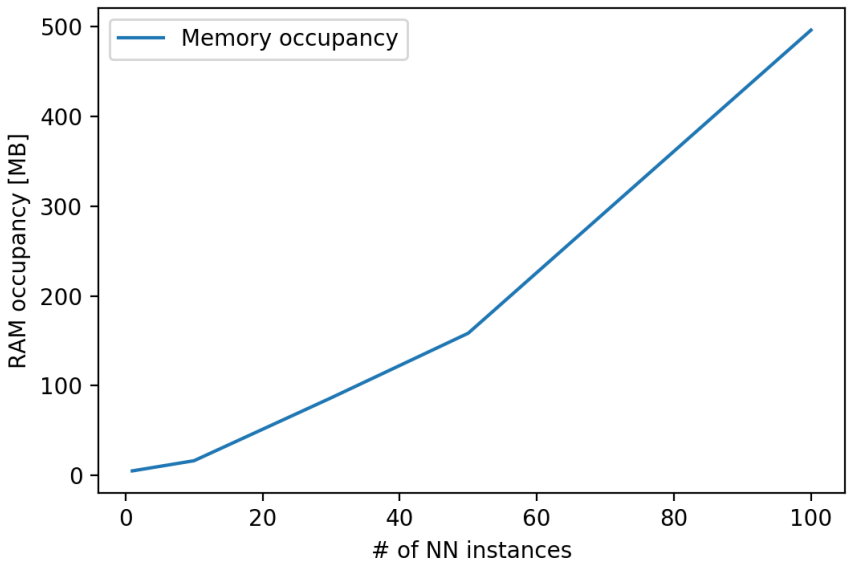

FIGURE 4.11: Memory occupancy for different NN model number

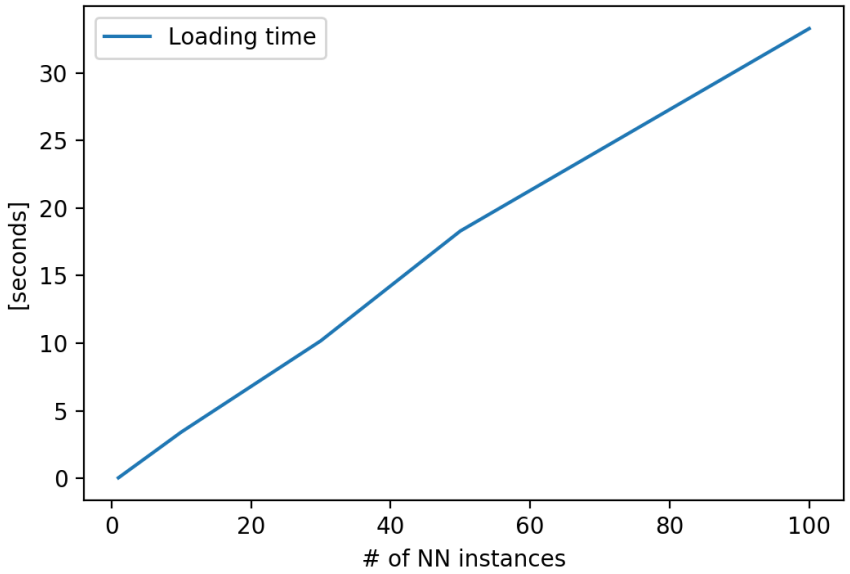

FIGURE 4.12: Loading time for different NN model number

4.4 Chapter summary

In this chapter, the use of the Cognitive Smart Object has been described. The Cognitive smart object was used to support the management of thermal comfort in Smart Home context.

It was explained in detail how a cognitive smart object is defined by the association of smart devices and a hidden cognitive object. In this way, it is possible to obtain the result of increasing the HW/SW resources of the smart device.

Neural Networks were used for suggested action prediction and anomaly detection operations. Cognitive Smart Object integration was described for HVAC control.

Details on the Neural Network model used, a description of the Neural Network functional steps and the application of the continuous learning concept have been provided in the chapter. The use of Neural Network models in terms of RAM occupation and loading time was analyzed to carry out a scalability analysis of the proposed solution.

Chapter 5

Energy efficiency of devices in the Smart Home environment

This chapter will be present the testbed used to analyze the assisted comfort solution created by applying the concepts introduced with the cognitive smart object.
The proposed solution aims to find the right trade-off between thermal comfort management and a possible reduction in energy consumption in the context of Smart Home Management. The novel contributions that represent an alternative to traditional control methods in the smart home environment are the following:

- IoT architecture without the usage of intermediate devices
- Proposal of an adaptive solution compliant, thanks to object virtualization, with devices with very low computational capabilities
- Initial dataset standard-compliant for thermal comfort and continuous learning process to shape the NN and obtain results that follows user behaviours

The research focus is based on the analysis of user habits using the assisted comfort system. The assisted comfort system has changed the user's habits in terms of thermal comfort, producing them closer to standard-compliant thermal comfort levels. It will be shown how the use of the assisted comfort system produces a reduction in energy consumption while maintaining an acceptable level of comfort.

5.1 Testbed description

In the context of Smart Home Management, it is desirable to find the right trade-off between the control of thermal comfort and a possible reduction of energy consumption. [68]
By applying the concepts introduced with the use of the cognitive smart object described in the previous chapter, there are two main goals to be achieved:

- adaptation of the NN to the user's comfort habits
- user education in the concept of "standard" thermal comfort

To analyze the effects in terms of energy efficiency of the proposed testbed has been implemented and tested using a smart device for HVAC control, the associated cognitive smart object and a smart meter. We use the HVAC controller to get information about the home environment and manage thermal comfort and the smart meter to acquire the energy consumption related to the management of thermal comfort.
A group of 12 test users used the test configuration for an observation period of approximately 60 days. Each user manages the thermal comfort within their home environment independently.

Testbed info	
# of test users	*12*
avg of daily interactions	*8*
min daily interactions	*3*
max daily interactions	*11*
avg total feedbacks	*253*
total feedbacks	*3087*
avg days of observation	*60*

TABLE 5.1: Testbed data summary

Table 5.1 reports some data that characterize the testbed such as the number of test users, the number of observation days and other info related to the feedbacks and interactions of the various users with the "assisted-comfort" system.

The proposed and implemented architecture for the testbed is shown in fig. 5.1. The cloud-based platform provides a smart device and user interaction management.
The main entities of the core-layer of the proposed assisted-comfort solution are:

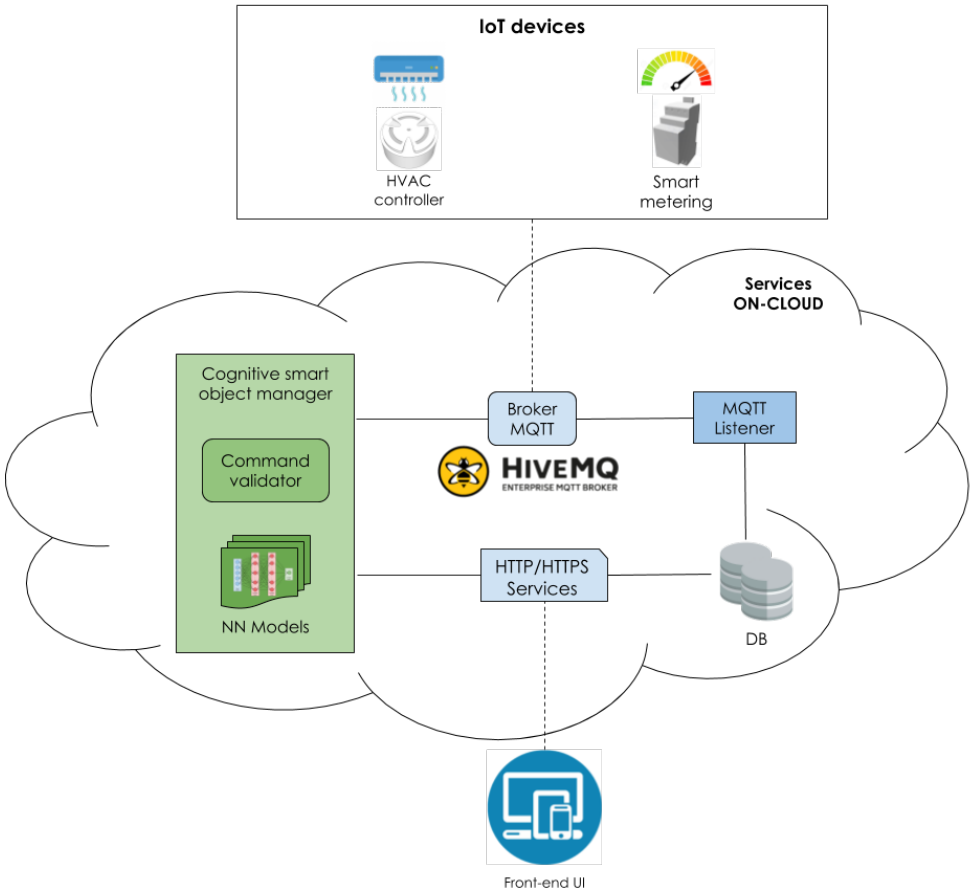

FIGURE 5.1: Testbed architecture schema

- *DB*: MySQL database for data storage
- *HTTP/HTTPS Services*: REST web services used to access the features offered by the platform
- *MQTT Broker*: MQTT broker required for the MQTT protocol
- *MQTT Listener*: SW module used for collecting and sending the data via the MQTT protocol
- *Cognitive smart object manager*: SW module that manages instances of cognitive smart objects

The smart devices used for the testbed are an HVAC controller and a smart meter. sers interact with the assisted-comfort system using a web-app and a mobile-app for the use and management of the smart devices used.

5.2 Feedback data flow & parameters

The assisted-comfort system is based on the analysis of user interactions with the smart HVAC controller device used to manage thermal comfort in the home environment.
The data-flow of the information used is depicted in fig. 5.2. The main steps are:

- user command
- validation and prediction of the suggested setpoint
- user feedback analysis for continuous learning

The module called Command validator performs initialization or loading of the pre-trained NN model used for the prediction phase. The initial NN model is the same for all users and is customized during the re-training phases. The feedback analysis phase, in addition to sending the command to the smart device, uses a feedback collector module to store the feedback sequences used for continuous learning.
The NN re-training steps are activated using the re-training threshold parameter. This parameter indicates the re-shaping speed of the neural network with respect to the user habits that deviate from the defined standard behaviour. The NN re-training operation is the most costly operation in terms of computational resources and processing time, as analyzed in the previous chapter and is approximately 32 seconds. To establish the right trade-off between a rapid re-shaping and usage of computational resources, tests were carried out on varying the re-training threshold parameter.

By choosing a number of about 200 user interactions with the system during the observation period, an experimental analysis was carried out to choose the right re-training threshold.
Fig. 5.3 shows the number of re-training operations as the re-training threshold parameter changes. The re-training threshold used for the testbed is 30. Considering an average number of interactions in the period equal to approximately 200 and with a re-training threshold equal to 30, a re-training operation takes place approximately every 6.77 days. In this way, the system collects user feedback about a week of use

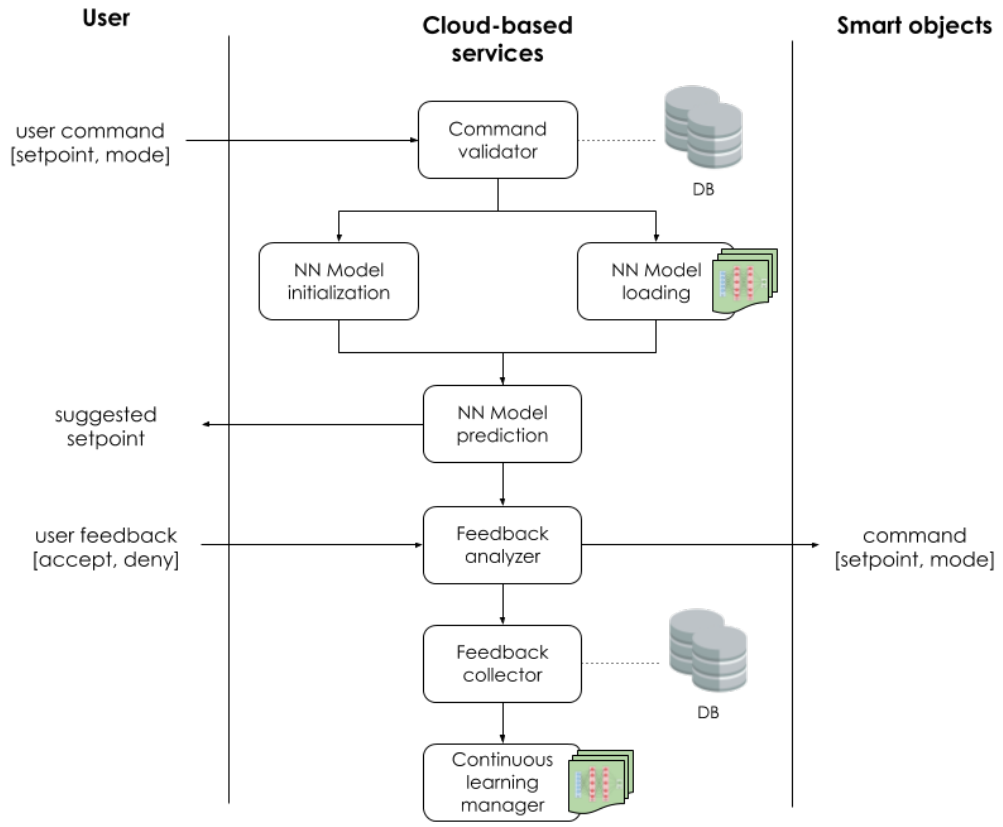

FIGURE 5.2: Feedback data flow

of the system before carrying out the re-training operation and the tailoring of the NN with respect to user interactions.

From the investigation of user interactions with the system, it emerged that the average number of interactions with the system was equal approximately to 253 and with the use of the re-training threshold parameter set at 30, on average during the observation period approximately 8 re-training operations which correspond about to one re-training operation per week.

5.3 Individual comfort vs Assisted comfort

By analyzing the change in user habits using the NN feedback system, is possible to observe as an adaptation of the NN to the user's habits in terms of thermal comfort pushes the latter to "approach" with continuous use at standard-compliant thermal comfort levels. [69]

FIGURE 5.3: Number of re-training operations

On average, in the observation period, the reduction in energy consumption is approximately 8%, while in the case considered in fig. 5.4 it is around 12% in the observation period.

The fig. 5.4 shows the results obtained by analyzing the user with the greatest number of interactions and feedbacks.

For our test, three contiguous time windows of 5 weeks each were analyzed. In the "individual comfort" period the user did not use any type of suggestion and managed the thermal comfort autonomously.

The *"first assisted-comfort period"* is a sort of training period for the NN. This period is followed by *"second assisted comfort period"* that we can define as validation period where the improvement of the results provided by the NN is better and the percentage of suggestions accepted by the user increases, thanks to the application of the continuous learning.

The improvement over time in the percentage of suggestions accepted in the observation time is shown in fig. 5.5. The application of the concept of continuous learning provides to obtain an adjustment of the NN to the user's comfort habits. The repeated re-training operations of the NN allows the tailoring of the NN and results that follow the user habits.

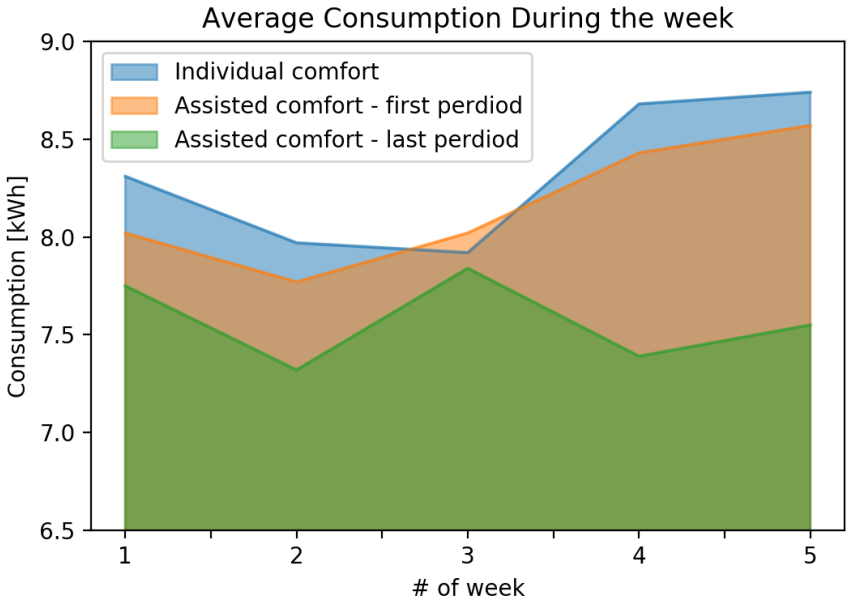

FIGURE 5.4: Average consumption during the week

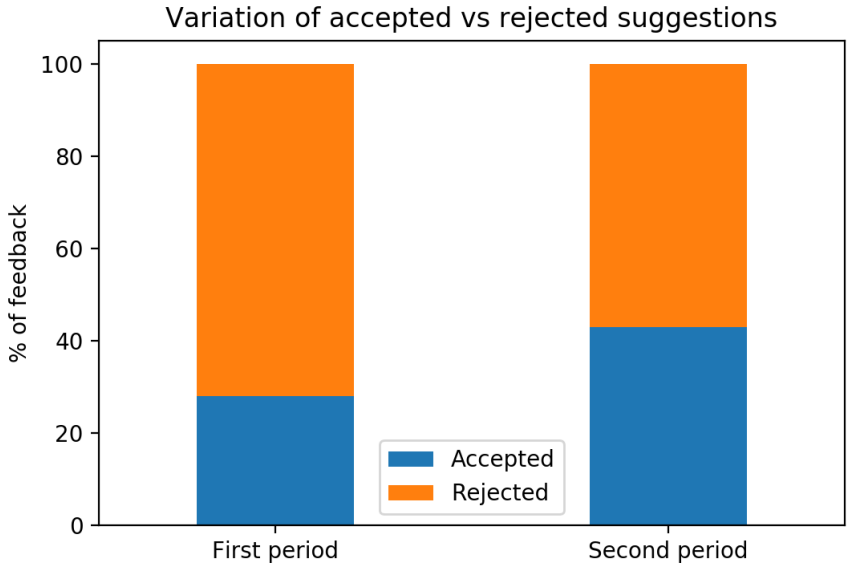

FIGURE 5.5: Accepted vs rejected NN suggestions

5.4 Divergent users analysis

The data collected during the testbed showed that there are users called "divergent-users" who, using the assisted comfort system, have negative feedbacks that do not decrease over time. This is because their behaviours have not had any similarity to what has been defined as a standard-compliant behaviour, as happened in 90% of the cases analyzed.

These users have a higher average negative feedback rate than all other test users. Analyzing energy consumption, it is possible to identify two different types of divergent users:

- users characterized by very high consumption *(a)*
- users characterized by very low consumption *(b)*

It is possible to define these users as outliers for which the concept of standard thermal comfort is unmatched.

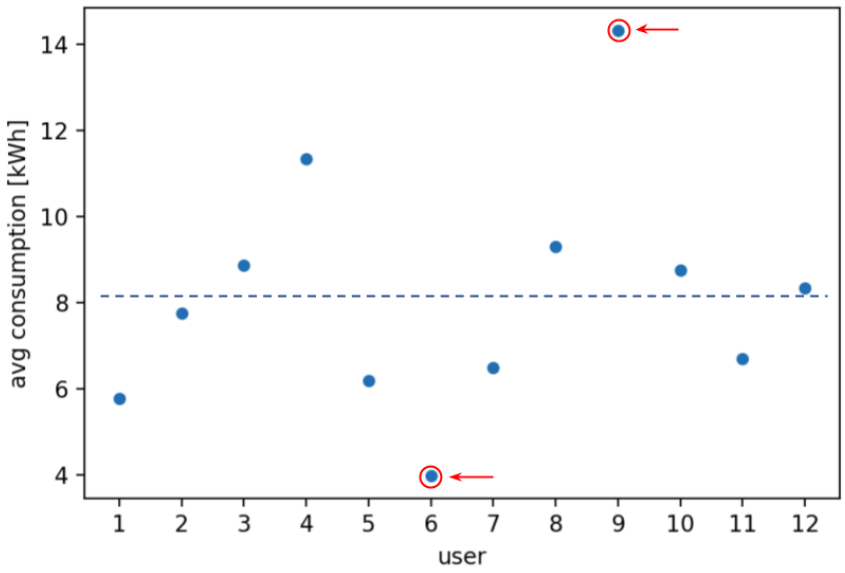

FIGURE 5.6: Energy consumption and divergent user

Some "divergent users" concerning their habits sacrifice comfort levels to the exclusive advantage of low consumption, others who do not take into account high consumption by exclusively preferring personal comfort.

Fig. 5.6 shows the average consumption of test users during the observation period. Consumption of divergent users is highlighted in red. The average daily consumption registered during the observation period is about 8.15 kWh and is shown by the dashed line.

In the case of divergent users (a), the average daily consumption is approximately 14.32 kWh. In the case of a divergent user (b) it is about 3.98 kWh. If the outliers represented by diverging users are not taken into consideration, the average daily consumption in the observation period is approximately 7.95 kWh. The average

consumption of divergent users differs considerably from what can be considered standard consumption profiles.

5.5 Chapter summary

This chapter describes the testbed used to analyze the results of an assisted-comfort solution based on the use of Cognitive Smart Objects. Details were provided on the IoT architecture used for the tests, the observation period and the number of users involved.

The proposed solution is based on the use of feedback and the analysis of user interactions with the system for HVAC control. The goal was to find the right trade-off between comfort and energy/economic savings. It was also shown how the use of the assisted-comfort system typically leads the user to adjust their habits and how the concept of continuous learning applied allows Neural Networks to learn and adapt to the comfort habits that the user prefers.

Chapter 6

Conclusions

During this period of PhD, I have focused my researches on the study of IoT technologies and protocols in the field of Energy Management in a Smart Home context. Initially, the research activities were principally focused on the study and analysis of the protocols and technologies most used in the IoT field with a particular focus on the MQTT protocol and its possible applications. It has been defined as a reference IoT architecture also used in other application contexts such as e-Health and Internet of Vehicles (IoV).

After, the application of Neural Networks for the analysis and prediction of consumption was studied. For the time-series forecasting and prediction task, the use of LSTMs was a natural choice. The LSTM networks usage to predict electric consumption in a different environment was compared with other data mining techniques and the qualitatively better results obtained using LSTM networks were shown.

I have invested a significant amount of time studying the IoT solutions in the smart home environment. In this application context, smart objects are typically characterized by limited resources. To increase the computational capabilities of these smart devices, a hidden cognitive object was used to associate with the various devices to perform, for example, anomaly detection and suggested action prediction using pre-trained NN and continuous learning. The proposed Cognitive Smart Object can be considered as the combination of a smart device and a hidden cognitive object.

The introduced Cognitive Smart Object has been used in applications used for thermal comfort control. The goal was to find the right compromise between the management of thermal comfort and a possible reduction in energy consumption. Using the concepts introduced, two considerable effects were obtained: modification of the

NN results to the user's comfort habits and user education in the concept of standard thermal comfort.
The Assisted comfort solution, through the use of cognitive smart objects, guide to a reduction in energy consumption while maintaining comfort levels that satisfy the user's needs.

6.1 Future works

In the next period, I am going to extend the study of Energy Management in a Smart Home context. I will try to use the proposed Cognitive Smart Object to support other key aspects of Smart Home Applications such as smart metering, lighting, fault detection and air quality control.
I also want to deeply investigate the combination of consumption prediction techniques and concepts related to energy saving and comfort. I will expect that it is possible to take advantage of the combined consumption data and feedback on suggestions to classify user behaviour.
The possible obtained results could be used to provide users with detailed information on the behaviours that lead to energy waste. I think that the analysis of user behaviour is useful to improve life quality and reduce energy waste and CO_2 emissions.

Chapter 7

Pubblications

- **Cognitive IoT enabled by Layered Architecture and Neural Networks in a Smart Home Environment**
  A. Serianni, F. De Rango, P. Raimondo
  12th Wireless Days Conference (WD 2021)

- **Application Layer Protocols for Internet of Things**
  A. Serianni, F. De Rango
  Book chapter in Advances in Computing, Informatics, Networking and Cybersecurity, Springer-Nature Book

- **Cognitive IoT solution based on Neural Networks for the Smart Home**
  A. Serianni, F. De Rango, P. Raimondo
  IEEE Internet of Things Journal *(Submitted)*

- **AI-powered Multi time-scale Energy forecasting in IoT enabled Smart Home Environment**
  A. Serianni, F. De Rango, P. Raimondo
  IEEE Access *(Submitted)*

- **Fuzzy Inference System design for promoting an Eco-friendly Driving Style in IoV Domain**
  F. De Rango, M. Tropea, A. Serianni, N. Cordeschi
  Vehicular Communications Journal, Elsevier *(Submitted)*

- **A real IoT device deployment for E-Health applications under lightweight communication protocols, activity classifier and Edge data filtering**
  A. F. Santamaria, F. De Rango, A. Serianni, P. Raimondo
  Computer Communications - Volume 128, September 2018, Pages 60-73

- **Drones Support in Precision Agriculture for Fighting Against Parasites**
  G. Potrino, N. Palmieri, V. Antonello, A. Serianni
  2018 26th Telecommunications Forum (TELFOR), 1-4

- **Improving Intelligent Transportation System (ITS) Introducing a FOG Cooperative Strategy**
  P. Raimondo, A. Serianni, N. Palmieri, G. Potrino
  2018 26th Telecommunications Forum (TELFOR), 1-4

- **An energy aware smart station for an UAV fleet in the smart farming application**
  A. Serianni, P. Raimondo, N. Palmieri
  Proc. SPIE 11008, Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping IV, (14 May 2019)

- **Cooperative IoV environment for a decentralized and scalable ITS system**
  A. Serianni, P. Raimondo, G. Potrino
  Proc. SPIE 11009, Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019, (2 May 2019)

- **Drones coordination protocols in the precision agriculture context**
  G. Potrino, A. Serianni, N. Palmieri
  Proc. SPIE 11008, Autonomous Air and Ground Sensing Systems for Agricultural Optimization and Phenotyping IV, (14 May 2019)

- **A real IoT device deployment for E-Health applications under lightweight communication protocols, activity classifier and Edge data filtering**
  A. F. Santamaria, F. De Rango, A. Serianni , P. Raimondo
  Computer Communications - Volume 128, September 2018, Pages 60-73

- **A two stages fuzzy logic approach for Internet of Things (IoT) wearable devices**

  A. F. Santamaria, P. Raimondo, F. De Rango, A. Serianni

  Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium

- **Smart wearable device for health monitoring in the Internet of Things (IoT) domain**

  A. F. Santamaria, A. Serianni, P. Raimondo, F. De Rango, M. Froio

  SCSC '16 Proceedings of the Summer Computer Simulation Conference Article No. 36

- **A New Application for Analyzing Driving Behaviour and Environment Characterization in Transportation Systems based on a Fuzzy Logic Approach**

  P. Fazio, A. F. Santamaria, F. De Rango, M. Tropea, A. Serianni

  Proc. SPIE 9837, Unmanned Systems Technology XVIII, 983707 (May 13, 2016); doi:10.1117/12.2228432

Bibliography

[1] A. Rayes and S. Salam, *Internet of Things From Hype to Reality: The Road to Digitization*. Springer, 2019.

[2] E. Borgia, "The internet of things vision: Key features, applications and open issues", *Computer Communications*, vol. 54, pp. 1–31, 2014.

[3] D. Evans, "The internet of things: How the next evolution of the internet is changing everything", *CISCO white paper*, vol. 1, no. 2011, pp. 1–11, 2011.

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications", *IEEE communications surveys & tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[5] S. Vashi, J. Ram, J. Modi, S. Verma, and C. Prakash, "Internet of things (iot): A vision, architectural elements, and security issues", in *2017 international conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, IEEE, 2017, pp. 492–496.

[6] T. Salman and R. Jain, "Networking protocols and standards for internet of things", *Internet of Things and Data Analytics Handbook*, vol. 2015, pp. 215–238, 2015.

[7] V. Karagiannis, P. Chatzimisios, F. Vazquez-Gallego, and J. Alonso-Zarate, "A survey on application layer protocols for the internet of things", *Transaction on IoT and Cloud computing*, vol. 3, no. 1, pp. 11–17, 2015.

[8] T. Salman and R. Jain, "A survey of protocols and standards for internet of things", *arXiv preprint arXiv:1903.11549*, 2019.

[9] E. Ferro and F. Potorti, "Bluetooth and wi-fi wireless protocols: A survey and a comparison", *IEEE Wireless Communications*, vol. 12, no. 1, pp. 12–26, 2005.

[10] M. Park, "Ieee 802.11 ah: Sub-1-ghz license-exempt operation for the internet of things", *IEEE Communications Magazine*, vol. 53, no. 9, pp. 145–151, 2015.

[11] M. Siekkinen, M. Hiienkari, J. K. Nurminen, and J. Nieminen, "How low energy is bluetooth low energy? comparative measurements with zigbee/802.15. 4", in *2012 IEEE wireless communications and networking conference workshops (WCNCW)*, IEEE, 2012, pp. 232–237.

[12] C. Gomez and J. Paradells, "Wireless home automation networks: A survey of architectures and technologies", *IEEE Communications Magazine*, vol. 48, no. 6, pp. 92–101, 2010.

[13] A. Stanford-Clark and H. L. Truong, "Mqtt for sensor networks (mqtt-sn) protocol specification", *International business machines (IBM) Corporation version*, vol. 1, p. 2, 2013.

[14] D. Locke, "Mq telemetry transport (mqtt) v3. 1 protocol specification", *IBM developerWorks Technical Library*, vol. 15, 2010.

[15] P. Saint-Andre *et al.*, "Extensible messaging and presence protocol (xmpp): Core", 2004.

[16] R. T. Fielding and R. N. Taylor, *Architectural styles and the design of network-based software architectures*. University of California, Irvine Irvine, 2000, vol. 7.

[17] X. Chen, "Constrained application protocol for internet of things", *URL: https://www. cse. wustl. edu/˜ jain/cse574-14/ftp/coap*, 2014.

[18] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes", *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[19] A. Serianni and F. De Rango, "Application layer protocols for internet of things", *Book chapter in Advances in Computing, Informatics,Networking and Cybersecurity, Springer-Nature Book*, 2021.

[20] D. Thangavel, X. Ma, A. Valera, H.-X. Tan, and C. K.-Y. Tan, "Performance evaluation of mqtt and coap via a common middleware", in *2014 IEEE ninth international conference on intelligent sensors, sensor networks and information processing (ISSNIP)*, IEEE, 2014, pp. 1–6.

[21] P. Thota and Y. Kim, "Implementation and comparison of m2m protocols for internet of things", in *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD)*, IEEE, 2016, pp. 43–48.

[22] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing", 2011.

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

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

[25] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog", *Ieee Access*, vol. 7, pp. 150 936–150 948, 2019.

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

[27] H. F. Durrant-Whyte, "Sensor models and multisensor integration", in *Autonomous robot vehicles*, Springer, 1990, pp. 73–89.

[28] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for iot interoperability", in *2015 IEEE International Conference on Mobile Services*, IEEE, 2015, pp. 313–319.

[29] A. F. Santamaria, F. De Rango, A. Serianni, and P. Raimondo, "A real iot device deployment for e-health applications under lightweight communication protocols, activity classifier and edge data filtering", *Computer Communications*, vol. 128, pp. 60–73, 2018.

[30] EclipsePaho. (2020). Eclipse paho java client java client, [Online]. Available: `http://www.eclipse.org/paho/clients/java/`.

[31] HiveMQ. (2020). Hivemq-enterprise mqtt broker, [Online]. Available: `http://www.hivemq.com/`.

[32] E. H. Mamdani, "Application of fuzzy logic to approximate reasoning using linguistic synthesis", *IEEE transactions on computers*, no. 12, pp. 1182–1191, 1977.

[33] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control", *IEEE transactions on systems, man, and cybernetics*, no. 1, pp. 116–132, 1985.

[34] K Nanthini and R. M. Devi, "Adaptive fuzzy c-means for human activity recognition", in *International Conference on Information Communication and Embedded Systems (ICICES2014)*, IEEE, 2014, pp. 1–5.

[35] F. De Rango, M. Tropea, A. Serianni, and N. Cordeschi, "Fuzzy inference system design for promoting an eco-friendly driving style in iov domain", *Vehicular Communications Journal*, Submitted.

[36] M. Van Ly, S. Martin, and M. M. Trivedi, "Driver classification and driving style recognition using inertial sensors", in *2013 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2013, pp. 1040–1045.

[37] M. Bramer, *Principles of data mining*. Springer, vol. 180.

[38] C. C. Aggarwal, *Data mining: the textbook*. Springer, 2015.

[39] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection", *IEEE Communications surveys & tutorials*, vol. 18, no. 2, pp. 1153–1176, 2015.

[40] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial", *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[41] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[42] C. C. Aggarwal *et al.*, *Neural networks and deep learning*. Springer, 2018.

[43] I. A. Basheer and M. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application", *Journal of microbiological methods*, vol. 43, no. 1, pp. 3–31, 2000.

[44] L. R. Medsker and L. Jain, "Recurrent neural networks", *Design and Applications*, vol. 5, 2001.

[45] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[46] C. Olah, "Understanding lstm networks", 2015.

[47] A. Serianni, F. De Rango, and P. Raimondo, "Ai-powered multi time-scale energy forecasting in iot enabled smart home environment", *IEEE Access*, 2021 - Submitted.

[48] J. Zheng, C. Xu, Z. Zhang, and X. Li, "Electric load forecasting in smart grids using long-short-term-memory based recurrent neural network", in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, IEEE, 2017, pp. 1–6.

[49] W. Kong, Z. Y. Dong, Y. Jia, D. J. Hill, Y. Xu, and Y. Zhang, "Short-term residential load forecasting based on lstm recurrent neural network", *IEEE Transactions on Smart Grid*, vol. 10, no. 1, pp. 841–851, 2017.

[50] M. Abdel-Nasser and K. Mahmoud, "Accurate photovoltaic power forecasting models using deep lstm-rnn", *Neural Computing and Applications*, vol. 31, no. 7, pp. 2727–2740, 2019.

[51] U. M. L. Repository. (2020). Individual household electric power consumption data set, [Online]. Available: `https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption`.

[52] Keras. (2020). Keras - a deep learning api written in python, [Online]. Available: `https://keras.io/api/`.

[53] WEKA. (2020). Weka - the workbench for machine learning, [Online]. Available: `https://www.cs.waikato.ac.nz/ml/weka/`.

[54] L. Breiman, "Random forests", *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[55] Y. Wang and I. H. Witten, "Induction of model trees for predicting continuous classes", in *Poster papers of the 9th European Conference on Machine Learning*, Springer, 1997.

[56] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive internet of things: A new paradigm beyond connection", *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 129–143, 2014.

[57] P. K. D. Pramanik, S. Pal, and P. Choudhury, "Beyond automation: The cognitive iot. artificial intelligence brings sense to the internet of things", in *Cognitive Computing for Big Data Systems Over IoT*, Springer, 2018, pp. 1–37.

[58] M. Zhang, H. Zhao, R. Zheng, Q. Wu, and W. Wei, "Cognitive internet of things: Concepts and application example", *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 6, p. 151, 2012.

[59] D. Kelaidonis, A. Somov, V. Foteinos, G. Poulios, V. Stavroulaki, P. Vlacheas, P. Demestichas, A. Baranov, A. R. Biswas, and R. Giaffreda, "Virtualization and cognitive management of real world objects in the internet of things", in *2012 IEEE International Conference on Green Computing and Communications*, IEEE, 2012, pp. 187–194.

[60] I. E. Agency. (2020). Buildings a source of enormous untapped efficiency potential, [Online]. Available: `https://www.iea.org/topics/buildings`.

[61] R. J. de Dear, T. Akimoto, E. A. Arens, G. Brager, C. Candido, K. Cheong, B Li, N Nishihara, S. Sekhar, S Tanabe, *et al.*, "Progress in thermal comfort research over the last twenty years", *Indoor air*, vol. 23, no. 6, pp. 442–461, 2013.

[62] H. Do and K. S. Cetin, "Data-driven evaluation of residential hvac system efficiency using energy and environmental data", *Energies*, vol. 12, no. 1, p. 188, 2019.

[63] D. N. Avendano, J. Ruyssinck, S. Vandekerckhove, S. Van Hoecke, and D. Deschrijver, "Data-driven optimization of energy efficiency and comfort in an apartment", in *2018 International Conference on Intelligent Systems (IS)*, IEEE, 2018, pp. 174–182.

[64] T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building hvac control", in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1–6.

[65] R. A.-C. E. AMERICAN SOCIETY OF HEATING, *Standard 55: Thermal environmental conditions for human occupancy*, 2010.

[66] F. Tartarini and S. Schiavon, "Pythermalcomfort: A python package for thermal comfort research", *SoftwareX*, vol. 12, p. 100 578, 2020.

[67] D. Maltoni and V. Lomonaco, "Continuous learning in single-incremental-task scenarios", *Neural Networks*, vol. 116, pp. 56–73, 2019.

[68] A. Serianni, F. De Rango, and P. Raimondo, "Cognitive iot enabled by layered architecture and neural networks in a smart home environment", *12th Wireless Days Conference (WD 2021)*, 2021.

[69] A. Serianni and F. De Rango, "Cognitive iot solution based on neural networks for the smart home", *IEEE Internet of Things Journal*, 2021 - Submitted.