

A Distributed Architecture for Fuzzy Logic Systems and its Application in Human Activity Recognition

Bhavesh Himmatbhai Pandya

A thesis submitted in partial fulfilment of the requirements of
Nottingham Trent University for the degree of

Doctor of Philosophy

March 2023

This thesis is dedicated to my wife, parents, children, and the entire Pandya family for their unwavering spiritual, emotional, and financial support.

Acknowledgements

The work described in this thesis was carried out at Nottingham Trent University between April 2019 and March 2023, whilst the researcher was working as a full-time doctoral research student.

During the last four years of my Ph.D. research, I have been fortunate to have Dr. Amir Pourabdollah as my honourable director of studies. I am eternally grateful to him for all the time, effort, encouragement, and knowledge he has generously embedded in me. Despite his various responsibilities, his office was always open to any queries. Furthermore, his unwavering efforts and extensive suggestions regarding my work were crucial for the achievement of this research.

I would like to express my gratitude to Prof. Ahmad Lotfi for his guidance, support, feedback, and expertise during the development of my thesis. Their dedication to the success of this project was remarkable.

I am also grateful to Dr. Taha Osman for providing constructive feedback during the annual review progress meetings.

I thank my wife, Rachana Pandya, for being the driving force behind my success. It would not have been possible without her constant support and unrelenting efforts to manage the workload while I pursued my Ph.D. I would like to extend my sincere gratitude to my children (Nishtha, Dirgh, and Darsh), sister (Mamta Jani), brother (Prakash Pandya), sister-in-law (Dharmishtha Pandya), niece (Suhani), nephew (Jaymin), and (Prithvi), as well as to our extended family and many friends. Special thanks to my parents, Himmatbhai Pandya and Savitaben Pandya, as well as my father-in-law,

Arunbhai Jani, and mother-in-law, Geetaben Jani, for their words of encouragement and push for perseverance. The sacrifices of both of my parents have contributed to who I am today. I have no words to express my gratitude for all you have done for me.

Additionally, I dedicate this dissertation to all my supportive friends. I will always be grateful for what they have done, especially Dr. Abdallah Naser's assistance with the countless hours of proofreading. This work is dedicated to my closest buddy Ajay who was my constant source of motivation throughout my doctoral study. In addition, I would like to express my appreciation to my friends and family, who have always been there for me during the good times and the bad. Their prayers and faith inspired and encouraged me to embark on this journey.

Specifically, I want to express my sincere gratitude to Dr Manoj Kowar, Dr. Rajesh Bansode, Madhur, Kunal, and Dr Vinayak Bharadi for their motivation to pursue my Ph.D. studies. Your confidence in my capabilities really helped me in more ways than one.

In addition, I would like to express my gratitude to the Computational Intelligence and Applications Research Group members with whom I have worked and collaborated over the years.

I am grateful to Nottingham Trent University for funding my Ph.D. studies through a fully-funded scholarship scheme. The financial support provided by the School of Science and Technology to attend various international conferences is also greatly appreciated. I would not have been able to achieve this academic milestone without the support of this scholarship.

Bhavesh Himmatbhai Pandya
March 2023

Abstract

Fuzzy Logic Systems (FLS) have the full potential in handling imprecise and uncertain data due to the inherent advantages of the Fuzzy Inference System (FIS). Traditionally, fuzzy logic systems are linked to specific hardware or software systems. The literature review reveals that dispersed and distributed architectures of FLS are in high demand due to their capability to handle the complexities of fuzzy logic computations. However, the absence of best practices and standard methodologies prevents widespread adoption. As a result, some specific requirements, such as web communications and Service-Oriented Architecture (SOA), which can be found in many modern systems, are rarely adapted for FLSs. Sharing FLSs accessibility as web services (called Fuzzy-as-a-Service alias FaaS), in which the service is developed independently from a specific client platform, allows for autonomy, openness, load balancing, efficient resource allocation and eventually cost-effective, particularly for computationally intense FLSs.

The proposed novel architectural solution (FaaS) is a web-based service that distributes the main services for FLS on more than one client and servers nodes that can reach multiple users. By extending the IEEE-1855 (2016) standard in terms of system definition and data exchange, this research offers a standard solution for building FaaS as a novel method of implementing fuzzy logic systems by means of a cloud-based collecting, processing, and examining data over the web. Recent advances in standardising Fuzzy Mark-up Language (IEEE 1855-2016) and its associated software libraries (such as JFML and Simplful) have made this achievable. Two

different cloud service providers and software libraries (Amazon Web Services using JFML as a java-based library and Azure Web Services using Simplful as a python-based library) are exploited to realise the FaaS on the cloud.

As a case study to establish the efficacy of the proposed FaaS, Human Activity Recognition (HAR) that plays a pivotal role in monitoring the health status of the Persons Under Observation (PUO) has been taken under consideration. In order to monitor the data related to HAR and physiological data, which are imprecise and uncertain in nature, various previous researchers have developed a good number of machine learning tools. However, such monitoring systems suffer from certain limitations due to the nature and amount of data being analysed.

A number of experiments are carried out in order to showcase and evaluate FaaS performance in different HAR scenarios. The first scenario has been a real-time walking/running detection. Secondly, a fall detection system via FaaS is designed based on IEEE 1855-2016 and JFML. In view, the pandemic caused due to COVID-19, the third application dealt with developing a system to determine the health status of individuals by remotely monitoring their Oxygen saturation and heartbeat rate using wearable sensors. Finally, a performance comparison between a stand-alone fuzzy system and a FaaS solution for fall detection is performed on two different cloud services, namely AWS and Azure. Research findings exhibit that while the proposed algorithm can keep the same accuracy as a stand-alone fuzzy system (90%), it can significantly improve the processing time, e.g., reducing the processing time for 10K data samples from 179 to 45 seconds (78% improvement).

Towards the end of this PhD project, the new IEEE 1855 extension is taken as a proposal into the consideration of the IEEE standards committee and is currently in the process of final approval in 2023.

Publications

The following publications have been published as a direct result of this thesis:

Refereed Journal Papers

Pandya, B., Pourabdollah, A. and Lotfi, A. A Comparative Study of Stand-alone and Cloud-Based Fuzzy Logic Systems for Human Fall Detection. *Int. J. Fuzzy Syst.* (2022). <https://doi.org/10.1007/s40815-022-01437-2>

Pandya, B.; Pourabdollah, A.; Lotfi, A. Comparative Analysis of Real-Time Fall Detection Using Fuzzy Logic Web Services and Machine Learning. *Technologies* 2020, 8, 74. <https://doi.org/10.3390/technologies8040074>

Refereed Conference Papers

Bhavesh Pandya, Amir Pourabdollah, and Ahmad Lotfi. 2023. An Integrated Development Environment for the Design of Fuzzy-Human-centric System in accordance with IEEE Standard 1855-2016. In Proceedings of the 16th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '23). (This research paper was submitted at the PETRA '23 conference, but a final decision has not been made at the time of submission of this thesis.)

B. Pandya, A. Pourabdollah, A. Lotfi and G. Acampora, "An Integrated Fuzzy Logic System under Microsoft Azure using Simplful," 2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2022, pp. 1-9, doi: 10.1109/FUZZ-IEEE55066.2022.9882612.

Bhavesh Pandya, Dhaval Shah, Amir Pourabdollah, and Ahmad Lotfi. 2022. Developing and Comparing Cloud-based Fuzzy Systems for Monitoring Health Related Signals in Assistive Environments. In Proceedings of the 15th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '22). Association for Computing Machinery, New York, NY, USA, 407–413. <https://doi.org/10.1145/3529190.3534742>

B. Pandya, A. Pourabdollah, A. Lotfi and G. Acampora, "Developing a cloud-based service-oriented architecture for fuzzy logic systems," 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2021, pp. 1-8, doi: 10.1109/FUZZ45933.2021.9494530.

Bhavesh Pandya, Amir Pourabdollah, and Ahmad Lotfi. 2021. A cloud-based pervasive application for monitoring oxygen saturation and heart rate using fuzzy-as-a-service. In The 14th PErvasive Technologies Related to Assistive Environments Conference (PETRA 2021). Association for Computing Machinery, New York, NY, USA, 69–75. <https://doi.org/10.1145/3453892.3453998>.

B. Pandya, A. Pourabdollah and A. Lotfi, "Fuzzy-as-a-Service for Real-Time Human Activity Recognition Using IEEE 1855-2016 Standard," 2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2020, pp. 1-8, doi: 10.1109/FUZZ48607.2020.9177781.

Bhavesh Pandya, Amir Pourabdollah, and Ahmad Lotfi. 2020. Fuzzy logic web services for real-time fall detection using wearable accelerometer and gyroscope sensors. In Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '20). Association for Computing Machinery, New York, NY, USA, Article 54, 1–7. <https://doi.org/10.1145/3389189.3397989>.

Bhavesh Pandya, Amir Pourabdollah, and Ahmad Lotfi. 2023. An Integrated

Development Environment for the Design of Fuzzy Systems in accordance with IEEE Standard 1855-2016. In Proceedings of the 16th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '23). (The above-mentioned research paper was submitted at the PETRA '23 conference, but a final decision has not been made at the time of submission of this thesis.)

Nomenclature

The following Acronyms are used throughout the thesis.

Acronyms

AAL	Ambient Assisted Living
ADL	Activities of Daily Living
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AmI	Ambient Intelligence
ANN	Artificial Neural Network
AnYa	Angelov-Yager
API	Application Programming Interface
ARS	Activity Recognition Systems
AWS	Amazon Web Service
BOA	Bisector of Area Method
bpm	beats per minute
CIN	Central Intelligent Node
COA	Centroid of Area
COG	Center of gravity
COS	Center of Sums
DA	Delta A
DL	Deep Learning
DOLARS	Distributed On-line Activity Recognition System
EBT	Ensemble Bagged Tree
EC2	Elastic Compute Cloud
ECOC	Error-correction-output-codes
ECG	electrocardiogram

FaaS	Fuzzy-as-a-Service
FCL	Fuzzy control Language
FF	Feed-Forward
FIS	Fuzzy Inference System
FL	Fuzzy Logic
FLC	Fuzzy Logic Controller
FLS	Fuzzy logic System
FML	Fuzzy Mark-up Language
FN	False Negative
FOM	First of Maxima
FP	False Positive
GPS	Global Positioning System
GSM	Global System for Mobile Communication
HR	Heart Rate
HAR	Human Activity Recognition
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
IoT	Internet of Things
JFML	Java Fuzzy Mark-up Language
KB	Knowledge Base
kNN	K-nearest neighbours
LOM	Last of Maxima
MF	Membership Function
ML	Machine Learning
MTBF	Mean Time Between Failure
MHS	Medical Health Server
MOM	Mean of Maxima
NodeMCU	Node Microcontroller Unit
PaaS	Platform as a Service
PERS	Personal Emergency Response System
PI	Perfusion Index
PR	Pulse Rate
PUO	Person under observation
QoS	Quality of Service
QSVM	Quadratic Support Vector Machine

RB	Rule Base
RF	Random Forest
RFID	Radio-Frequency Identification
SaaS	Software as a Service
SMS	Short Message Service
SPHERE	Sensing Platform for Healthcare in a Residential Environment
SOA	Service-Oriented Architecture
SOWBAN	Service Oriented Wireless Body Area Networks
SVM	Support Vector Machine
SW-SHMS	Smart Healthcare Monitoring System
TFC	Transparent Fuzzy Control
TN	True Negative
TP	True Positive
TSK	Takagi-Sugeno-Kang
WBAN	Wireless Body Area Network
WHO	World Health Organisation
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformations

Contents

Dedication	i
Acknowledgements	ii
Abstract	iv
Publications	vi
Nomenclature	ix
Contents	xii
List of Figures	xvi
List of Tables	xix
1 Introduction	1
1.1 Motivation	1
1.2 Human Activity Recognition	3
1.3 Decision-Making Processes	5
1.4 Fuzzy framework for decision making process in HAR	9
1.5 Distributed Architecture in HAR	12
1.6 Cloud Computing in HAR	12
1.7 Major Considerations in Implementation	13
1.8 Research Objectives	15
1.9 Research Approach and Design	17
1.10 Relevance and Contributions	18

1.11 Thesis Outline	19
2 Literature Survey and Scope of Work	22
2.1 Introduction	22
2.2 Distributed and Cloud-based HAR	22
2.3 Fuzzy Logic Systems for HAR	24
2.4 Distributed and Cloud-based Fuzzy Logic Systems for HAR	27
2.5 Data Exchange Formats and Standards for FLSs	30
2.6 Scope of The Work	34
3 Methods, Materials and Framework	37
3.1 Introduction	37
3.2 Motivation	38
3.2.1 Openness and Platform Independence	39
3.2.2 Configuring the Fuzzy Systems from Anywhere	40
3.2.3 Reuse of Computational Results	40
3.2.4 Scalability of Fuzzy Logic System	40
3.2.5 Need for a Standard Web-based Language	41
3.3 Fuzzy-as-a-Service: The Proposed Software and Hardware Architecture	41
3.3.1 Architecture Components	42
3.3.1.1 Fuzzy Logic Server(s)	43
3.3.1.2 Environmental Devices and Apps	44
3.3.1.3 Management Station(s)	45
3.3.1.4 Monitoring Station(s)	45
3.4 Proposed Extension of IEEE 1855	45
3.4.1 Need for Extending IEEE 1855 for FaaS	46
3.4.2 Extension Proposal and API Design	46
3.4.2.1 createFLS	48
3.4.2.2 setInput	50
3.4.2.3 getInput	51
3.4.2.4 getOutput	51
3.4.2.5 queryFLS	52

3.4.2.6	deleteFLS	53
3.4.3	FaaS Implementation on the Cloud	54
3.4.4	Deployment Plan	54
3.4.5	Data Collection	54
3.5	Chapter Summary	55
4	A Cloud-based FaaS for Fall Detection to Recognize Human Activities	57
4.1	Introduction	57
4.2	Proposed System for Fall Detection	64
4.2.1	Data Acquisition using Wearable Sensors	64
4.2.2	Data Diagnosis	65
4.2.3	Data Processing	67
4.2.4	Fuzzy Logic System Design	69
4.2.5	Modelling Approach using Machine Learning Techniques .	70
4.3	Experiment Methodology for HAR	75
4.3.1	Dataset Used	75
4.3.2	FLS Design	77
4.3.3	Rule-base Training	77
4.4	Experimental Settings	79
4.5	Results and Discussions of experiment performed for fall detection	80
4.6	Results and Discussions of experiment performed for HAR	84
4.7	Chapter Summary	86
5	A Cloud-Based FaaS for Pervasive Health Conditions Monitoring	88
5.1	Introduction	88
5.2	Methodology	90
5.3	Fuzzy System Design	93
5.4	Software Components	95
5.5	Experimental Results	97
5.6	Chapter Summary	99

6	A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection	101
6.1	Introduction	101
6.2	Methodology	102
6.2.1	Data Collection	102
6.2.2	Fuzzy Logic System Design	104
6.2.2.1	Stand-alone Architecture	104
6.2.2.2	Cloud-based Architecture	105
6.2.3	Non-FLS Machine Learning Design	106
6.2.4	UI Design	107
6.3	Experimental Results	107
6.3.1	Offline Standalone FLS	107
6.3.2	Offline Non-FLS Machine Learning	109
6.3.3	Real-time Cloud-based FLS	109
6.4	Discussion	109
6.5	Chapter Summary	117
7	Conclusions and Future Scopes	119
7.1	Thesis Summary	119
7.2	Results and Discussion	120
7.3	Conclusions	121
7.3.1	Revisiting the Objectives	122
7.4	Directions for Further Research	125
	Appendix A: Development of AWS and Azure Web Services	126
	Appendix B: JFML Library Details	134
	References	139

List of Figures

1.1	Architecture of Fuzzy Inference System.	11
1.2	Schematic block diagram of distributed FLS.	15
1.3	Thesis structure indicating the organization of the chapters and their respective dependencies.	21
2.1	The object model tree structure of FML in defining a Fuzzy controller.	33
3.1	An illustrative diagram of client-server communication through the API.	42
3.2	A schematic representation of the proposed system.	43
3.3	Example of FML labelled tree.	44
3.4	Main class diagram of JFML.	44
3.5	Schema of request/response the designed fuzzy-as-a-service API. The fuzzySystem element is the core element that follows the IEEE1855-2016 standard, whereas the other elements are to be considered as extensions for the Web services	47
4.1	System architecture of a fall detection system using both ML and FaaS.	65
4.2	Trend Plot for fall and non-fall activities: (a) Side Fall, (b) Forward Fall, (c) Backward Fall, (d) Normal Walk and (e) Running.	66
4.3	Normal distribution of the obtained accelerometer and gyroscope data	67
4.4	Histogram plots of the obtained accelerometer sensor	68

LIST OF FIGURES

4.5	Selection of a data sample.	69
4.6	Membership for fuzzy inputs:(a) SVM_G (Degree/s) (b) ΔA (Degree) and membership for output (c) Fall Detection.	71
4.7	Flow chart of the proposed fall detection system.	73
4.8	Partial KB and RB of the adopted FML.	76
4.9	A sample list of rules created by the rule-base training algorithm.	78
4.10	The elements of the carried-out experiment for the HAR scenario.	79
4.11	Real-time human activity classification process using HTTP request/responses to the developed Web Services. A sensor client console view: sending back-to-back XML requests to the server to set the input variables (i.e., the accelerometer and gyroscope data).	81
4.12	Consoles views of two monitoring clients: On the right side the server acknowledgements per request are received; On the right side, the output values (i.e., the activity classification) per request are coming from the server (the client's requests for getting the output values are not shown here).	84
4.13	A screenshot of fall detection request/response in FaaS	85
5.1	Schematic illustration of the proposed system.	90
5.2	Framework of the proposed architecture including data collection point, data processing, monitoring point, design point, and fuzzy logic system.	91
5.3	Transmittance oxygen saturation monitoring principle.	92
5.4	Membership functions for fuzzy inputs:(a) SpO_2 (b) Pulse rate and membership for output (c) Health Condition.	94
5.5	List of methods created over Microsoft Azure.	96
5.6	The User interface of the FLS Designer application. Under the "Create FLS" tab, an FLS is being designed on the client side for being sent to the server side. Other tabs are responsible for other input/output tasks as defined in the proposed extended FML.	97
5.7	Real-time human health monitoring and processing result a) Critical b) Normal and c) Alert.	98

LIST OF FIGURES

6.1	System architecture of a standalone fuzzy system using the Mobile application.	104
6.2	The suggested architecture includes data collecting, data processing, feature extraction, and a fuzzy logic system.	105
6.3	UI screen for Android mobile application.	108
6.4	Real-time human health monitoring and processing result using A)Create FLS B) Set Input C) Get Output methods.	108
6.5	Snapshot of fall detection output using smartwatch A) Non-fuzzy machine learning using cloud B) Fuzzy with simplful libraries and Azure C) Fuzzy with JFML libraries and AWS D) Fuzzy standalone.	110
6.6	Comparative analysis of human fall detection.	111
6.7	Fall detection using smartwatch A) Total time required to process the samples B) Number of samples processed per second.	113
6.8	Fall detection using mobile phone A) Total time required to process the samples B) Number of samples processed per second.	116
1	Create FLS.	128
2	Setinput FLS.	128
3	GetOutput FLS.	128
4	Delete FLS.	129
5	Azure services.	129
6	FLS Simplful.	130
7	List of functions.	131
8	Get FLS.	131
9	Get FLS code.	132
10	Database connect.	133
11	Main Class Diagram for the definitions of KB.	135
12	Main Class Diagram for the definitions of RBs.	137

List of Tables

2.1	Comprehensive Overview of HAR Features Explored in the Relevant Studies.	25
2.2	List of software that can be used to make FISs.	35
3.1	request/responses messaging format for the minimum services. . .	49
4.1	Rules for FLS to detect the risk of fall.	72
4.2	A sample part of the run/walk dataset used in this experiment for both training and testing.	75
4.3	Fall detection result using FaaS	82
4.4	Training and testing of a dataset with respective time using of ANN.	82
4.5	Result analysis of ANN.	83
4.6	Comparison of different machine learning algorithms.	83
4.7	Result analysis of Random Forest.	85
4.8	Real-time Human Activity Monitoring result.	86
5.1	Dataset with a few instances of feature-extracted data.	91
5.2	FLS rules for calculating health status.	95
5.3	The analysis of cloud-based FLS on the provided dataset.	99
6.1	Dataset with a few instances of feature-extracted data.	103
6.2	Comparative analysis of real-time and non-real-time human fall detection.	111
6.3	Summary of real-time and non-real-time human fall detection system.	111
6.4	Number of samples processed per second using the smartwatch.	112

LIST OF TABLES

6.5	Average processing time (in seconds) with respect to number of samples using the smartwatch	114
6.6	Average processing time (in seconds) with respect to number of samples using smart phone	114
6.7	Number of samples processed per second using smart phone	116

Chapter 1

Introduction

1.1 Motivation

Human resource is the best living resources to be preserved and maintained carefully for the betterment of human civilisation. Better healthcare services are facing significant complications in terms of quality, cost, transparency, sustainability, and secrecy of individual data. Hence, technology intervention has become inevitable in developing a sustainable healthcare service system for the ultimate growth of the global population and economy. The utilisation of information and communication technology to provide autonomous and proactive healthcare services will be extremely beneficial. Consumer-driven healthcare, web-based platforms, and electronic health records have enabled a variety of enhanced healthcare solutions during the last several decades. Human Activity Recognition (HAR) is one of the most essential steps in monitoring the health status of a person, especially older adults. HAR provides benefits such as health monitoring, detecting activities and discovering their patterns, and improving general well-being. However, addressing the high computation demands in HAR while dealing with high data sample rates and possible data uncertainties exist in such applications, have become some of the major challenges. Taking up a research project on the design and implementation of a cloud-based architecture based on fuzzy logic with human activity recognition as a case study can be motivated by several factors and potential benefits such as:

- **Emerging Technology:** Cloud computing and fuzzy logic are both emerging and rapidly evolving fields. Researching the combination of these technologies can lead to innovative solutions that address complex problems more effectively.
- **Personal Interest and Curiosity:** Researchers might have a personal interest in artificial intelligence, human-computer interaction, or smart systems. Exploring the fusion of fuzzy logic and cloud architecture for human activity recognition can be intellectually stimulating.
- **Real-World Applications:** Human activity recognition has numerous real-world applications, including healthcare monitoring, smart homes, security systems, and more. Designing a cloud-based architecture using fuzzy logic could enhance the accuracy and adaptability of such systems.
- **Improved Accuracy and Robustness:** Fuzzy logic can handle uncertain and imprecise data effectively. By integrating it with cloud-based architecture, researchers can aim to improve the accuracy and robustness of human activity recognition systems.
- **Scalability:** Cloud computing offers scalability, enabling systems to handle large amounts of data and users. Researching how fuzzy logic can be integrated into cloud architecture can lead to scalable solutions for human activity recognition.
- **Customization and Adaptability:** Fuzzy logic allows for rule-based systems that can be customized easily. When implemented in a cloud-based architecture, these systems can adapt to different users' preferences and environments, enhancing the overall user experience.
- **Cross-Disciplinary Research:** This type of project can bridge the gap between computer science, artificial intelligence, and human behavior studies. It encourages collaboration between researchers from different disciplines, leading to diverse insights and perspectives.
- **Challenging Technical Problems:** Designing a cloud-based architecture that efficiently processes and analyzes human activities using

fuzzy logic involves various technical challenges, such as data preprocessing, rule generation, and real-time analysis. Solving these challenges can contribute to the advancement of the field.

- **Commercial Potential:** Successful implementation of a cloud-based system for human activity recognition could have commercial potential in various industries, attracting interest from companies looking to enhance their products or services.
- **Contributions to Knowledge:** This research can contribute to the academic and scientific community's understanding of how fuzzy logic can be integrated into cloud architectures for practical applications. It could lead to the development of new algorithms, methodologies, and best practices.
- **Ethical and Privacy Considerations:** Exploring the implications of using cloud-based human activity recognition systems raises ethical and privacy concerns. Researchers can address these issues and propose solutions that prioritise user privacy and data security.

In summary, a research project on the design and implementation of a distributed architecture using fuzzy logic for human activity recognition can bring together cutting-edge technologies, address real-world challenges, and contribute to both theoretical knowledge and practical applications. The main focus of the research presented in this thesis is the development of a cloud-based distributed system for monitoring human activities and health status using Fuzzy logic-based decision support system.

1.2 Human Activity Recognition

Ubiquitous computing, also known as pervasive computing, refers to a concept in computer science and human-computer interaction where computing technology is seamlessly integrated into the environment and becomes an integral part of everyday life. The idea behind ubiquitous computing is to create a computing

environment that is so pervasive and unobtrusive that it virtually disappears, blending into the background of our daily activities.

HAR: Human activity recognition (HAR) can be referred to as the art of identifying and naming activities using Artificial Intelligence (AI) from the gathered activity raw data by utilizing various sources (so-called devices).

Uncertainties in HAR: The challenges of human activity recognition lie in the complexity of human behaviours and movements, uncertainties in the data gathered by the sensors due to noisy environments, running out of batteries, imprecise outputs, missing activities, misnaming activities, communication failures, etc.

A prime objective of ubiquitous computing is to provide accurate and timely information on people's activities and routines. During the past decade, advancements in microelectronics and computer systems have allowed for the customisation of sensors and mobile devices. The devices' small size, high capacity for network connectivity, and reasonable cost make them attractive for widespread usage. Several tactical, military, law enforcement, and medical uses might be devised. Ubiquitous sensing, an active area of study whose primary goal is to derive insight from data gathered by ubiquitous sensors, was inspired by this challenge [1].

Unexpectedly, HAR has emerged as a significant priority in the field, notably for healthcare, military, and security applications. Patients with specific diseases, including diabetes, obesity, and cardiovascular disease, may be recommended an exercise regimen [2]. Consequently, acknowledging actions like walking, jogging, or cycling is beneficial for providing feedback to the carer regarding the patient's behaviour. Also, monitoring individuals with dementia or other mental diseases might help spot abnormal behaviour and prevent undesirable consequences [3]. Understanding of human activity has grown to be one of the most widely applied fields of study, enabling significant improvements in elderly people's quality of life as well as patient care at home [4].

Since HAR can gather data on people's daily activities, its potential uses have expanded with the availability of acquisition tools such as smartphones and video cameras. These programs improve the quality of life and medical care for the elderly and dependants [5, 6]. HAR is used in fields such as: HAR has a wide

range of applications across various fields due to its potential to understand and analyze human behavior in different contexts. Here are some fields where human activity recognition can be applied: **Healthcare and Well-being** include Fall detection for the elderly, Monitoring patients' daily activities to ensure they are following prescribed routines, Physical therapy, and rehabilitation monitoring, and Activity tracking for fitness and wellness.

Smart Homes and Ambient Assisted Living ensembles Automating home appliances based on user activities (e.g., turning off lights when no one is in the room), Detecting unauthorised access to secure areas, and Adjusting temperature and lighting based on occupants' preferences and presence.

Sports and Athletics comprise performance analysis for athletes to improve technique and prevent injuries, tracking movements and activities in team sports for strategy optimization, and monitoring physical activities for training purposes.

Security and Surveillance contains intruder detection and alert systems, suspicious behavior detection in public spaces, and monitoring for unusual activities in restricted areas.

Human-Computer Interaction (HCI) covers gesture recognition for controlling devices without physical touch, context-aware interfaces that adapt based on user activities, and natural user interfaces in virtual reality (VR) and augmented reality (AR) systems.

Transportation and Automotive comprehends driver behavior analysis for insurance purposes, monitoring fatigue and alertness of drivers for road safety.

Elderly Care consists of providing insights into the daily routines and activities of seniors living independently and detecting deviations from normal behavior that might indicate health issues.

Emotion Recognition deals with identifying emotional states based on facial expressions and body language, and personalised content delivery in entertainment and therapy.

1.3 Decision-Making Processes

The foremost goal of HAR is to predict the movements or actions of a person based on the past action data collected by some data acquisition devices.

Although numerous Machine Learning (ML) and Deep Learning (DL) techniques within Artificial Intelligence (AI) framework offer for predicting activities based on the data collected from the employed sensors, it is a challenging task as it involves dealing with huge amounts of unlabelled, imprecise and uncertain data. It becomes more challenging when it comes to drawing decisions based on uncertain data using standard decision-making algorithms. Various decision-making techniques for HAR include Decision Trees, Random Forests, Support Vector Machines (SVM), Neural Networks, Hidden Markov Models (HMM), Gaussian Mixture Models (GMM), K-Nearest Neighbors (K-NN), Ensemble Learning, Dynamic Time Warping (DTW), Fuzzy Logic, Bayesian Networks, and Long Short-Term Memory (LSTM).

The decision-making process in human activity recognition (HAR) involves several steps that allow a system to identify and classify human activities based on sensor data and other input sources. Here's an overview of the decision-making process in HAR:

Data Collection: HAR systems rely on data from various sensors, such as accelerometers, gyroscopes, and sometimes additional inputs like audio or video. These sensors capture raw data related to body movements, gestures, and environmental conditions.

Data Preprocessing: Raw sensor data often contain noise, outliers, and variations that need to be filtered or normalized. Preprocessing involves techniques like filtering, feature extraction, and data alignment to ensure that the data is suitable for analysis.

Feature Extraction: Relevant features are extracted from the preprocessed data to represent distinct patterns and characteristics of different activities. These features might include statistical measures, time-domain features, frequency-domain features, and more.

Feature Selection/Dimensionality Reduction (Optional): In some cases, not all extracted features are relevant. Dimensionality reduction techniques can be used to reduce the number of features while preserving the most important information. This helps improve the efficiency and accuracy of the recognition process.

Model Training: HAR involves machine learning techniques, where models

are trained on labeled training data. Common algorithms include decision trees, random forests, support vector machines, and neural networks. During training, the model learns to associate specific features with corresponding activity labels.

Model Evaluation: The trained model is evaluated using a separate dataset (validation or testing set) that it hasn't seen before. Evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices are used to assess the model's performance.

Model Optimization (Optional): If the model's performance is not satisfactory, optimization techniques like hyperparameter tuning or changing the model architecture might be employed to enhance accuracy and generalization.

Real-Time Recognition: In a real-world scenario, the HAR system receives new sensor data in real time. The trained model predicts the activity label based on the extracted features from the incoming data.

Post-Processing (Optional): Additional post-processing steps might be performed to smooth predictions over time, eliminate short-lived fluctuations, or apply temporal constraints to the recognised activities.

Decision and Output: The final decision is made based on the prediction probabilities or confidence scores assigned to each recognised activity class. The activity label with the highest probability becomes the recognised activity.

User Feedback and Interaction (Optional): In some applications, user feedback can be used to further refine the recognition process. For instance, users might correct misclassifications, helping the system adapt and improve over time.

Continuous Learning (Optional): Some HAR systems employ techniques for continuous learning, where the model updates itself with new data over time to adapt to changes in user behavior or the environment.

The HAR system has many possible configurations since each phase can be implemented in several ways. As a result, the possibilities become even more convoluted when considering the area of application, the type of data collection device, and the processing of AI algorithms for activity detection. Available data analysis techniques widely differ in the data types, how they transform the collected data, and the statistical methods they apply for inference and/or classification. The objective of classifier selection is to choose a method that yields the highest achievable classification accuracy given the available training

data and processing environment (e.g., online vs. offline).

Activity Recognition Systems (ARS) are crucial components in various fields, implemented through a range of machine learning techniques. Classically supervised and unsupervised methods are commonly employed in developing ARS. Supervised analysis techniques, as highlighted by [7], include Decision Trees, Support Vector Machines, Naive Bayesian Classifiers, Artificial Neural Networks, Decision Tables, and Logistic Models. These algorithms are adept at learning from labeled data, enabling accurate classification of activities.

On the other hand, Unsupervised Learning methods, such as Clustering and Association Rules Dimensionality Reduction, play a significant role in ARS development. These techniques excel in identifying patterns and relationships within unlabeled data, offering insights into complex activity datasets. Ensemble Learning techniques like Stacking, Bagging, and Boosting further enhance ARS performance by combining multiple models to improve predictive accuracy and robustness. These methods leverage the diversity of individual models to collectively achieve superior results. Moreover, the emergence of Deep Neural Networks (DNNs) has revolutionized ARS, enabling multi-level feature extraction and learning for knowledge discovery. DNN-based approaches facilitate more nuanced analysis of activity data, leading to deeper insights and improved performance.

In recent years, the evolution of machine learning has extended to include Reinforcement Learning (RL) for ARS. RL algorithms enable agents to learn optimal behavior by interacting with an environment, receiving rewards for desirable actions and punishments for undesirable ones. This dynamic learning paradigm enhances ARS adaptability and autonomy, making it well-suited for real-world applications where activities may vary over time.

In conclusion, the diverse array of machine learning techniques, from classical supervised and unsupervised methods to advanced ensemble learning and reinforcement learning approaches, collectively contribute to the development of robust and effective Activity Recognition Systems. These techniques enable ARS to accurately detect and classify activities, extract meaningful insights, and adapt to dynamic environments, ultimately enhancing their utility in various domains.

Activity Recognition Systems (ARS) are implemented by employing classically supervised or unsupervised-based machine learning. Among the unique algorithms of Supervised Analysis, the following can be highlighted [7]: Decision Tree, Support Vector Machine, Naive Bayesian Classifier, Artificial Neural Networks, Decision Tables, and Logistic Models. Regarding Unsupervised Learning, several methods can be found, among which Clustering and Association Rules Dimensionality Reduction can be highlighted. Ensemble Learning techniques such as Stacking, Bagging, and Boosting can also be highlighted. Methods or techniques based on Deep Neural Networks with several levels of analysis for knowledge discovery have also gained much attention. Nowadays, ML has evolved to analysis based on Reinforcement Learning, which allows the algorithm that is strengthened in a system of rewards and punishments to permeate the learning process.

The above decision-making processes provide satisfactory results as long as the size of the data is manageable. Hence, significant technical challenges occur due to the nature and size of the data.

1.4 Fuzzy framework for decision making process in HAR

The data collected from wearable sensors of a concerned persons vary a lot depending on the physiological factors affecting them. Moreover, no sharp distinctions can be drawn among the boundaries of the physiological data collected for making a decision. Hence, bi-valued logic is not suitable for concluding the status of the health of the concerned person. Multi-valued logics, such as Fuzzy logic, are the alternative approaches for cases where bi-valued logic can not be applied in decision support systems[8, 9].

Based on the facts provided earlier, the adoption of fuzzy logic in our platform is inspired by four primary reasons: first, the characteristic of data to merge, which are measurements received from various sensors, may be ambiguous and imprecise. Secondly, the lack of training sets that reflect activities of daily living can greatly influences the decision-making process. Thirdly, fuzzy logic can gather

performance and intelligibility, dealing with imprecision and uncertainty in data. Finally, the fuzzy logic takes a close approach to how humans (i.e., experts) make decisions. Its history shows that it is used in many applications in healthcare including pattern recognition and clinical diagnosis [10]. For medical experts, it is easier to map their knowledge onto fuzzy sets and fuzzy rules than to manipulate complex probabilistic tools.

In real-world situations, we do not have enough knowledge to determine whether a particular condition is true or untrue. Fuzzy logic provides valued thinking flexibility. As a result, we may consider potential errors and ambiguities in each given circumstance. In the Boolean system, 1.0 represents the absolute truth value, and 0.0 represents the absolute false value. Fuzzy logic allows for values that are neither completely true nor completely false.

Fuzzy sets theory was introduced by L. A. Zadeh [11] when the sets boundaries are not precise, and the degree of belonging to a set is defined as a membership degree. Fuzziness includes imprecision, uncertainty, and degrees of the truthfulness of values. It is widespread in all areas where human judgement, evaluation, and decisions are essential.

Due to the very nature of fuzzy logic, it enjoys certain advantages making it suitable for the application under consideration: Fuzzy systems may employ any type of input, including erroneous, distorted, or noisy information. The architecture of the Fuzzy Logic System is simple and straightforward [12, 13]. Set theory mathematical notions are a prerequisite for fuzzy logic and may be justified relatively easily. Because of the fact that it resembles human reasoning and decision-making [14], it can offer a very effective answer to complex problems that may be found in every aspect of life. Eventually, algorithms may be stated with minimal data, utilising minimal memory.

The architecture of a Fuzzy Inference System (FIS) contains four parts, as shown in Figure.1.1. The identified components are:

- **Rules:** It governs the decision-making system that is based on linguistic information and comprises the set of rules as well as the IF-THEN conditions that were offered by the experts. Recent advancements in fuzzy theory have yielded a number of useful strategies that may be utilised in

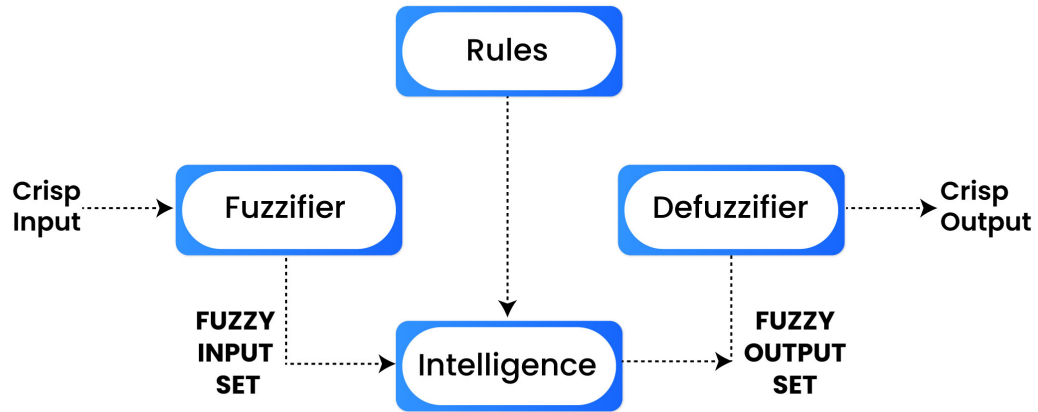


Figure 1.1: Architecture of Fuzzy Inference System.

the process of creating and optimising fuzzy controllers. The majority of these advancements lead to a reduction in the total number of fuzzy rules.

- **Fuzzifier:** It is used to convert inputs, such as crisp values, into fuzzy membership functions based on predefined fuzzy sets. Crisp inputs are sensor-measured inputs that are sent to the control system for processing, such as temperature, pressure, rpm, etc.
- **Intelligence/Inference Engine:** It determines the matching degree of each fuzzy rule to the current fuzzified inputs (firing strength). Next, the outputs of the fired rules are combined to form the control actions.
- **Defuzzifier:** It is used to convert the fuzzy sets obtained by the inference engine into a crisp value. Several defuzzification methods are available, and the best-suited one is used with a specific expert system to reduce the error.

As Fuzzy Logic (FL) depends on the level of truth, a Fuzzy Logic System (FLS) generates a specific output by using the degree of truth of the input and linguistic variables [15] to resolve complex situations involving ambiguous input data. FLS can become overworked when dealing with large amounts of data [16]. Hence deployment of the stand-alone FLS-based decision-making systems

is not a wise decision. The alternative to overcome the challenges is the employment of a distributed architecture. Although various approaches in distributed architectures for HAR exist, only a few of them use FLSs.

1.5 Distributed Architecture in HAR

As HAR is crucial in the health monitoring domain, the question remains of what type of architecture and where the architecture is to be deployed. Decisions making processes using FLS to analyse a huge amount of data which are imprecise and highly varying in nature, require high computational capacity, and are very much starving for resources. Thus a standalone system will not be suitable for the real-time processing of the data for HAR. Hence, a distributed system that communicates and coordinates their actions by passing messages from one component to another from any system must be deployed. Real-time performance enhancement and fault tolerance can be achieved through a distributed fuzzy system's ability to adapt to changes in the computing network during run-time. A standard solution to this problem is the deployment of cloud services.

1.6 Cloud Computing in HAR

With the growing popularity of cloud computing, it is obvious to use it to spur innovation and digitisation. It is possible to boost agility by using a cloud-first strategy for digitisation [17]. Cloud-based IT infrastructure resolves critical issues such as storage, computational resources, uninterrupted data access, and processing capabilities of a huge amount of data hindering greater data discovery and evaluation. Other advantages of cloud computing include decreased prices, improved security against external threats, and efficient usage of resources.

In cloud computing, the computing resources from providers are deployed as services (storage, computation, and communications), ready to be consumed by users. These services can be broken down into three main abstract layers: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). Over the last ten years, cloud computing has more and more shifted

the traditional on-premise paradigm towards an on-demand service delivery model carried out via the Internet [18, 19]. The key advantages of cloud computing, such as its scalability, on-demand availability, and pay-per-use pricing models, are rapidly turning it into a more popular alternative to costly on-premises systems.

Despite this, cloud computing still needs to be enabled by the standard methods that are used to construct FLSs. Thus, aspects such as flexibility, portability, elasticity, or the service discovery are very rarely considered by FLS designers. The proposal of Fuzzy-as-a-Service (FaaS), which inherits the stated cloud advantages, can thus be used as a natural extension of the FLS to act as a service in cloud computing platforms [20]. Realisation of FaaS can be achieved by implementing the FLS framework on commercial cloud platforms such as Azure and AWS, provides versatility, scalability, efficiency, and economical solution for analysing large amounts of data from sensors [21]. The cloud mentioned above services illustrates Service Oriented Architecture (SOA), a methodology for enabling software components to be reused and interoperable via standard interfaces. Services are readily integrated into new applications due to their standardised interface and reusable architecture [22, 23]. This eliminates the need for the application developer to redesign or replicate existing functionality and understand how to link or offer compatibility among existing functions. This removes tasks from the application developer who previously redeveloped or duplicated existing functionality or had to know how to connect or provide interoperability with existing functions. Hence design and development of an SOA over the cloud pose a solution for the difficulties faced in exploiting FLS in HAR under a distributed architecture.

1.7 Major Considerations in Implementation

The primary aim of HAR is the monitoring of the status of the health of a Person Under Observation (PUO) on a 24/7 basis. The required data can be gathered from the PUO using wearable gadgets. The characteristics of the collected data depend on many factors, including the physical and physiological status of the PUO. A high level of variation is observed in the collected data, and it makes decision-making a difficult task since no common rule can be applied. The amount

of data collected is huge, and storing them in standalone devices becomes very tedious. In addition, there is a specific MTBF (Mean Time Between Failure) during which the collected data are missed. The decision-making process based on the data collected requires a high computational capability of the devices under use. Therefore the challenges in implementing HAR using the distributed system are also high, as enlisted below:

- **Variability of Health Data:** The variability of health data is substantial as a result of the presence of various sources, including wearable sensors, medical devices, electronic health records, and self-reported information. The presence of a wide range of data types, formats, and attributes poses difficulties in terms of standardisation, integration, and interpretation.
- **Volume of Data:** The proliferation of digital health technologies has led to the generation of vast amounts of health data. Wearable devices, medical sensors, and health monitoring systems continuously generate streams of data, resulting in big data challenges related to storage, processing, and analysis.
- **Computational Cost:** Analyzing large-scale health datasets requires significant computational resources, including high-performance computing infrastructure and scalable data processing frameworks. The complexity of algorithms and the need for real-time or near-real-time analysis further contribute to computational challenges.
- **Availability of Decisions 24/7:** Ensuring the availability of decision-support systems based on health data round-the-clock poses logistical and technical challenges. Factors such as system reliability, data accessibility, and real-time response capabilities are critical for meeting the demands of continuous healthcare monitoring and intervention.
- **Integration and Interoperability:** Integrating heterogeneous health data sources and ensuring interoperability among disparate systems are essential for holistic patient care and effective decision-making. However, achieving seamless data integration across platforms, standards, and protocols remains a significant challenge in healthcare.

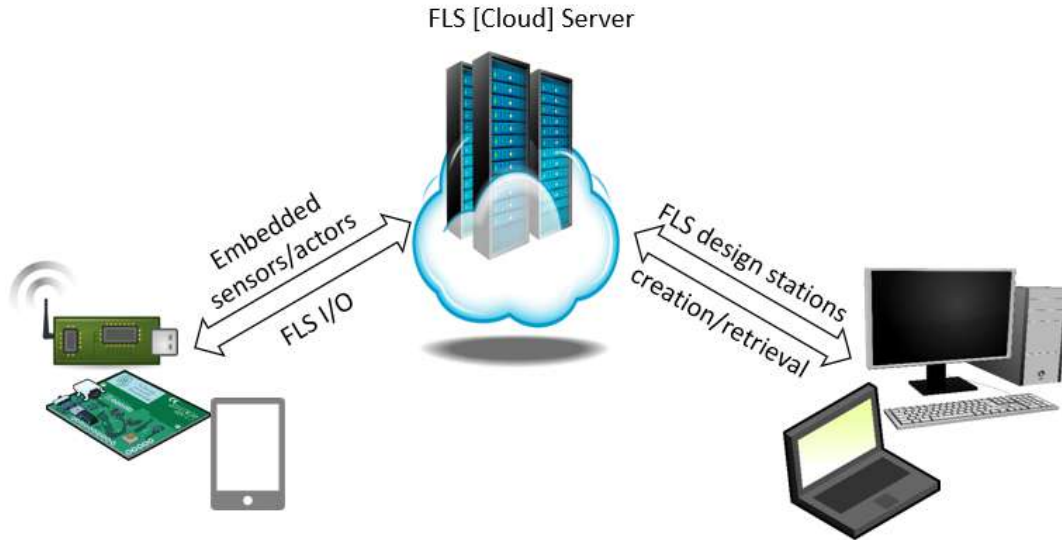


Figure 1.2: Schematic block diagram of distributed FLS.

Distributed fuzzy computing is implemented using a cloud-based architecture in order to address the challenges mentioned above. An illustration for designing the cloud-based architecture of FLS is shown in Figure.1.2.

For effective communication in a cloud-based FLS, selecting the appropriate web-based data language is a strategic criterion. Fuzzy Mark-up Language (FML), standardised as IEEE 1855 (2016) [24] an XML-based mark-up language most suitable for the human-readable and hardware-independent definition of an FLS, is exploited. FML and FML-compatible pieces of software, such as Java-based FML (JFML) [25] and Python-based library Simpful [26] are suitable for the development of FLS as per IEEE 1855 (2016) standard are used as the basic design libraries in this study, and the extensibility of this standard is a key solution proposed for the architectural growth.

1.8 Research Objectives

The major objective of the current research work is to exploit the technological powers of computation processing and decision making based on uncertain and highly varying nature of data using a multi-valued logic system like a Fuzzy

system and cloud technologies for developing a cloud-based distributed architecture for providing better handling of huge amount of data. The targeted deployment system will be health status monitoring using FaaS, a distributed FLS under a service-oriented architecture. The present research is based on the question: “Can a distributed fuzzy logic system provide any advantage in processing uncertain data over the current stand-alone and/or non-fuzzy systems considering HAR as a test case?” In order to address the answers to the above question, the following objectives are identified:

1. To conduct comprehensive research on existing tools and techniques and related research work to explore the possibilities of exploiting the powers of multi-valued decision-making processes like FLS.
2. To develop a service-oriented and web/cloud-based architecture for addressing the associated problems with uncertain data processing. This will involve exploring the applications and possibly extending the standard web communication protocols for fuzzy logic systems, i.e., IEEE 1855 (2016) and its associated software libraries.
3. To investigate the efficacy of the developed method in data processing for uncertain decision-making support scenarios considering HAR as a test case for a better health care monitoring system.
4. To apply the developed method for fall detection of the persons under observation using suitable sensors.
5. To measure the efficiencies of the proposed method in monitoring physiological (non-invasive) data obtained from a person.
6. To compare the performances of the developed architecture with other standard ML-based decision-making processes for HAR.
7. To compare the performances of the developed architecture with a stand-alone fuzzy system in realising the support system for HAR.

1.9 Research Approach and Design

The research approach is experiment-based research where data gathering, data processing, result analysis, and comparing results with various soft computing, cloud-based, and non-cloud-based systems are the main objectives to be achieved. The research design is based on the following activities:

Using a fuzzy system to handle uncertainties in HAR: Fuzzy systems have already shown their capabilities in learning from, processing, and predicting uncertain information. However, it has been limitedly applied in HAR, partly because of the computational complexities associated with high data volumes and/or data rates.

Proposing a service-oriented architecture (FaaS) specialised for HAR applications in order to overcome the computation problems: Recent developments in cloud computing, the internet of things (IoT), and SOAs have led to a shift from classical client-server models to service-oriented models. An FLS-specific software/hardware solution proposed based on SOA over the Web, named FaaS, will enable balancing the data collection, processing, and dissemination over the Web in order to achieve a higher processing power and a more manageable architecture than the classical stand-alone fuzzy logic systems.

In order to implement and test FaaS for HAR applications so as to fulfil the objectives stated, the research has been designed in a phased manner as follows:

- **Phase I:** A fuzzy logic-based HAR in both stand-alone and distributed is implemented using the data obtained from open data sources (e.g., kaggle.com).
- **Phase II:** Datasets are generated using a smartwatch with accelerometer and gyroscope for a fall detection scenario. The dataset is then fed into the designed FaaS.
- **Phase III:** The experiments performed during Phase II are repeated using various machine learning algorithms to ensure the effectiveness of FaaS against some other ML-based stand-alone systems.

- **Phase IV:** Another experiment is performed using collected physiological data such as blood oxygen saturation (SpO_2) and heart rate. Two cloud services, namely Microsoft Azure and Amazon Web Services (AWS), and two FML processing libraries are used to test. The considered combinations are:
 1. Mamdani FIE + JFML with AWS.
 2. TSK FIE + JFML with Azure.
- **Phase V:** This part of the work is dedicated to all types of comparison of the performances by various techniques such as Fuzzy stand-alone with fuzzy real-time using Azure and AWS, Machine learning with fuzzy real-time using Azure and AWS, and fuzzy real-time using Azure and AWS.

1.10 Relevance and Contributions

The following are the major contributions achieved:

- Identification of the challenges in dealing with uncertain, complex, and real-time data processing in standalone architectures while, particularly the limitations of developing classical FLSs in such applications.
- Development of the novel distributed architecture for fuzzy logic systems (FaaS), with a particular focus on their HAR applications, as a solution to address the identified challenges. The architecture can also be reused in other contexts beyond HAR, such as in time series prediction (e.g., for finance applications).
- Investigating the implications of the cloud services that would be used for implementing FaaS.
- Developing HAR test cases for evaluating FaaS performance compared to some non-FaaS solutions. This includes online and offline fall detection and health monitoring scenarios.

- Creation, and submission to IEEE for approval, of an extended schema of IEEE-1855 based on the newly found requirements.

1.11 Thesis Outline

As shown in Figure.1.3 this thesis is divided into eight chapters that outline the research work. The structure of the thesis is as follows:

Chapter 1 describes the background and technicalities of the research work carried out to provide the motivation and rationale for the study, progressing from general to specific and has been accomplished by establishing a research area and identifying a gap in that area. The study's purpose and significance are stated, and objectives are listed.

Chapter 2 aims to provide a comprehensive overview of current research work carried out by previous researchers in the selected area as well as in related areas, in addition to explaining the grounds for study in order to justify how the current research work adds to, contradicts, or augments the existing knowledge.

Chapter 3 is a tour of the theoretical background required to solve the research problems using appropriate justifications and assumptions. Furthermore, in this chapter, the framework of the new methodology developed and deployed is presented.

Chapter 4 provides the main components of design, data collection, and the software tools used for the development of a distributed architecture of a fuzzy logic system, including the use of FML and cloud for developing an SOA-based fuzzy system (i.e., FaaS) to be deployed for healthcare applications. It also presents the design and development aspects of fall detection methodologies in a distributed architecture using a service-oriented architecture. The results of using real-time data from the wearable accelerometer and gyroscope sensors as sensors have been presented and discussed.

Chapter 5 deals with the design and development of cloud-based FaaS for condition-based health monitoring based on data received for blood oxygen saturation and pulse rate via a wearable pulse oximeter sensor.

Chapter 6 presents the solution of human fall detection design settings, including the data collection, the non-fuzzy machine learning model, the FaaS, and then the performance measure and comparison methodology are presented.

Chapter 7: provides findings, implications, and recommendations based on analysis of the research. It addresses the issues raised in the introduction, evaluates the findings, and illustrates their underlying relevance. Lastly, suggestions for further broadening the work of future researchers in the same domain are provided.

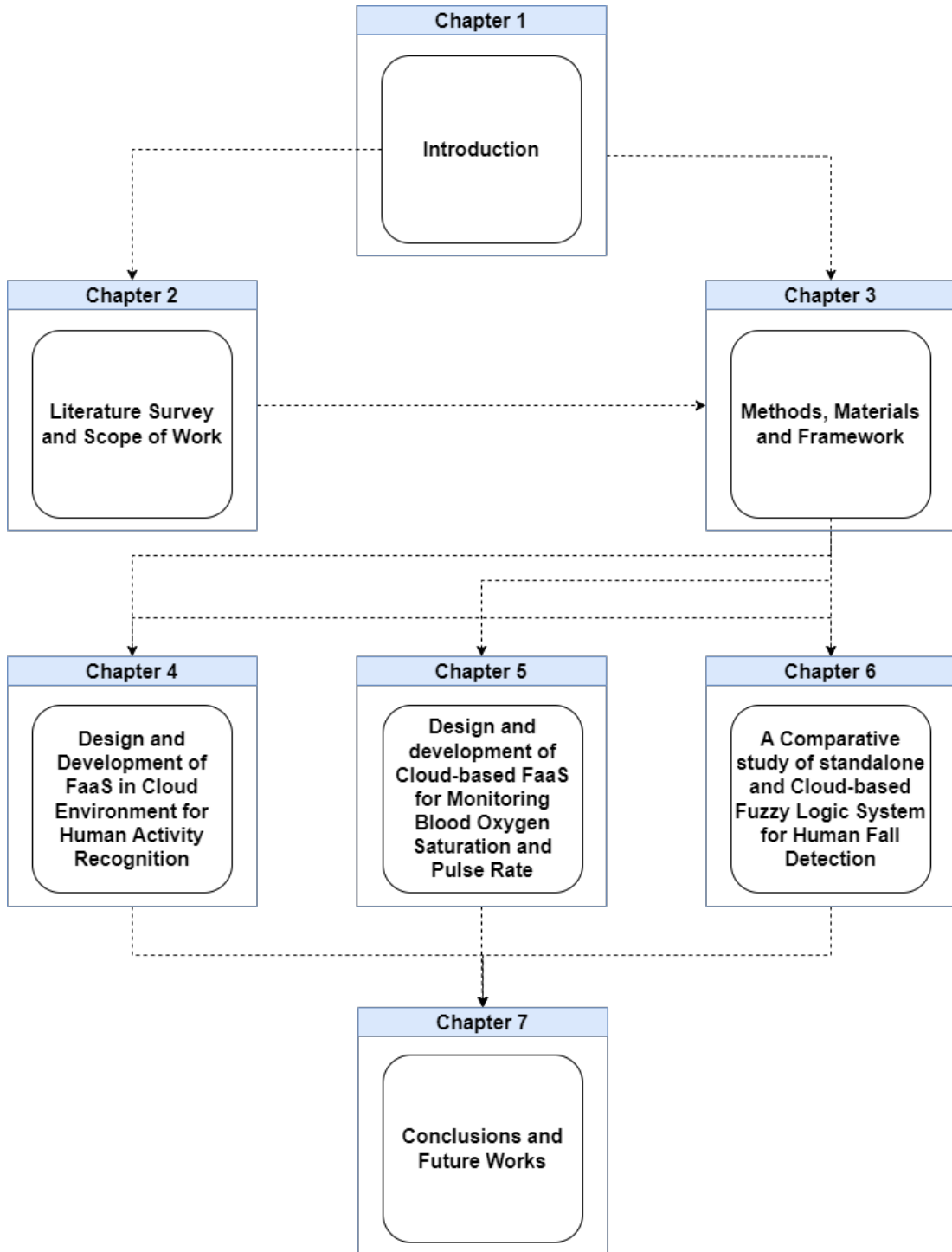


Figure 1.3: Thesis structure indicating the organization of the chapters and their respective dependencies.

Chapter 2

Literature Survey and Scope of Work

2.1 Introduction

This chapter presents state-of-the-art and a global overview of research challenges. This includes real-time distributed HAR with or without using fuzzy logic and/or cloud environment. Activity recognition systems are a significant area of research and development that are currently concentrating their efforts on developing more sophisticated machine learning algorithms, advancing the state of the art in terms of hardware architecture, and reducing the costs of monitoring while simultaneously increasing levels of safety [27]. HAR systems usually require 24×7 monitoring; hence a huge amount of real-time data is generated during the monitoring process. Also, standalone systems cannot handle such an amount of data with considerable speed of execution. Thus a variety of distributed systems for HAR are designed [28, 29], which will be reviewed in this chapter with a particular focus on fuzzy logic systems.

2.2 Distributed and Cloud-based HAR

This section presents a technical survey of recent literature on monitoring systems. The survey provides an exploration of contexts, technologies, and

2. Literature Survey and Scope of Work

existing approaches to figure out the further developments required to design and exploit a complete HAR in distributed and cloud-based formats.

Activity as a Service (Activity-aaS), a cyber-physical architecture that facilitates community, online, and offline human activity detection and monitoring in mobility, was introduced by authors in [30]. Activity-aaS can fill the requirement for Cloud-Assisted Body Area Networks systems and apps that make it easier for both people and communities to monitor and analyse human activities. Activity-aaS is based on the BodyCloud platform, which allows for efficient collecting of BSN-based sensor data, local processing (Body-side), high-performance computation, cloud storage (Cloud-side), workflow-based programming of analysis, and enhanced visualisation of findings (Viewer-side). It offers specialised, adaptable, and potent programming abstractions enabling the fast development of efficient and activity-oriented systems that support human activity. Several prototypes in step counting, physical energy calculation, automatic fall detection, and smart wheelchair support have demonstrated the proposed framework's effectiveness. By examining the processing load, data transmission time, CPU utilisation, memory footprint, and battery consumption on four diverse mobile devices, the performance evaluation of the proposed framework at the body side of the activity categorisation has been carried out. These gadgets are examples of low, medium, and high-performance mobile platforms.

Islam et al. [31] presented an activity monitoring and recognition framework based on a multi-class cooperative categorisation procedure to improve the activity classification accuracy in videos supporting the cloud computing-based blockchain architecture. In their approach, frame-based salient features were extracted from videos of different human activities, which were further processed into action vocabulary for efficiency and accuracy. Similarly, activities were classified using a Support Vector Machine (SVM) based on the Error-Correction-Output-Codes (ECOC) framework. They reported, using the experimental results, that the proposed approach was more efficient and achieved higher accuracy regarding human activity recognition than other state-of-the-art action recognition approaches.

An online activity detection framework called DOLARS (Distributed

On-line Activity Recognition System), created by Lupion et al. [32] in 2021, allows for the real-time evaluation of data from a variety of heterogeneous sensors, such as binary, wearable, and location sensors. A common feature vector was created by combining several descriptors and metrics from the heterogeneous sensor data, and its extraction was done in real-time using a sliding window technique. DOLARS offered a distributed architecture where a stage for processing HAR data was deployed in distributed nodes. Temporal cache modules computed metrics that efficiently aggregated sensor data for computing feature vectors. Publish-subscribe models were integrated to spread data from sensors and orchestrate the nodes (communication and replication) for HAR. ML algorithms were used to categorise and recognise objects. Results showed a promising level of performance in recognising activity sequences and demonstrated the necessity of distributed architectures for real-time recognition.

Table 2.1 summarises the different features adopted by different healthcare-monitoring-related studies in the literature.

2.3 Fuzzy Logic Systems for HAR

The very essence and important part of human activity recognition is to make the correct decision based on the data acquired from the person under observation using various sensors. Different ML techniques have been employed to obtain the correct decision. However, the nature and characteristics of the sensor data are full of uncertainties since it largely varies from person to person and is also environment-dependent. Hence, many authors have employed Fuzzy Logic either as a supportive technique or as a replacement for ML tools and have shown the effectiveness of such methods in activity monitoring systems. This section provides a review of some of the works reported in this aspect.

Brulin et al. [44] suggested a posture recognition system based on computer vision for geriatric fall detection in the home setting. The data collected by the strategically placed cameras served as the input for a decision-maker fuzzy logic system. The system was designed to recognise four static poses. However, it was unaffected by the subject's distance from the camera. They reported a posture recognition accuracy of 74.29%.

2. Literature Survey and Scope of Work

Table 2.1: Comprehensive Overview of HAR Features Explored in the Relevant Studies.

Feature	Type	Sample Studies (Ref.)
Monitoring mode	Local	[[33], [34], and [35]]
	Remote	[[36], [37], and [38]]
Transmission type	Cloud -based	[[37], [33], [34], and [38]]
	Device-to-Device	[[36], [39], and [35]]
Communication protocol	Wi-Fi	[[33], and [38]]
	Bluetooth	[[33] and [34]]
	Mobile cellular network	[38]
	Zigbee	[36]
Monitored activity/condition	Respiration	[[40], and [41]]
	Heart rate & SpO_2	[[33], and [34]]
	Body temperature	[[33], and [35]]
	ECG	[[36], [39], [38], and [35]]
	Blood pressure	[[42], and [43]]
	Patient position	[[35], and [6]]

2. Literature Survey and Scope of Work

Rashidpour et al. [45] proposed a HAR system based on an adaptive neuro-fuzzy inference system for learning and inference that uses smartphone motion sensors. The algorithm could discriminate between nine different daily activities and detect four types of fall.

Using a Takagi-Sugeno fuzzy inference engine with a triggering alert and a two-input Mamdani fuzzy inference engine, Kwolek and Kepski [46] developed a novel HAR method. The first Mamdani engine's output was a fuzzy set that assigned grades of membership to the possible values of dynamic transitions. The Mamdani engine's output was a second fuzzy set that assigned membership grades to potential body postures. It was found to be very helpful in fall detection, as demonstrated experimentally since a lesser amount of training data was typically utilised than in non-fuzzy counterpart systems. They also demonstrated experimentally that the proposed framework permitted reliable and unobtrusive fall detection in real-time at a low computational cost.

Jayalakshmi et al. [47] suggested a context-aware health monitoring system to track the physical and mental well-being of COVID-19-afflicted or confined elderly people. The framework was created with a fuzzy reasoning technique to use the event and medical context to anticipate or decide on the patient's health. The qualities are categorised using a fuzzy system based on patient activities. Different classification models, including Bayesian Network, Decision Tree, K-Nearest Neighbours, and Neural Network, were used to identify the patient's activities and medical history; however, the fuzzy adapted model produced the most remarkable accuracy. The suggested strategy significantly enhanced the reasoning engine's reliability, specificity, and efficiency.

A fuzzy system solution to the posture detection problem with regard to fall detection was offered by Kala et al. [48]. For the implementation, they presented a system with two sets of fuzzy rules, the first derived from domain knowledge and the second from rough set theory. They lowered the number of rules from 81 to just 44 in order to simplify the analytical process while retaining a high degree of classification accuracy. Two fuzzy inference methods and their aggregation were also considered, both with and without the knowledge measure.

2.4 Distributed and Cloud-based Fuzzy Logic Systems for HAR

Acceptance and use of fuzzy systems have increased and expanded into many fields, including control, electronics, and mechanics. Historically, the software used to build such systems has been derived from onsite tools, platforms, and languages. Yet the widespread adoption of cloud computing has introduced a game-changing approach to delivering IT services. Increased demand for cloud services has spawned new niche markets, such as data mining and machine learning[20].

The accelerated development, advances in cloud computing, and distributed architecture have also changed how HAR is approached. From the user to the health service provider, three layers of devices can be considered in the distributed HAR approach: sensor terminal, gateway terminal, and service (business) platform. The gateway terminal connects to the sensor terminal to receive physiological indicators and transmit them to the business platform. In addition to providing data, the gateway terminal may additionally receive health recommendations and other instructions from the enterprise system. The sensor terminal measures the user's physiological characteristics and structural data, such as their position, activity level, blood pressure, ECG, blood oxygen saturation, and heart rate. This type of classification is essential in community health care, where patients may be split based on their place of residence. Assigning the appropriate service physicians and customer service workers (nurses) is also a solution. The medical term allows users to obtain physiological and positioning data both indoors and outdoors. These transmissions are sent to the underlying health platform system across the mobile GSM-TD (Global System for Mobile-Time Division) communication network. The Web/WAP provides users access to prior health reports and medical experts' suggestions. The adoption of a cloud computing environment can dramatically lower storage costs as the volume of medical data gathered from the subject of observation increases, provide rapid access to the data stored there owing to the cloud's powerful processing capacity and enhance storage security. With the audit technology, the defined access control rules may eliminate superfluous operation

2. Literature Survey and Scope of Work

errors and unwanted operations while restricting the user's access. The stored physiological database can also be centrally checked for security aspects in order to observe the medical data privacy protection guidelines.

An example is a cloud-based fuzzy system for health monitoring was suggested by Thilagavathy et al. [49] to improve the effectiveness of health services in general. The proposed analysis was shown to be able to track the fitness risk related to blood pressure, heart rate, and kidney function. The fuzzy rule-based component of the fuzzy logic system was used to represent the deterministic parameters of interest for the subject's fitness. However, the study may exhibit several gaps and limitations that warrant attention. Firstly, the lack of comprehensive validation of the proposed method could undermine its reliability and accuracy in practical healthcare settings. Real-world validation studies encompassing diverse patient populations and medical conditions are crucial for assessing the robustness and generalizability of the FIS-based health monitoring system. Additionally, the scalability of the method on cloud infrastructure may not have been adequately addressed, raising concerns about its ability to handle increasing data volumes and user demands over time. Furthermore, the research may overlook critical security considerations associated with storing and processing sensitive health data in the cloud, potentially exposing patients to privacy breaches and cyber threats. Moreover, the absence of user acceptance testing and usability evaluation could hinder the adoption and effectiveness of the health monitoring system among healthcare professionals and patients. Addressing these gaps through rigorous validation, scalability testing, security assessments, and user-centered design approaches is imperative to enhance the reliability, efficiency, and usability of the proposed health monitoring method using FIS via the cloud.

A novel Android and cloud-based module easily adaptable to various hardware designs was also proposed by authors in [50]. To help novice users, the new module could automatically create executable files for Android and cloud-based (i.e., those without programming skills). The authors explained their innovative module using two case studies. Wearable accelerometers and gyroscope sensors were used to study real-time fall detection using fuzzy logic internet service providers. In their study, wearable sensors were used as an

2. Literature Survey and Scope of Work

example of how to monitor human behaviour using a rule-dependent fuzzy logic system. To discriminate between the occurrence of falls and non-falls, they created an algorithm with 90% accuracy, 88.89% sensitivity, and 91.67% specificity. However, several gaps and limitations in the study merit consideration. Firstly, the research may lack comprehensive validation of the proposed falling detection algorithm across diverse real-world scenarios and environments. Robust validation studies involving varied demographic groups, environmental conditions, and types of falls are essential to assess the algorithm's accuracy, sensitivity, and specificity. Additionally, the scalability of the algorithm to handle large-scale deployment and real-time processing may not have been thoroughly investigated. As falling events can occur unpredictably and require immediate response, evaluating the algorithm's performance under different workload conditions and resource constraints is crucial for practical application. Furthermore, the research may overlook potential security and privacy concerns associated with collecting and processing sensitive data from multiple sensors. Ensuring data confidentiality, integrity, and compliance with privacy regulations is paramount to protect individuals' privacy and prevent unauthorized access to personal health information. Moreover, the usability and user acceptance of the falling detection system may not have been adequately assessed. Involving end-users, such as caregivers and elderly individuals, in usability testing and feedback sessions can provide valuable insights into the system's effectiveness, ease of use, and integration into daily routines. Addressing these gaps through rigorous validation, scalability testing, security assessments, and user-centered design approaches is essential to enhance the reliability, efficiency, and usability of the proposed falling detection algorithm based on multisensor data fusion with SVM.

2.5 Data Exchange Formats and Standards for FLSs

A prevalent constraint arises from the absence of uniformity in data formats, resulting in compatibility challenges and complications in achieving interoperability among diverse cloud platforms and services. In the absence of standardised criteria, the methods used to communicate data can differ significantly, which can create difficulties when trying to seamlessly integrate and collaborate in diverse cloud settings. In addition, certain current methods may not possess adequate adaptability and scalability to suit changing data needs and various application situations. The inflexibility of these systems might inhibit the efficient processing and sharing of data, hence reducing the overall effectiveness of cloud-based systems. Furthermore, specific data formats may not sufficiently accommodate intricate data structures or semantics, hence restricting the extent and depth of information shared inside cloud infrastructures. The presence of these limitations emphasises the necessity for standardised, adaptable, and compatible data formats and exchange protocols to address current difficulties and unleash the complete capabilities of cloud-based architecture.

A significant challenge in constructing fuzzy systems is that there needs to be a standard for modelling this type of system. Although using FLS in HAR has the potential to be more efficient than others in distributed and cloud-based architectures, a major design issue is the need for a specific standard for web-based data exchange format. In a public use case of fuzzy systems, particularly for HAR applications, it is important to consider the principles of Explainable AI [51] termed as Transparent Fuzzy Control (TFC), meaning that "the joint use of fuzzy logic together with abstraction data tools, such as XML, able to allow the implementation of distributed intelligent decision-making systems independently from the details of hardware devices" [52]. Fuzzy Markup Language (FML), defined in IEEE 1855 (2016) [24], is used in this project as a unified and well-defined language for exchanging data about fuzzy systems. Moreover, its schema has been extended to fit the purpose of this project, which is recently submitted and is waiting for final approval by the

2. Literature Survey and Scope of Work

IEEE standard association committee IEEE SA-P1855. FML is an XML-based language used for data exchange specific to fuzzy logic systems. It is presented as a new approach to defining a detailed structure of the fuzzy rule-based system in an independent way from its legacy representation and the consequent hardware implementation, which makes it suitable for the targeted distributed architecture of the HAR systems in this project.

The use of FML enables system designers to swiftly convey their ideas, which, as a result, speeds up the process of developing any given complicated system [53]. This innovative representation of fuzzy systems can guarantee high interoperability of HAR network devices characterised by hardware heterogeneity, such as those composing a HAR framework. The design of a Semantic Web-based search engine is another objective of proposing FML because of being a markup language, which allows HAR designers to search for and find the best set of fuzzy inference engines (coded in FML) and download them to their devices' set so they can automatically configure themselves.

The preceding chapter describes fuzzy logic and its most well-known application, the Fuzzy Logic Controllers (FLCs), which are computational systems that emulate human decision-making processes by employing linguistic variables and fuzzy sets to model uncertain and imprecise information. FLCs have been implemented in a variety of technical and social, economic, and political domain scenarios. Nevertheless, despite their prevalence, the design activity of an FLC may be impacted by major challenges associated with the implementation of a given fuzzy system on various hardware architectures, each of which is characterised by a unique set of electrical/electronic/programming constraints. [54]. Particularly in a distributed, ubiquitous and pervasive system, the problem is more critical since different components of a fuzzy system can be implemented using different hardware/software configurations interacting with each other.

Most other language proposals focus on providing environments to model fuzzy systems for both general-purpose domains and specific fields. Prior to FML, FCL (Fuzzy Control Language, as a part of IEC 61131-2000) was introduced as an FLS definition language [55]. XFL [56] is another language specification for fuzzy modelling of general-purpose systems used on the Xfuzzy [57]. On the other

2. Literature Survey and Scope of Work

hand, XFSML [58], based on XML serves as a starting point for the definition of a standard modelling language. Due to their inability to describe data abstractly, these languages are unable to guarantee a sufficient enough level of hardware compatibility [59]. The current research work has investigated the possibility of integrating FML, as the chosen data exchange language, within the proposed architectural design for a distributed fuzzy system for HAR.

The object model used in FML for the description of an FLS is illustrated as a tree structure in Figure 2.1. This object model is encoded in an XML schema that can be found in [24].

In compliance with IEEE 1855-2016, JFML is an FML-based Java library to support designing and running FLSs [25]. JFML is an open-source Java library certified under the GPLv3 license. It adheres to an object-oriented technique and a modular architecture based on a similar tree structure used by FML. Four fuzzy inference systems (Mamdani, TSK, Tsukamoto, and AnYa) that are used in the basic specification of FML standard are supported by JFML. The authors give three case examples to demonstrate JFML's potential and the advantages of sharing fuzzy logic structures across different software. Designing a fuzzy logic system was the original case study for the well-known tipper regression problem. The second case study focused on creating a fuzzy logic framework to control a robot's tendency to follow walls. The early construction of a fuzzy logic classification system utilising the MATLAB fuzzy logic toolbox is the subject of the third case study. Since its early version, the library design was subsequently improved over the years.

An early application of JFML was designing an interoperability module for Arduino boards to construct and run fuzzy logic systems for embedded systems [60]. In addition, a communication protocol was built between the JFML and Arduino, which helped to reduce the embedded systems' limited computing capabilities. To show the capacity of the interoperability module, the authors analysed a wall-following fuzzy controller that manages a mobile robot in two settings according to IEEE Std 1855-2016. This enables developers to extend JFML without changing the grammar of the language. JFML facilitates using standard fuzzy inference systems that are present in XML schema definition, which includes membership functions and fuzzy operators. Other components

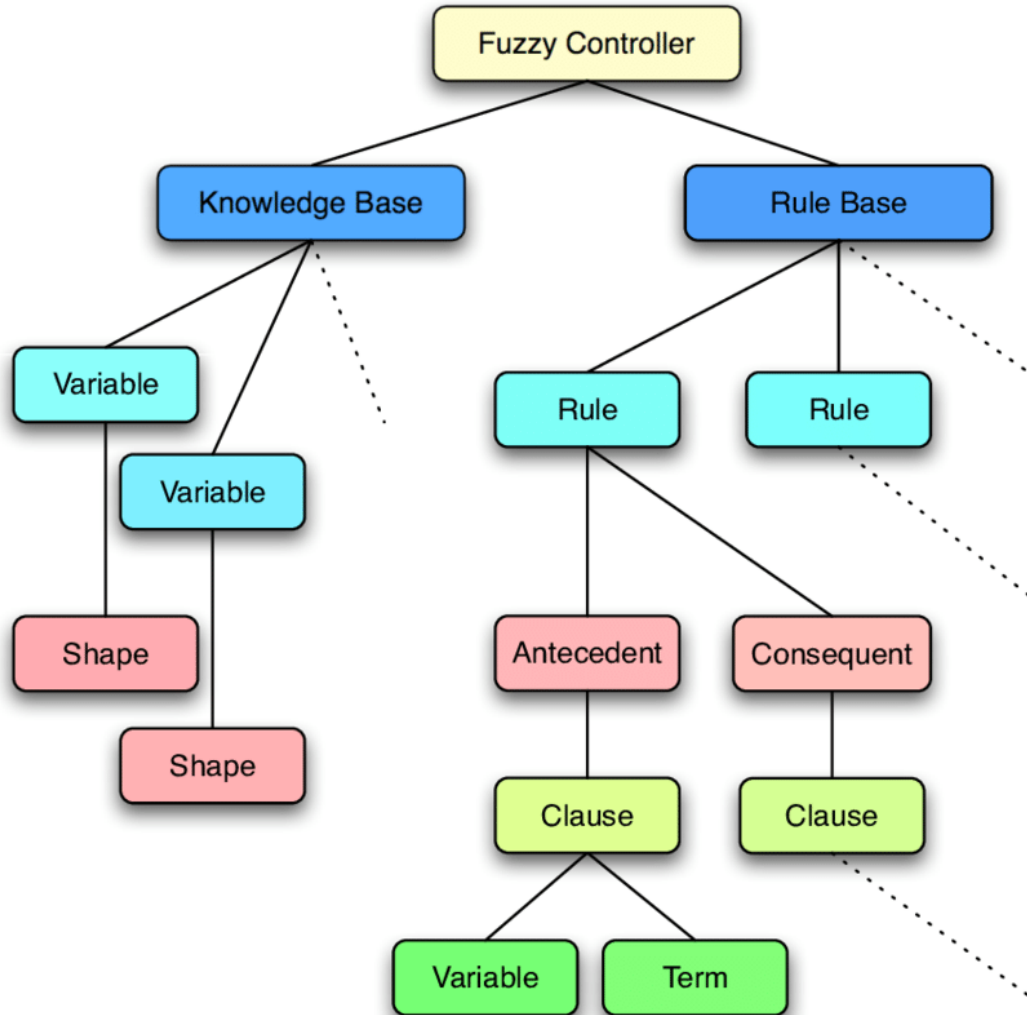


Figure 2.1: The object model tree structure of FML in defining a Fuzzy controller.

could be needed for further use by researchers; however, those components are not included in the current FML schema. Therefore, JFML makes available custom methods for all elements specified in XSD, offering a method to expand the library in fulfilment with this standard devoid of any changes in the grammar of the language.

Simplful is another library for FLS design that supports FML. It is a Python library that offers a complete implementation of the IEEE-1855 standard and has

2. Literature Survey and Scope of Work

the capability to import/export fuzzy systems in accordance with other standards too.

Table 2.2 summarises the major software tools, standards, and libraries for FLS design that can be used in a distributed FLS architecture.

Several other research works followed up the initial FML introduction and developing the associated libraries. Pourabdollah et al. [64] created web servers and suggested improvements to use extended FML to run fuzzy logic systems on a client-server basis. Their novel method involved integrating several components of the fuzzy logic paradigm into a single web server framework using HTTP requests/responses for web-based communication. The use of fuzzy logic structures in AmI contexts demonstrated the viability of this approach. Acampora and Vitiello [65, 66] showed how Arduino designs might create compatible fuzzy rule-based systems using the intrinsic expansion capabilities of IEEE Std 1855. Additionally, by avoiding addressing the hardware and software of a specific Arduino device, this feature allowed programmers to concentrate on abstract notions. A recommendation for a fuzzy-as-a-service solution. The three main objectives of their proposal were to describe cloud services for fuzzy systems using semantic technologies, to build services, and to use cloud computing models in cloud platforms to interact with other services [20]. VisualJFML, an improved JFML GUI-based visual framework, was created by Acampora et al. [67] to enable the modelling of fuzzy systems in compliance with the IEEE 1855 standard. Since it lets designers create shared fuzzy structures without any programming experience, VisualJFML provided significant feedback, according to their analysis. They demonstrated a user-friendly interface enabled by VisualJFML using a case study addressing the problem of iris classification. For open embedded hardware systems, Soto-Hidalgo [68] created a new JFML module that allowed programmers to create and use fuzzy ruled-based frameworks.

2.6 Scope of The Work

The major takeaway from the literature survey made as reported in the preceding sections are as follows:

2. Literature Survey and Scope of Work

Table 2.2: List of software that can be used to make FISs.

Sr. No	Libraries with Language	Ref.	Year	Description with Gaps
1	FML	[24]	2016	The two major issues across all applications are identifying contextual data from sensors in a generalised and effective manner and how contextual data is handled in order to improve service functions. FML exhibits limited functionality compared to code-based libraries, with usability hindered by the necessity to learn a markup language.
2	Juzzy Libraries Using Java	[61]	2018	A package or toolkit that simplifies the design and implementation of type-1, interval, and general type-2 FLSs. Limited language interoperability restricts juzzy to Java-based projects exclusively.
3	JFML Libraries using Java	[25]	2018	There will eventually be gaps in the internet of things, which will rely on FML's distributed processing capabilities. The JFML library, on the other hand, can take advantage of the situation by assisting fuzzy developers in performing scenarios in a more straightforward manner. Like juzzy, jfml encounters comparable restrictions regarding language interoperability.
4	JFML Libraries using Python wrapper on Java	[62]	2019	Membership functions, rules, and fuzzy inference in Py4JFML requires the use of some existing Python packages, such as learning and visualisation modules. The wrapper nature of jfml in Python adds complexities and limitations.
5	Simpful libraries using Python	[63]	2020	Tsukamoto and AnYa fuzzy logic operations and fuzzy inference methods are not supported in the current Simplful version. There is no support for type-2 FISs or stochastic fuzzy reasoning, either. The library's restricted language interoperability, tailored exclusively for Python projects, may limit its applicability

2. Literature Survey and Scope of Work

- The renowned fact is that FLSs, particularly for HAR applications, are usually implemented on dedicated local hardware and software systems, and most of these contemporary systems do not comprise distributed pervasive and scalable architectures.
- Contemporary progress in cloud computing and SOAs has caused a transition from traditional local or client-server paradigms to a service-oriented model for HAR.
- By changing the classical approaches to FLSs software/hardware architecture, a fuzzy logic system can be perceived as-a-service that, if maintained on the cloud, permits computing to be divested and concealed from devices in an intelligent environment, as well as being scalable, thus providing advantages over the other local solutions.
- An open-service model is very much in need in the healthcare domain. This openness can be accomplished by developing the system to be fully web-based and device self-sufficient, particularly by utilising standard formats for data exchange that are both consistent and readable as realistically as possible.
- A web-based data language for FLS characterisations is the key criterion for implementing such an architecture. The current standard for this purpose is the IEEE-1855 (2016), also known as FML, an XML-based mark-up language allowing the human readable and hardware-independent definition of an FLS. FML and FML-compatible software, such as JFML, can be used as the basic design standard in this study. The extensibility of this standard is a solution to architectural growth.

In Summary, the reviewed literature shows a research gap being the rarity of the standardised architecture adaptation for distributed FLSs for addressing the growing requirements in HAR. Based on this, the proposed solution is explained in the next chapter.

Chapter 3

Methods, Materials and Framework

3.1 Introduction

There is a need for a distributed and service-oriented architecture for FLS in order to unveil the potential of FLSs in effectively processing real-time and uncertain data in HAR scenarios. In this chapter, the details of the identified need and the details of the proposed architecture called Fuzzy-as-a-Service (FaaS) are provided.

In order to implement the monitoring system for HAR, the potential of FLS can not be overlooked. FLSs are intelligent systems that can approach the ability of the human mind to approximate vague data, extract useful information from them and infer a decision. Its potential can be applied to numerous domains, particularly to the medical field, to model the high complexity and uncertainty that characterises medical processes. An FLS provides the opportunity to transfer human and expert information into intelligent and automatic models by making use of linguistic terms. The knowledge transfer process begins with the construction of a knowledge base with the assistance of medical professionals and clinical experts. Since fuzzy logic permits the interpretation of data with preset linguistic variables in accordance with proper IF-THEN rules, fuzzy sets are used to treat uncertainty and describe knowledge by means of rules. Fuzzy sets are written as follows:

IF situation THEN conclusion

where the situation (premise) is called antecedent consisting of fuzzy terms connected by fuzzy operators and the conclusion is called consequent [9]. In the context of the HAR, fuzzy rules constitute the inferential engine of the FLS by defining the necessary inferential mechanisms for arriving at the output value related to the status of the persons under observation by using the physiological parameters obtained through sensors. This is accomplished by using the physiological data collected by the sensors. Human efficacy is said to stem not just from exact cognition but also from fuzzy concepts and reasoning. Fuzzy classification methods benefit over traditional methods since they produce a fuzzy decision or a value indicating the degree to which the data falls inside a range. Because it is based on natural language, fuzzy logic is becoming increasingly popular in the field of artificial intelligence for use in decision-making. It allows the natural description of problem domains in linguistic terms rather than in terms of relationships between precise numerical values.

The remaining sections of the chapter are organised as follows: Section 3.2 discusses the motivation for designing FaaS for human activity monitoring. Section 3.3 reviews the proposed FaaS architecture design, including the design attributes and the new software components. Section 3.4 explains an extension of IEEE 1855-2016 and API setup using the FaaS architecture for HAR applications; The implementation plan of FaaS in HAR will be explained in Section 3.4.3. Finally, Section 3.5 describes the summary of the chapter with an overview of the next chapter.

3.2 Motivation

Traditionally, fuzzy logic systems are linked to specific hardware or software systems. Observations reveal that dispersed and distributed designs of intelligent systems are gaining attraction. Due to the complexities of fuzzy logic computations, distributed architectures can add value to the development of fuzzy systems, especially in the field of HAR, where a large amount of data need

to be handled in real-time. Cloud computing is increasingly becoming a generalised alternative to expensive on-premise infrastructures with essential elements such as scalability, on-demand availability, or pay-per-use. Cloud computing is not supported by the conventional approaches taken to design FLSs. Hence, elements like flexibility, portability, scaling-up of resources in response to demand, and the discovery of services are not taken into consideration. Implementing, building, and deploying fuzzy logic models as a service on cloud computing platforms is a logical architectural extension for fuzzy systems, bringing several advantages, including scalability, development speed, portability, and interoperability.

The prime motivational forces to form the concept of FaaS in an AmI and distributed context include the distributed architecture of AmI, the requirement of high-power computation, reuse of computation results, scalability, openness and Accessibility, security, cost, and advances in cloud computing and IoT technologies [64]. Some of the aspects are specified below.

3.2.1 Openness and Platform Independence

This is a general motive not limited to AmI for some form of fuzzy logic system applications. It is noticeable that the tools available for computing fuzzy logic systems, e.g., some libraries/tools in MATLAB and R software tools generated for fuzzy logic computation, are typically fitted for a single purpose. When an independent architecture is developed, network connectivity to cloud service providers removes any need to install specific tools/libraries for fuzzy logic system computations[69, 70]. The platform-independent design is advantageous for academic and educational purposes, where usability and dissemination of both resources and outcomes are essential. Moreover, such an architecture leads to developing systems that can work in a cross-platform environment. The other benefit of developing a platform-independent architecture is that FLS computation is accessible from any web-connected device at any geographical location.

3.2.2 Configuring the Fuzzy Systems from Anywhere

Cloud computing can be regarded as a valuable extension of the client-server architecture because it makes allocating system resources more flexible based on the required computational power for a specific FLS. The fuzzy systems can be configured remotely and independently from their working environment. For example, the system rule base can be updated from any place if a new or updated set of experts' views are to be applied. This feature is of high importance for HAR scenarios since it requires minimum or no intervention with the living environment of the subjects under monitoring.

3.2.3 Reuse of Computational Results

To circumvent recurrent computations, a server can record inputs and produce output. For instance, if a specific device's inputs have been handled in the past, the stored outputs can be used as a reply through a lookup table. A single FLS definition for a specific application may be combined and reprocessed by several applications in dissimilar surroundings. Due to the distributed nature of inputs/outputs, this is only possible if FLS servers can be inquired for FLSs' definitions and past input/output descriptions within a distributed architecture.

3.2.4 Scalability of Fuzzy Logic System

Fuzzy logic systems can be extended further and merged with artificial intelligence to develop intelligent systems in an AmI environment with the help of a service-oriented architecture. This possibility arises from the well-known Zadeh's Incompatibility Principle, stating that when the complexity of a specific system rises, precise categorical statements about that system lose meaning, and meaningful statements cease to be precise and categorical. The linguistic approach used by fuzzy logic allows us to overcome this limitation and make fuzzy systems scalable to be embedded in complex frameworks such as AmI environments [71]. In these scenarios, fuzzy rule bases semantically related to different aspects of the controlled environment will be designed as pluggable into the components of the artificial intelligence ecosystem.

3.2.5 Need for a Standard Web-based Language

Scalability is an important concept, but it could lose many of its potentials if not accompanied by technology for data abstraction that allows the model of computing with words in a human-readable and hardware-independent manner [72]. As described in chapter 2, IEEE 1855-2016, also known as FML, is an XML-based language achieving the above goal, so that it will be a part of the proposed architecture. This abstraction language, introduced initially as a tool for modelling the behaviour of devices in smart homes, has been used in different application domains. The main benefits of IEEE 1855 are related to being directly convertible to programming logic in server-side programming to minimise the development time of this kind of architecture. This aspect is beneficial in Aml environments, where a collection of heterogeneous devices need to be opportunely programmed and communicated among them [73].

3.3 Fuzzy-as-a-Service: The Proposed Software and Hardware Architecture

Advances in HAR require decision-making tools and techniques to be accessible by a large number of networked devices, ranging from small and low-power sensors to embedded devices to PCs and large servers. The distributed architecture preferred for ambient computing systems implies that the sensors, processors, and output units can be physically distributed in the environment. FLSs are potentially considered to be computationally intensive, particularly when it comes to multi-input, multi-output, and multi-rule FLSs [74] or when it comes to higher orders and complex forms of fuzzy systems such as type-2 or non-singleton fuzzy systems [75]. Many designers avoid designing advanced forms of FLSs because of the increased complexity. While small or embedded devices, such as wearable or pervasive devices, are specially designed for networking and data communication, they cannot undertake intense computation. A standard solution for the case of non-mobile computer systems is to offload the complex logic from the clients to specialised servers in a client-server model. However, recent advances in cloud computing, IoT, and SOAs have led to a move from the classical client-server

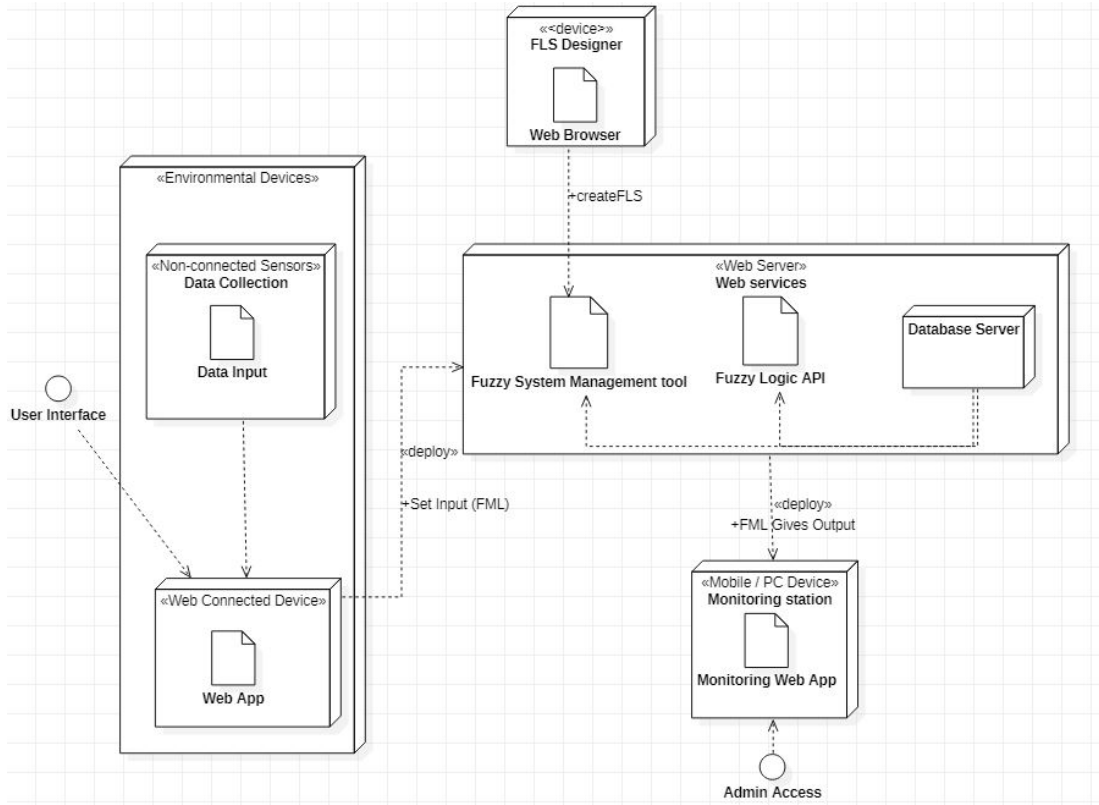


Figure 3.1: An illustrative diagram of client-server communication through the API.

model to a service-oriented model for ambient intelligence [76, 77]. By applying this approach to FLSs, fuzzy logic can be viewed as-a-service, called FaaS. If held in a cloud-based architecture, it enables computation to be offloaded and hidden from the devices in an ambient intelligent environment enjoying the advantages of cloud computing as well as handling large volumes of data at a time.

3.3.1 Architecture Components

The proposed system uses web-connected devices to develop a cloud-based service-oriented architecture for fuzzy logic systems. A sensor is a device to detects physical quantities and then converts them into equivalent electrical signals whereas web-connected devices widely known as IoT devices have the capability to get connected to the internet and can send data wirelessly to the

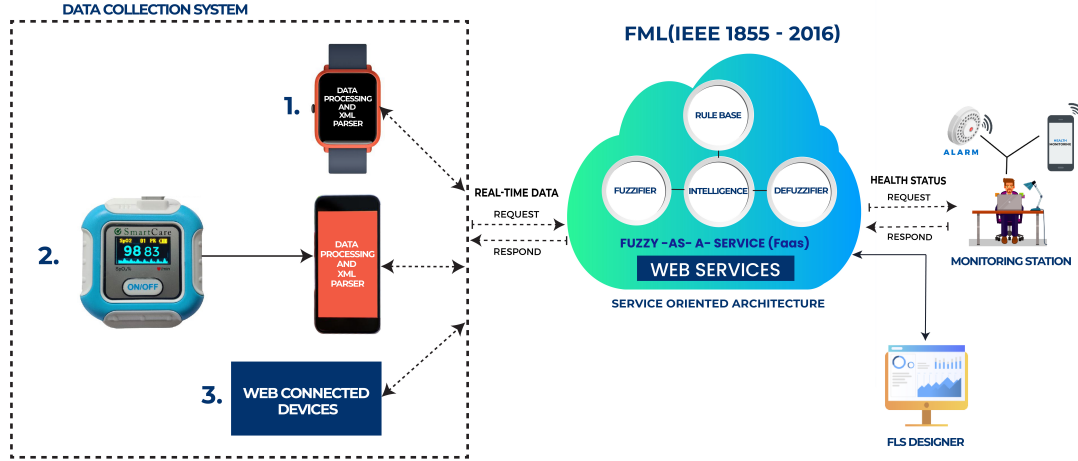


Figure 3.2: A schematic representation of the proposed system.

receiver. The significant benefit of using such sensors is that they can be connected to any device, such as IoT devices [78, 79]. The other benefit is that real-time data can be obtained from these sensors, which can be used for prediction purposes. Using public cloud services adds advantage to the FaaS architecture particularly when real-time data is obtained from the sensors and processed using a cloud service in HAR applications, which gives the desired results instantaneously.

The software/hardware components of FaaS include a fuzzy logic server(s), environmental devices/apps, management station(s), and monitoring station(s). The deployment diagram of the FaaS architecture is shown in Figure.3.1. An example schematic diagram of the architecture implemented in a HAR scenario is also shown in Figure 3.2. The architectural components are explained in the following sections.

3.3.1.1 Fuzzy Logic Server(s)

One or more servers are dedicated for processing calculations of fuzzy systems. Given an FLS definition in FML, the FML-based FLSs are generated, stored, and maintained. An FLS definition in FML enables the representation of the elements of a standard FLS using a labelled tree structure, in which an element, an attribute, and a text node represent each XML tag, XML attribute, and value

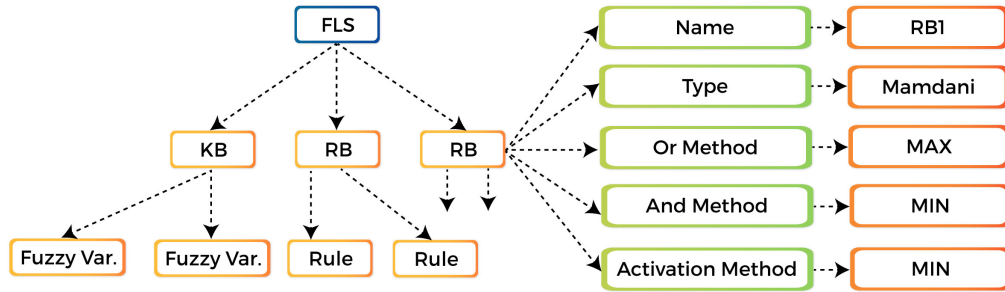


Figure 3.3: Example of FML labelled tree.

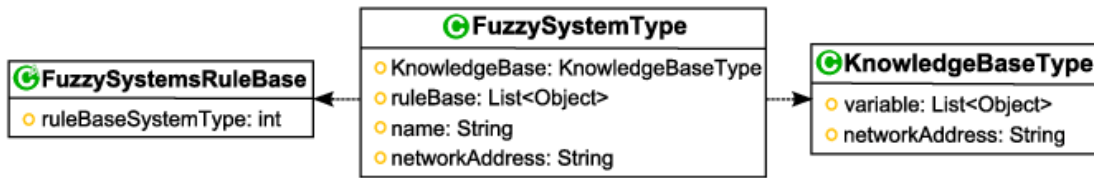


Figure 3.4: Main class diagram of JFML.

of an attribute. According to IEEE1855 and as depicted in Figure 3.3, the linkages of the tree describe the relationships between XML elements and attributes [24].

JFML and Simpful (explained in chapter 2) are two alternative software libraries responsible for FLS processing on the server(s). Figure.3.4 demonstrates the main class diagram for this library. The details of the JFML library are explained in Appendix B in order to provide the reader with a general understanding of the possibilities offered by the library.

3.3.1.2 Environmental Devices and Apps

The devices and apps to collect data from the environment or act on the environment based on the received data are parts of the FaaS architecture. These are the client-side devices and no FLS processing exists on them. They are assumed to be connected to the network and able to encode/decode their data in/from FML.

3.3.1.3 Management Station(s)

These stations allow authorised people to design/redesign FLSs remotely as per their requirements. People can add/remove the system, and anybody can use it at any time from anywhere using HTTP request and response. A web tool is designed to make it simpler for people utilising cloud computing. OAP, WSDL, and REST are examples of web service applications. The best way to use RESTful web resources is through the cloud. When a web service uses REST, it identifies limitations, such as a single interface that give rise to desired characteristics that allow services to work effectively on the web. In the REST architectural style, data and features are referred to as resources and can be accessed via URIs, typically web links. Using numerous well-defined operations, the resources will be used. An API consisting of an HTTP request and response configures a web service invocation for each feature.

3.3.1.4 Monitoring Station(s)

The outputs of the FLS calculations, along with the inputs are remotely available to the authorised people through the monitoring station(s). The data communication are in FML standard. This not only fits well to the HAR scenarios due to minimising the intervention into the living environments but also can offload possible post-processing tasks from the environmental devices or other architectural components.

3.4 Proposed Extension of IEEE 1855

IEEE-1855 basic schema provides a comprehensive mechanism for defining an FLS, however, the request and response features required in FaaS differ from what IEEE-1855 basic schema can offer. It is therefore necessary to predict mechanisms to send input/output data between the FaaS components.

The API data exchange in FaaS uses the core schema, IEEE standard 1855-2016, and the proposed extension. The extension is incorporated into the schema to facilitate input/output value exchange between clients and servers. The IEEE standards committee has recently approved the general framework of

this extension and is currently in the process of approving the details to be published in the next IEEE-1855 revision.

3.4.1 Need for Extending IEEE 1855 for FaaS

FML standard makes the information about fuzzy systems easily exchangeable over the web. The current FML standard can comprehensively “define” a fuzzy logic system. While applying FML for different applications, particularly for developing web-based systems, it is observed that once the system is defined by FML, communicating with the defined fuzzy system in the form of request-response is not yet standardized. For example, there is no standard notation for setting input values and getting the resulting output values for the defined fuzzy system. Furthermore, in some web applications, it is necessary to redefine, edit, query, or remove a fuzzy system from a set of defined systems, for which FML can also be extended.

3.4.2 Extension Proposal and API Design

Based on this, it is proposed that FML support not only a system definition standard but also a messaging standard (in which the current system definition by FML is the most detailed one of the possible types of messages). Therefore, adding notations to the new FML update is proposed to support a standard method of client-server messaging related to fuzzy systems. This will make the standard more general than just defining a fuzzy system, extending its applications to a wide range of Internet applications, including IoT and ambient intelligence domains [80].

The proposed added XML elements lie within two significant elements, i.e., FLSRequest and FLSResponse. This has been done intentionally and placed within the IEEE-1855’s root element (Fuzzy Controller) [24]. In other words, FLS Request, FLS Response, and their sub-elements are candidate extensions of the original IEEE-1855 schema. The other interpretation would be that all responses and requests from or to the server could be substantiated through a single schema. Combining the schema for the newly required elements with the standard IEEE-1855 schema results in an extended schema. The extended schema

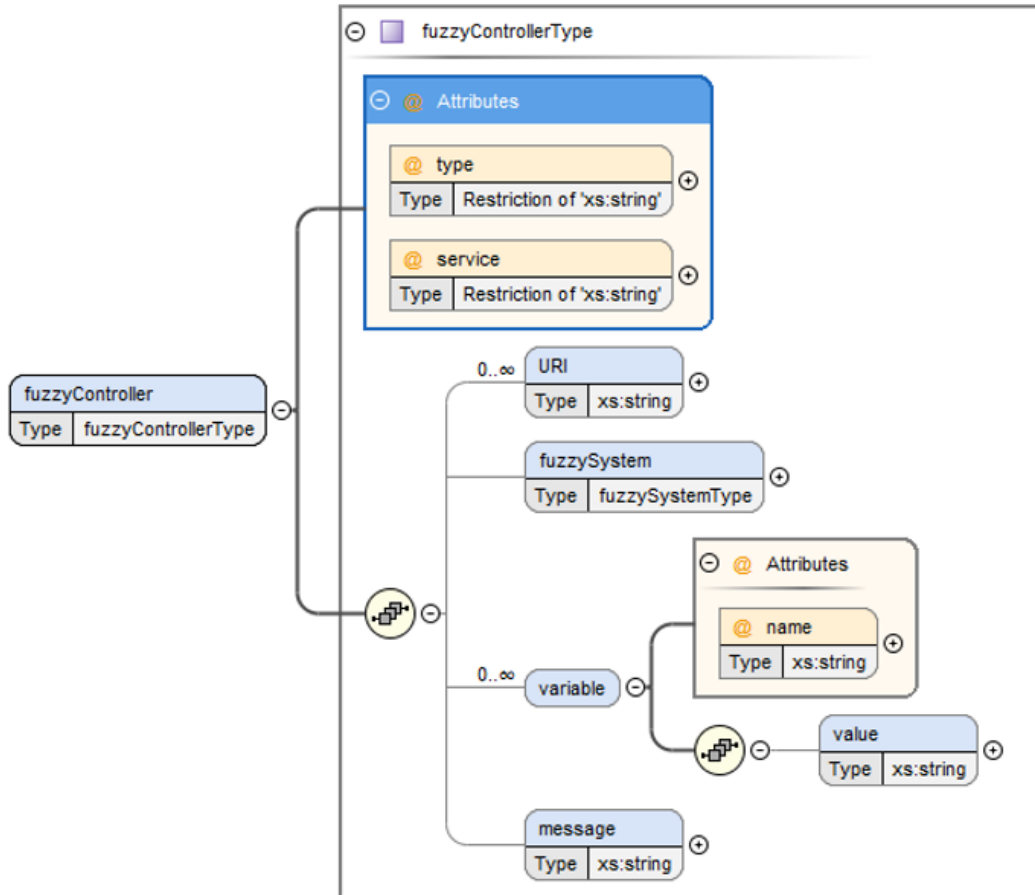


Figure 3.5: Schema of request/response the designed fuzzy-as-a-service API. The fuzzySystem element is the core element that follows the IEEE1855-2016 standard, whereas the other elements are to be considered as extensions for the Web services

would have the same root element as the original schema. Moreover, the original sub-elements, i.e., Knowledge Base and Rule Base [65, 68]. Would be added along with two additional complex-type elements for handling FLS requests and responses under the root element. The standard IEEE-1855 has been opportunely designed to enable a simple and direct extension to allow the original language to be adapted to different application scenarios.

In brief, a wrapper element is proposed to be defined above the current FML definition level. The sub-elements are proposed to be the desired services. Each

3. Methods, Materials and Framework

service has two data structures defined for either “request” or “response” types of messaging between a client and a server. For example, if a client wants to create a new fuzzy system for a server, a current FML data structure is wrapped inside a “request”-type message with the requested service as “createFLS”. Once the new system is created, the server responds to the client acknowledging the new system creation wrapped into a “response” -a typed message for the same service. Similarly, two other services called “setInput” and “getOutput” will be used to set the input values(s) and to get back the resulting output(s). A fuzzy system URI, currently existing as IP in FML, will be used to refer to each fuzzy system among different messages uniquely. Briefly, the “type” attribute determines if the HTTP packet is a request or a response. In contrast, the name of the requested functionality is encoded in the “service” attribute of both request and response packets. The request/responses messaging format is proposed for the minimum services defined in the table and schema graph shown in Figure 3.5. Table 3.1 defines the necessary parameters for request- and response-typed messages of each service. This is followed up by some examples of how to use the proposed services. Several methods accomplish the activities mentioned earlier. Such methods are listed in the following sections.

3.4.2.1 createFLS

The client machines generate a request that includes the FLS definition in IEEE 1855-2016 standard format, which contains information and values for fuzzy variables with a unique URI ID.

Create an FLS request consisting of URI and fuzzy system tags as input parameters and “createFLS” as a service attribute. The fuzzy system consists of two major elements, i.e., knowledge base and rule base. The knowledge base consists of fuzzy variables such as pulse and SpO_2 with values such as critical, alert, low, and normal. There are a number of rules in the rule base (considered as per the selected example), depending on which one can decide if a patient is COVID-19 critical or not.

Request:

```
<fuzzyController type="request" service="createFLS">
```

3. Methods, Materials and Framework

Table 3.1: request/responses messaging format for the minimum services.

service	type=request	type=response	
	parameters	parameters	Sample Messages
createFLS	URI	URI	System xx created successfully.
	System		System xx already exists.
	definition		Failed to create new system.
getFLS	URI	URI System	System xx retrieved successfully.
		description (FML)	System xx does not exist.
editFLS	URI	URI	System xx edited successfully.
	Edited description		System xx does not exist.
deleteFLS	URI	URI	System xx deleted successfully.
			System xx does not exist.
setInput	URI	URI	Input(s) set successfully.
	List of input names		Input variable xx does not exist.
	and values		Input is not validated.
getInput	URI	URI	Inputs retrieved successfully.
	List of input names	List of input names and values	Input variable xx does not exist.
getOutput	URI	URI	Outputs calculated successfully.
	List of output names	List of output names and values	Output variable xx does not exist.
			Input variables xx are not set.
listSystems		All systems' URIs	Systems list retrieved successfully.

```
<URI>Covid_FLS</URI>
  <fuzzySystem>...</fuzzySystem>
</fuzzyController>
```

Response:

```
<fuzzyController type="response" service="createFLS">
  <URI>Covid_FLS</URI>
  <message>System created successfully.</message>
</fuzzyController>
```

3.4.2.2 setInput

After the backend server has created the FLS file, values need to be set by the client in that fuzzy system for different variables so that the system can provide the desired output. The service “setInput” is used for this reason. Herein, the XML request service attribute is “setInput”, and the input parameter is a variable name. These crisp values are saved at the backend server so the fuzzy system can evaluate the result. The sample code is as follows:

Request:

```
<fuzzyController type="request" service="setInput">
  <URI>Covid_FLS</URI>
  <variable name="pulse">
    <value>50</value>
  </variable>
  <variable name="spo2">
    <value>80</value>
  </variable>
</fuzzyController>
```

Response:

```
<fuzzyController
  type="response" service="setInput">
  <URI>Covid_FLS</URI>
  <message>Input(s) set successfully.</message>
</fuzzyController>
```

The Client can repeatedly set data into the database, but the latest data can be used to evaluate the result.

3.4.2.3 getInput

This service is used to retrieve the latest information about variables. Values are retrieved for variable names provided in the request under the fuzzy controller type. A specific URI file is searched at the back-end server and tried to match the variable names passed in the request by iterating the entire file.

Request:

```
<fuzzyController type="request" service="getInput">
  <URI>Covid_FLS</URI>
  <variable name="pulse" />
  <variable name="spo2" />
</fuzzyController>
```

As shown in the sample code, the XML request service attribute is “getInput”. In the request, the unique “URI” is passed, and only those variable names whose latest information is required are taken into account. Values for matched variable names are sent back to the client, or the error response is “Input variables do not exist”.

Response:

```
<fuzzyController type="response" service="getInput">
  <URI>Covid_FLS</URI>
  <variable name="pulse">
    <value>50</value>
  </variable>
  <variable name="spo2">
    <value>80</value>
  </variable>
  <message>Input retrieved successfully</message>
</fuzzyController>
```

3.4.2.4 getOutput

The data input in createFLS and output in setInput is calculated in getOutput service. The service attribute used is “getOutput”, and in a request, the “variable name” and “URI” are passed. All possible values for the particular URI are checked against the system’s ruleset, and the result is then determined. If the

3. Methods, Materials and Framework

variable values are unsent using the “setInput” service, the output is “Input fields not set.” Also, if the variable does not exist at the fuzzy back-end system, the output is “Output variable does not exist”.

Request:

```
<fuzzyController type="request" service="getOutput">
  <URI>Covid_FLS</URI>
  <variable name="covid" />
</fuzzyController>
```

Response:

```
<fuzzyController type="response" service="getOutput">
  <URI>Covid_FLS</URI>
  <variable name="covid">
    <value>critical</value>
  </variable>
  <message>Output calculated successfully</message>
</fuzzyController>
```

3.4.2.5 queryFLS

Various clients may need access to information about the status of an FLS stored on the server. The stored FLS will be retrieved from the back-end server based on the URI passed in the request. The requested URI is searched in the system. If found, the XML is converted to a Java object and sent as a response, which includes the complete fuzzy system object (i.e., variable names, rule base, and knowledge base).

Request:

```
<fuzzyController type="request" service="getFLS">
  <URI>Covid_FLS</URI>
</fuzzyController>
```

The service attribute used is “queryFLS”, and only the “URI” is passed in a request. Suppose the requested FLS with a unique URI is not present. In that case, the response will be displayed as “System does not exist,” and if the URI is found in the database, then the server response will be “System URI ID retrieved

successfully”. The server response includes the FLS definitions mentioned earlier in the IEEE-1855 schema as createFLS and the latest value, which is set by using setInput.

Response:

```
<fuzzyController type="response" service="getFLS">
  <URI>FLS5</URI>
  <fuzzySystem>...</fuzzySystem>
  <message>System retrieved successfully</message>
</fuzzyController>
```

3.4.2.6 deleteFLS

Finally, clients should be able to request that an FLS be deleted from the database’s list of specified FLSs if the FLS description or past input/output history is no longer required. This service is used to delete the FLS generated using the “createFLS” service. In the deleteFLS service, the file at the backend with the given URI is searched. If the file is found, it is deleted, and a successful response is sent. However, if the file is not found, the error response “System does not exist” is sent back to the client. The service attribute used is “deleteFLS”, and the client needs to pass “URI” with the type of request to get a response from the server.

Request:

```
<fuzzyController type="request" service="deleteFLS">
  <URI>Covid_FLS</URI>
</fuzzyController>
```

Response:

```
<fuzzyController type="response" service="deleteFLS">
  <URI>Covid_FLS</URI>
  <message>System deleted successfully</message>
</fuzzyController>
```

3.4.3 FaaS Implementation on the Cloud

The proposed FaaS architecture is to be evaluated in the context of HAR. In this section, the methodological settings and plans for the implementation of the FaaS architecture in HAR scenarios (to be detailed in the next chapters) are provided.

3.4.4 Deployment Plan

The plan was deployed in the following phases.

First Phase: A comparison was made between Machine learning techniques and a Fuzzy logic-based system to explore the method's suitability for drawing the decision in health status monitoring.

Second Phase: This phase included the data collection using an Android App (Wearable sensor and Android phone), real-time conversion of data to IEEE 1855 standard (Cloud Version), and communicating the results in the extended FML.

Third Phase: Design of the FLS, Fuzzification, Fuzzy rule, Fuzzy Inference, Defuzzification (Android using JFML libraries).

Fourth Phase: Case studies of Fall detection and SpO_2 and pulse rate monitoring systems were developed using real-time data.

Amazon Web Service (AWS) and Microsoft Azure are used as two alternative cloud platforms for FaaS implementation. Android devices (such as smartphones and wearable devices) and ad-hoc sensors (such as oximeters) are used as environmental devices. Finally, web apps are implemented for monitoring and managing stations. The implemented apps are demonstrated in Appendix A.

3.4.5 Data Collection

During this research, various datasets were incorporated to evaluate the performance of the proposed approach. Initially, "Run or Walk" from

Kaggle.com’s public repository was utilized to evaluate the effectiveness of the proposed method. Six of the dataset’s attributes consist of accelerometer and gyroscope sensor data, each in three dimensions, and their corresponding timestamps. The dataset additionally labels each individual’s actual state, with a value of 1 signifying that the person is running and a value of 0 representing that the person is walking. The accelerometer and gyroscope data collection from a smartphone worn on the wrist has 88588 samples with a sampling rate of around 5 seconds. The additional details related to data collection and processing are explained in Chapter 4.

Research can be done more effectively if data is collected by the researcher as the researcher knows which type of data is required to evaluate the system’s performance. The BM2000A SmartCare Oximeter wrist-worn pulse oximeter sensor was used to capture human health monitoring data, namely SpO_2 in percent and heart rate in beats per minute. The sensor gathers heart rate and SpO_2 data at a rate of 10 milliseconds from a human fingertip. As soon as the fuzzy logic system receives the heart rate and SpO_2 data, it will progressively analyze them to determine the health condition. The additional details are described in Chapter 5.

The data collection system relied on wearable sensors like a three-axis linear accelerometer and gyroscope. Six users were surveyed for this study, two in the 20–25 age range, two in the 25–30 age range, one in the 30–35 age range, and one in the 60–65 age range. This information is laid out in six columns labelled Ax, Ay, Az, Wx, Wy, and Wz. Five distinct types of activity are used for data collection: walking, running, front fall, rear fall, and side fall. Information was gathered using a time series format. In this case, the sensor data collection rate was 419 Hz. The data collection process is explained in Chapter 6 and Chapter 7.

3.5 Chapter Summary

This chapter presents a novel approach toward a web-based service-oriented FLS architecture as an example of implementing the recently agreed IEEE 1855 standard. This chapter aimed to facilitate the flexible delivery of the relatively

3. Methods, Materials and Framework

complex computing required for FLS from clients to dedicated servers. The use of vitalised cloud services provided exceptional elasticity to the device. Fundamentally allowing universally accessible FaaS, other advantages of such architecture included network sharing, hardware/software control, data reuse, load balancing amongst FLS devices, and cost-efficiency.

Utilising FaaS for HAR in the cloud environment will be discussed in the next chapter.

Chapter 4

A Cloud-based FaaS for Fall Detection to Recognize Human Activities

4.1 Introduction

In the previous chapters, the contemporary progress in cloud computing, IoT, and SOAs that has caused a transition from traditional local or client-server paradigms to a service-oriented model for HAR has been reviewed. It is evident that the development of distributed architectures for FLSs is a relatively new field with very little progress. The previous chapter presented a proposal of a general architecture for changing the classical approaches to FLS software/hardware to a system that can be perceived as-a-service (FaaS). It was also shown that if the proposed architecture is maintained on the cloud, it permits computing to be divested and concealed from devices in an intelligent environment and scalable. Development and standardisation for the components of the underlying architecture and introduction of a fully-functional service-oriented solution to make it practical and cost-effective to design and implement complex FLSs, particularly in AmI environments, have been undertaken. Particularly two case studies are considered: the detection of fall of a person using the data obtained from wearable sensors and an overall HAR of

4. A Cloud-based FaaS in for Human Activity Recognition

the person under observation.

HAR offers numerous advantages across various domains due to its ability to automatically identify and classify human actions using sensor data and machine learning techniques. Here are some of the key advantages of human activity recognition:

Healthcare and Elderly Care:

- **Fall Detection:** HAR can be used to detect falls in elderly individuals and alert caregivers or medical personnel, enabling quick response and reducing the risk of injury.
- **Physical Therapy Monitoring:** HAR can track patients' exercises and movements during physical therapy sessions, ensuring proper technique and adherence to treatment plans.

Fitness and Sports:

- **Personalized Training:** HAR can provide real-time feedback on exercise form, intensity, and performance, helping individuals optimize their workouts.
- **Performance Analysis:** In sports, HAR can be used to analyze athletes' movements, enhancing training techniques and preventing injuries.

Human-Computer Interaction:

- **Gesture Control:** HAR can enable natural and intuitive interactions with computers, smartphones, and other devices through hand gestures and body movements.
- **Virtual Reality and Gaming:** HAR enhances immersive experiences by translating users' physical movements into virtual environments.

Security and Surveillance:

- **Intrusion Detection:** HAR can distinguish between normal activities and suspicious actions in security systems, improving intrusion detection accuracy.

4. A Cloud-based FaaS in for Human Activity Recognition

- **Anomaly Detection:** Unusual activities or behaviors can be identified, aiding in detecting potential threats or criminal actions.

Smart Environments:

- **Context-Aware Systems:** HAR contributes to building context-aware smart environments that adapt based on occupants' activities and preferences.
- **Energy Efficiency:** Smart buildings can optimize lighting, heating, and cooling based on activity recognition, saving energy.

Health Monitoring:

- **Sleep Tracking:** HAR can monitor sleep patterns and disturbances, helping individuals understand their sleep quality and make necessary adjustments.
- **Chronic Disease Management:** HAR can assist in managing conditions like Parkinson's disease by monitoring tremors and motor symptoms.

Safety and Industrial Applications:

- **Occupational Safety:** In industrial settings, HAR can identify risky activities and provide real-time alerts to prevent accidents.
- **Process Optimization:** HAR can analyze workers' movements to improve workflow efficiency and ergonomics.

Assistive Technologies:

- **Disability Assistance:** HAR can aid individuals with disabilities by enabling control over devices and interactions through movements and gestures.
- **Communication:** HAR can facilitate communication for people with speech impairments by translating gestures into text or speech.

Data-Driven Insights:

4. A Cloud-based FaaS in for Human Activity Recognition

- **Behavior Analysis:** HAR generates valuable data about individuals' routines, habits, and behaviors, which can be used for research and insights.

Automation and Efficiency:

- **Automated Workflows:** HAR can trigger automated actions or alerts based on recognized activities, streamlining processes.

In all these applications, HAR has the potential to enhance safety, convenience, efficiency, and overall quality of life. However, it's essential to consider privacy concerns, data security, and ethical implications when deploying HAR systems, especially when dealing with personal data and sensitive information. A fall detection system is a technological solution designed to automatically identify and alert caregivers, emergency services, or relevant personnel when a person experiences a fall. The primary objective of a fall detection system is to provide timely assistance to individuals who might be injured or unable to call for help themselves. These systems often incorporate sensors, algorithms, and communication components to achieve their purpose.

Fall detection systems offer several important advantages, especially in contexts where the safety and well-being of individuals are of concern. Here are some key advantages of fall detection systems:

Timely Assistance: One of the primary advantages of fall detection systems is their ability to provide timely assistance. When a fall occurs, the system can automatically alert caregivers, medical personnel, or emergency services, ensuring that help arrives quickly. This rapid response can significantly reduce the time a person spends injured or incapacitated without assistance.

Injury Prevention: Falls can result in serious injuries, especially among the elderly or individuals with certain medical conditions. Fall detection systems can help prevent injuries by enabling swift response and medical attention, minimizing the severity of potential injuries.

Independence and Quality of Life: Fall detection systems allow individuals, particularly seniors, to maintain a greater level of independence. Knowing that help is readily available in case of a fall can boost confidence and

4. A Cloud-based FaaS in for Human Activity Recognition

provide peace of mind, allowing individuals to continue with daily activities without excessive worry.

24/7 Monitoring: Fall detection systems operate around the clock, providing continuous monitoring and protection. This is especially valuable in situations where caregivers or family members cannot be present at all times.

Reduced Burden on Caregivers: For caregivers responsible for looking after elderly or at-risk individuals, fall detection systems can alleviate some of the constant worry. These systems serve as an extra layer of support, reducing the burden on caregivers while ensuring the safety of their loved ones.

Swift Medical Attention: Falls can lead to complications and medical emergencies. Fall detection systems can expedite medical attention, enabling healthcare providers to address potential issues sooner and improve the chances of successful treatment and recovery.

Customizable Alerts: Fall detection systems often allow customization of alert settings. Caregivers can receive alerts via phone calls, text messages, or notifications, ensuring they are informed in the way that suits them best.

Privacy and Dignity: Fall detection systems can operate unobtrusively in the background, preserving the individual's privacy and dignity. They do not rely on the person manually triggering an alert, which could be embarrassing or difficult in some situations.

High Sensitivity and Accuracy: Advanced fall detection systems use sophisticated algorithms and sensor technology to accurately distinguish between falls and normal activities. This high sensitivity helps reduce false positives and negatives.

Versatility: Fall detection systems can be integrated into various devices, including wearable devices, smartphones, and home monitoring systems, making them adaptable to different environments and user preferences.

Scalability: Fall detection technology can be scaled to accommodate multiple users or residents in care facilities, making it suitable for both individual and group scenarios.

Aid for Medical Conditions: Fall detection systems can be particularly beneficial for individuals with medical conditions that increase their risk of falling, such as Parkinson's disease, epilepsy, or certain neurological disorders.

4. A Cloud-based FaaS in for Human Activity Recognition

Data Insights: Some fall detection systems provide data insights into users' activity patterns and fall history. This information can be valuable for caregivers, healthcare professionals, and researchers to better understand and address fall-related issues.

Despite these advantages, it's important to consider the limitations of fall detection systems, such as the potential for false alarms and the need for user acceptance. Additionally, the effectiveness of a fall detection system depends on its accuracy, reliability, and the specific context in which it's deployed.

Falls are the main cause of susceptibility to severe injuries for humans, especially older adults aged 65 and over [81]. It is also one of the prime reasons for fatality among older adults and creates a barrier to independent living. Typically, falls are unnoticed and are interpreted as mere inevitable accidents [82, 83]. In certain countries, Personal Emergency Response System (PERS) facilities are developed for elderly people in case of falls, however, reports indicate that 80% of older adults are unable to stand after a fall and fail to use PERS to seek assistance [84, 85]. Therefore, an intelligent, automated, and precise fall detection system will be a substantial part of the daily living environment for elderly people to rapidly detect and provide fast medical responses [86]. Various fall detection systems involve using analytical data from images, videos, and audio, as well as data from inertial sensors such as accelerometers and gyroscopes [87, 88]. Various wearable fall warning devices have been created recently to ensure older people's health, however, most of these devices are dependent on local data processing [89, 90]. Moreover, several approaches are used to detect falls using machine learning techniques via human movement positional data to prevent accidents [48, 91]. The analysis is usually performed using wearable accelerometers and gyroscope sensors. In machine learning techniques, k-NN, decision tree, random forest, and extreme gradient boost are used to differentiate between a fall and a non-fall.

One of the main design objectives of fall detection systems is to alert all types of fall incidences, especially those related to Activities of Daily Lives (ADL) [87], thereby aiming to lessen injuries to the spine, head, or severe bone fractures. The main contribution of this chapter includes:

- A review on the performance of various fall detection machine learning

4. A Cloud-based FaaS in for Human Activity Recognition

techniques to detect a fall.

- An analysis of the performances of online and offline fall detection techniques specifically, FaaS utilising an online real-time approach and machine learning as an offline approach. Moreover, wearable sensors (i.e., accelerometer and gyroscope) that stimulate human activity monitoring using a rule-dependent FLS are developed and tested with real-time data.

In this chapter, the details of a novel experiment for evaluating the cloud-based FaaS for a human activity recognition scenario are presented. The subject of the experiment is to analyse the sensor data collected from an iOS device attached to a person's wrist in order to detect if the person is walking or running. An offline dataset is deliberately chosen so that different real-time data rates can be simulated in software. Moreover, the chosen training method can produce a very large rule base containing 240 rules, which can demonstrate the tolerated processing load and evaluates the system's limits in the higher sampling rates.

HAR using machine learning involves the development of algorithms and models that can automatically identify and classify different activities performed by humans based on input data, typically collected from sensors like accelerometers, gyroscopes, and sometimes other sources like images or audio. This technology has various applications ranging from healthcare to sports, entertainment, and more. It's important to note that the success of HAR models heavily depends on the quality and diversity of the training data, the choice of features, and the selection of an appropriate machine learning algorithm. This study uses fuzzy logic web services and machine learning techniques to perform a comparative analysis of real-time fall detection and HAR to determine the optimized approach that might work better for real-time fall detection and HAR. Accordingly, the cloud-based FLS has been designed and implemented using wearable sensors.

The innovation of this experiment is not to achieve better accuracy than the alternative machine learning methods, but to show the feasibility to shift from a local solution to an open-service model, accomplished by developing a fuzzy system to be entirely web-based and device self-sufficient, particularly by utilising standard formats for data exchange that are both consistent and readable as

realistically as possible. In FaaS, it is possible that an individual client computer may outline its necessary FLS, input data can be provided for the specified system by the same or another client computer(s), and lastly, the same or other client(s) can repossess the calculated output. For example, as will be shown in this chapter, requests from various clients can simultaneously be served in real-time by an FLS server, in which one sends the sensor data and the other one receives the person's status.

The remaining sections of the chapter are as follows: After explaining the experiment methodology in Section 4.3, Section 4.4 details the experimental settings of using the FaaS architecture for the focused HAR application. After providing the results in Section 4.6, Section 4.7 summarises the chapter with an overview of the next chapter.

4.2 Proposed System for Fall Detection

Figure 4.1 illustrates the general structure of the system proposed, containing three main steps, i.e., data acquisition 4.2.1, data processing 4.2.3, and feature extraction of fuzzy data. Specifically, data from the accelerometer and gyroscope are continuously sampled and stored prior to the fall event.

4.2.1 Data Acquisition using Wearable Sensors

Data from the accelerometer and gyroscope are continuously sampled and stored before the fall. The sensors, namely the linear accelerometer, and gyroscope were used for data collection. The wearable sensors used for data collection were part of a smartwatch. The data acquisition was developed using wearable sensors considering a three-axis linear accelerometer and gyroscope. Data were collected from a total of six users, i.e., two users between 20 and 25 years, two users between 25 and 30 years, one user between 30 and 35 years, and another user between 60 and 65 years old. Data contains 6 columns, i.e. A_x , A_y , A_z , W_x , W_y and W_z . Five classes are used for data collection: walking, running, forward fall, backward fall, and side fall. The sensor data collection rate was set to 419 Hz. Each individual performed the activities mentioned above once for the five

4. A Cloud-based FaaS in for Human Activity Recognition

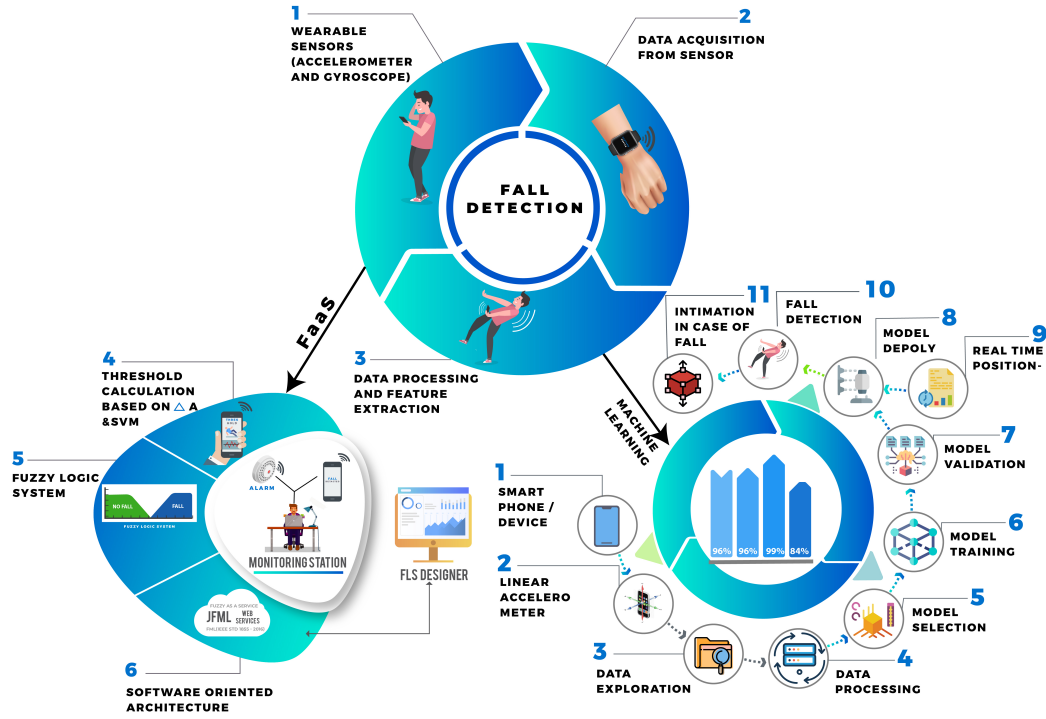


Figure 4.1: System architecture of a fall detection system using both ML and FaaS.

classes. The trends of fall/non-fall activities are in the place that should also be considered in accordance with time. As shown in Figure 4.2, it can be observed that data trends differed over time and were close to 0 in the event of a fall.

4.2.2 Data Diagnosis

Diagnosis of data helps analysing the nature and pattern of a fall as it provides an in-depth discernment. Positional data ensures continuous fall distribution, so analysing data that follows a normal distribution is invigorated. Figure 4.3 shows a normal distribution in case of a fall. Sharp peaks at 0 can be observed, indicating a fall's nature. From the figure, it is evident that data tends to follow a normal distribution in the case of a non-fall. For each class, data frequency in case of a fall and non-fall was checked, and assessed via a histogram plot shown

4. A Cloud-based FaaS in for Human Activity Recognition

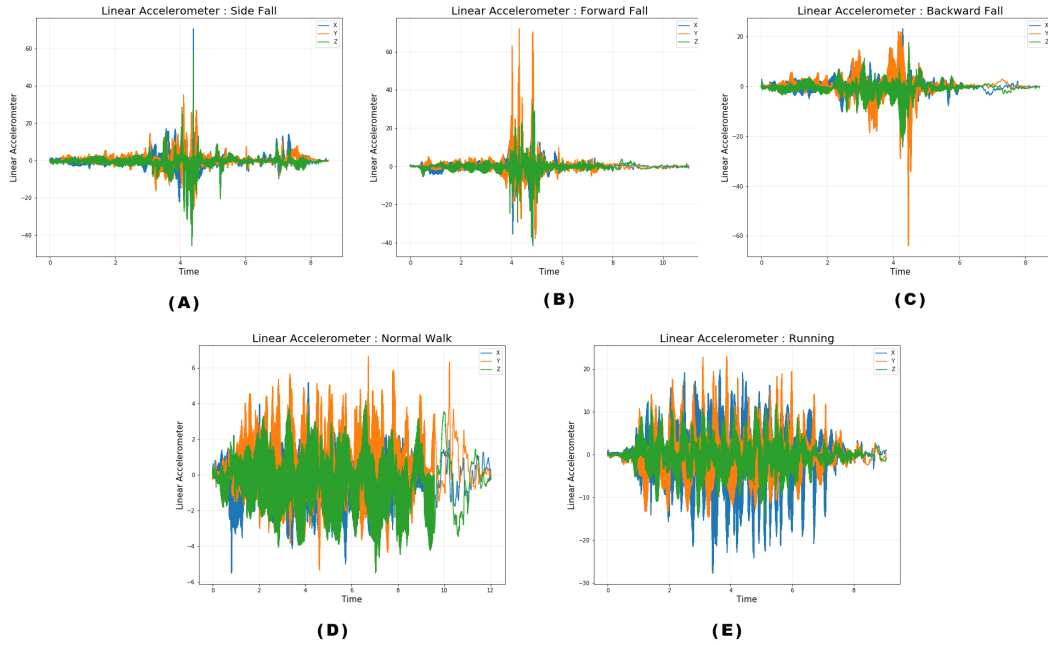


Figure 4.2: Trend Plot for fall and non-fall activities: (a) Side Fall, (b) Forward Fall, (c) Backward Fall, (d) Normal Walk and (e) Running.

in Figure 4.4.

From Figures 4.3 and 4.4, it is evident that data frequency at position 0 is very high (for example, forward fall, backward fall, and side fall), whereas in the case of a typical scenario, data frequency varies at different positions (for example, normal walk and running). After analysing the data distribution through normal distribution and histogram plots, data visualisation in three-dimensional (3D) space was needed. It is apparent that in the case of a fall, data is plotted based on a fall's direction, whereas in the case of a normal scenario, positional data is being concerned. 3D visualisations facilitate the depiction of high-dimensional data representation. It is essential to examine the positional data patterns throughout time. In a typical setting, it was noticed that positional data fluctuates over time and approaches 0 in the event of a fall. As already observed in normal distribution and trend plots, in the case of a fall, positional data ranges become close to 0, whereas, in the case of a normal situation, positional data varies intensely.

4. A Cloud-based FaaS in for Human Activity Recognition

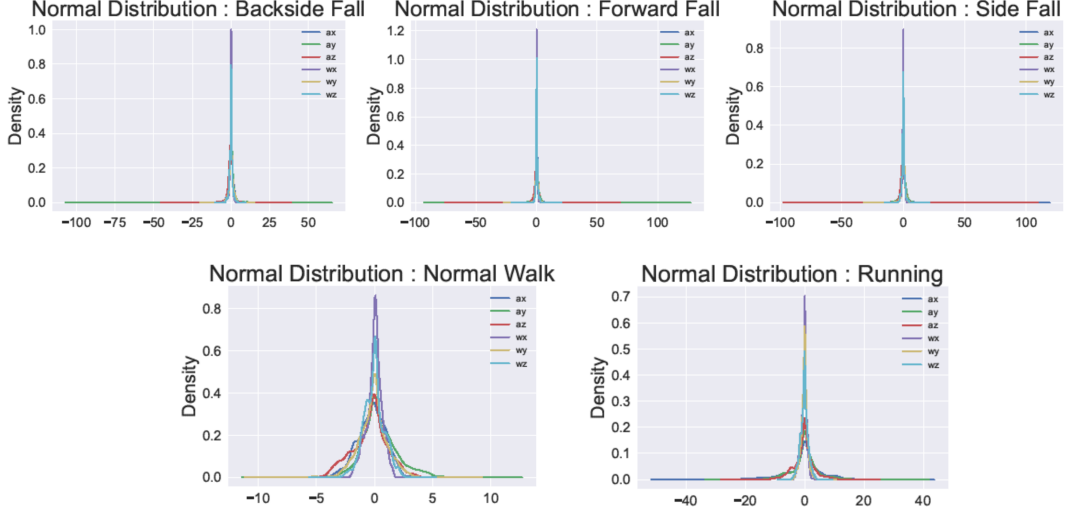


Figure 4.3: Normal distribution of the obtained accelerometer and gyroscope data

4.2.3 Data Processing

According to the method used in [92], two outputs can be obtained from the stage of data processing, including the magnitude of the linear accelerometer (SVM_A) and the acceleration difference (ΔA). The method of calculating SVM_A is defined as:

$$SVM_A = \sqrt{A_x^2 + A_y^2 + A_z^2} \quad (4.1)$$

where the linear accelerations in the x, y, and z directions are described by A_x , A_y , and A_z , respectively. Moreover, the magnitude value exceeding 6.0 indicates the possibility of fall occurrence. The program will compile 1000 data samples, gathering data on typical user behaviours before and after the occurrence. Before the fall, the first 950 data samples (expected to take 1.5 seconds) and 50 data samples will be gathered from the current time step. 950 data samples (estimated duration 1.5 seconds), and the following 50 data samples will be taken after the fall. The average of the 50 samples taken before and after the fall is being considered. The process of choosing data samples is illustrated in Figure 4.5.

The ΔA of the 950 data samples collected following the incident is used to perform the first analysis. For calculating the ΔA , the XYZ accelerometer will

4. A Cloud-based FaaS in for Human Activity Recognition

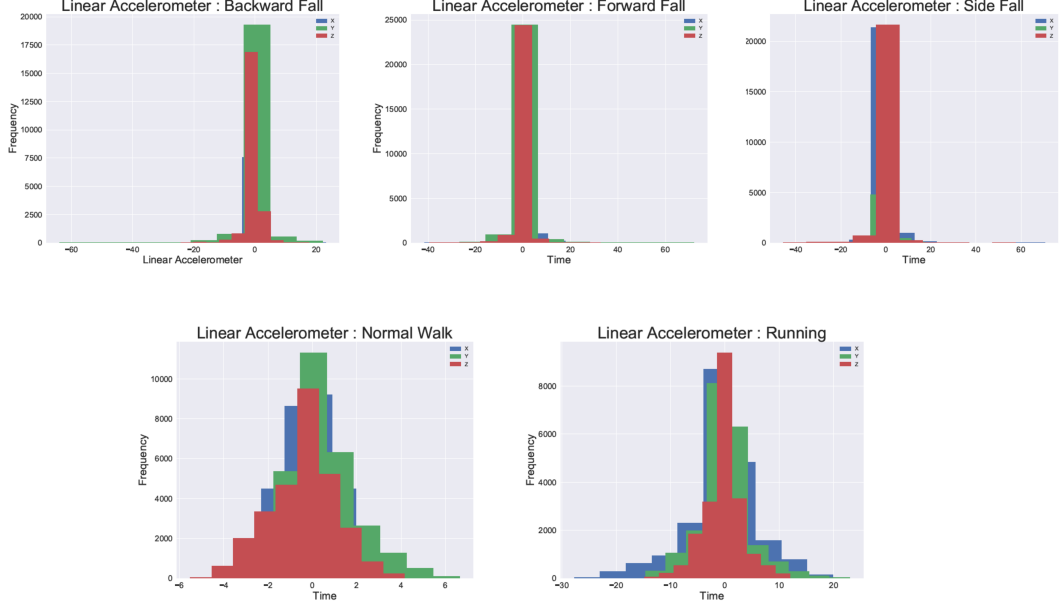


Figure 4.4: Histogram plots of the obtained accelerometer sensor

have both start and end coordinates [92], which is shown as:

$$\Delta A = \cos^{-1} \left(\frac{(A_{xS} \times A_{xE}) + (A_{yS} \times A_{yE}) + (A_{zS} \times A_{zE})}{\sqrt{(A_{xS}^2 + A_{yS}^2 + A_{zS}^2) \times (A_{xE}^2 + A_{yE}^2 + A_{zE}^2)}} \right) \quad (4.2)$$

where average accelerations along all three axes are denoted by the notation A_{xS} , A_{yS} , and A_{zS} . In contrast, average accelerations along all three axes at the endpoint are denoted by A_{xE} , A_{yE} , and A_{zE} . Since hand rotation is commonly utilised in situations involving falls, the data acquired by a gyroscope may be used to estimate the device's angular speed in three dimensions (degrees per second). This is possible because of the nature of the gyroscope. The unit can move in any direction during the fall; because of this, the directionless SVM_G takes into consideration the unit's potential to move in any direction. According to [92], the amount of rotation, which SVM_G represents, is calculated by making use of:

$$SVM_G = \sqrt{G_x^2 + G_y^2 + G_z^2} \quad (4.3)$$

where G_x , G_y , and G_z are the speeds of rotation along the X, Y, and Z axis,

4. A Cloud-based FaaS in for Human Activity Recognition

NUMBER	TIME	WX	WY	WZ	AX	AY	AZ	GMAGNITUDE	AMAGNITUDE	SVM	DELTAA	STATUS
1612	2.556	-0.45	0.03	-1.52	1.71	-0.15	-0.84	1.5854968	1.9110731	150	0	NOT FALL
.
1662	2.636	-0.27	-0.96	-1	1.41	-0.69	-1.67	1.4122677	2.2919643	250	0	NOT FALL
.
2562	4.073	0.31	-3.62	4.79	-18.37	-19.27	-6.72	6.0120378	27.458118	400	79.32046	FALL
.
3510	5.58	-0.18	-0.28	0.06	0.63	-1.58	0.78	0.3382307	1.8712829	150	0	NOT FALL
.
3560	5.657	0.13	0.32	0.03	-0.16	0.02	0.13	0.3466987	0.20712315	150	0	NOT FALL
.

Figure 4.5: Selection of a data sample.

respectively. According to Equation 4.3, the value of SVM_G is obtained that is used for the classification of the monitored individual's status within the FaaS.

A fall can be typically viewed in three states: zero gravity, state of impact, and inactivity. The accelerometer and gyroscope can check a sudden shift in acceleration in the zero gravity state. The impact states are simple methods for predicting the event of a fall. Similar to ADLs, hand clapping can easily trigger false alarms for wrist-wearing applications. During a fall, the flipping and spinning of hands might be critical. Nevertheless, two simple criteria that distinguish falls for different people from all fall ADLs, particularly for the presence of constantly moving hands, are almost impossible to determine.

4.2.4 Fuzzy Logic System Design

The fuzzy logic system suggested in [93] takes the inputs over and takes several steps to generate vagueness-dependent output in fall detection. Fuzzification turns signals into membership degrees within fuzzy sets defined for low, medium, and high input signals. The system is a dual input that contains ΔA and SVM, i.e., low, medium, and high ΔA values and low, medium, and high SVM_A values. Memberships in a scheme are drawn, and every member is classified as turning points with different ΔA values choosing three turning points, i.e., 20° , 45° , and 90° . Falls usually occur in 90° , but some falls can exceptionally occur in less than 90° in which the wrist-worn sensor cannot determine an exact ΔA .

4. A Cloud-based FaaS in for Human Activity Recognition

Figure 4.6(c) shows the input fuzzy sets and Figure 4.6(c) shows the output fuzzy set, in which all the three memberships functions have a triangular response with an average scale from 0 to 60, and 10, 30, and 50 were the value of output, respectively, and were allocated low, medium and high output.

The following steps are carried out to develop the FLS:

- Input fuzzy sets: SVM is the first input that contains three values, i.e., low, medium, and high.
- Compounding: Set a minimum angle, leading to a 45° fall, and see it as a medium angle. The lower and extreme angles are 20° and 90°, respectively. The size of all 0° to 180° memberships was calculated by the minimum and maximum angles that the sensor may calculate. If the angle is greater than 45°, the accident is more likely to be called a collision.
- Rule base: To conduct this experiment, a total of nine rules, are listed in Table 4.1, was developed to determine whether or not it is a fall.
- Output fuzzy set and defuzzification: Similar to the input given to the system that are transformed into three fuzzy sets, the system's output offered three values, i.e., low, medium, and high. After the inference, the centroid method is used for defuzzification which transforms the output fuzzy set into a crisp value at each time step.

The designed FLS has been adapted and implemented under the proposed FaaS architecture. Figure 4.7 shows the flowchart of the proposed fall detection system under FaaS.

4.2.5 Modelling Approach using Machine Learning Techniques

The generated dataset consists of positional data accompanied by response variables. A response variable is a multi-category variable with labels: backside, forward, side falls, normal walk, and running. The dataset has been normalised so that each answer variable has the same number of instances to exclude the possibility of a bias factor being incorporated into the model. This was done to

4. A Cloud-based FaaS in for Human Activity Recognition

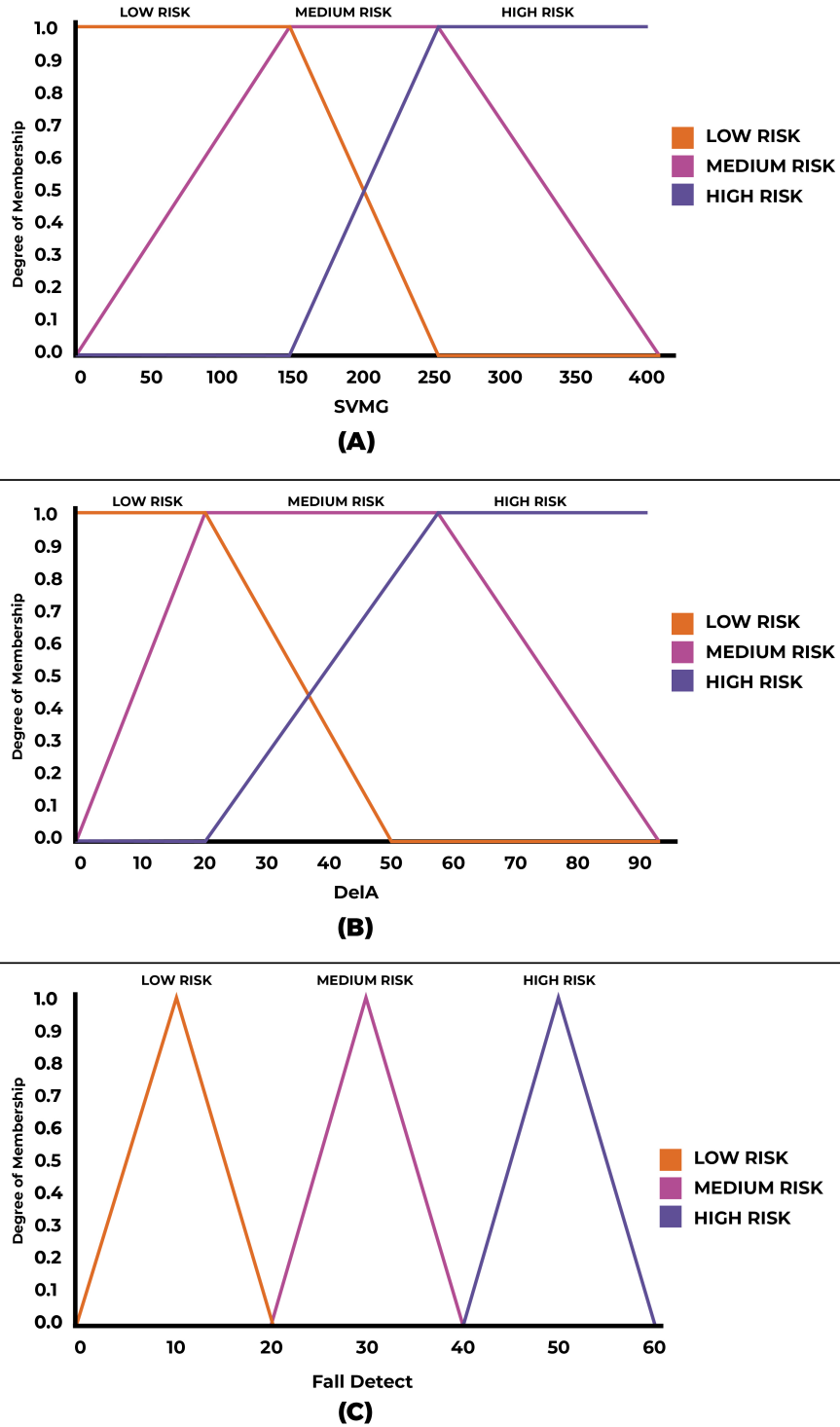


Figure 4.6: Membership for fuzzy inputs:(a) SVM_G (Degree/s) (b) ΔA (Degree) and membership for output (c) Fall Detection.

4. A Cloud-based FaaS in for Human Activity Recognition

Table 4.1: Rules for FLS to detect the risk of fall.

Rule Number	Input		Output
	ΔA	SVM_G	
1	Low Risk	Low Risk	Low Risk
2	Low Risk	Medium Risk	Low Risk
3	Low Risk	High Risk	Low Risk
4	Medium Risk	Low Risk	Low Risk
5	Medium Risk	Medium Risk	Medium Risk
6	Medium Risk	High Risk	High Risk
7	High Risk	Low Risk	Low Risk
8	High Risk	Medium Risk	Medium Risk
9	High Risk	High Risk	High Risk

ensure that the model did not include any abnormalities. In order to construct a model that is capable of delivering the desired result, it is necessary to separate the abstract features included within the dataset. Because of choosing a model is such a crucial step, we have analysed the results of a comparison between the performance of an artificial neural network model, the model selection process, and the selected machine learning model. The suggested fall detection system features a three-stage modelling process depicted in the flowchart shown in 4.7. In the first phase, the model selection technique is used to identify the best-performing multi-class classification model, which outperforms the others. The selected ML model is used in the second phase to build a fall detection system that can predict the response variable. In the final phase, the performance of the designed two-layered feed-forward artificial neural network is compared with the best-selected ML model from the first phase. In the model selection phase, four classification algorithms are taken into account, i.e., k-NN, decision tree classifier, random forest classifier, and extreme gradient boosting.

- **k-NN:** Each object is graded by a majority vote of its neighbours, and the entity is allocated to the most common class of its nearest k neighbours. The purpose of considering the k-NN model is because the nearby linear accelerometer or gyroscope sensor data points may form a specific pattern, which can be used to identify a fall or non-fall.

4. A Cloud-based FaaS in for Human Activity Recognition

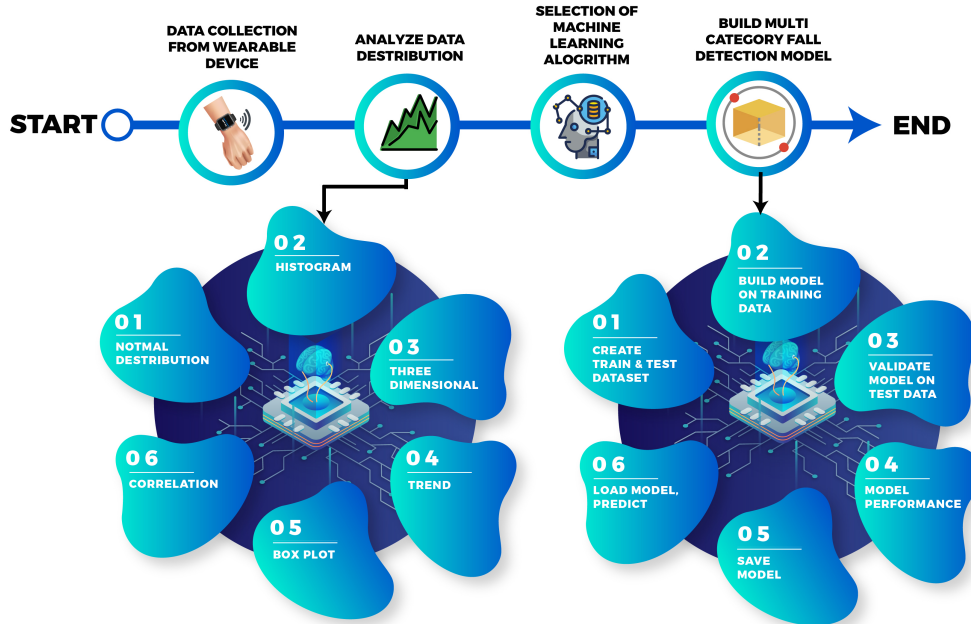


Figure 4.7: Flow chart of the proposed fall detection system.

- **Decision tree classifier:** In decision tree classification, branches represent independent variables, and leaves represent class variables. The purpose of using a decision tree classifier was to validate the effect of decision tree learning on predicting a fall and a non-fall.
- **Random forest classifier:** It is a classification method for learning an ensemble. It is a set of decision trees from a randomly chosen training subset. It predicts the final class by aggregating the votes from various trees for decisions.
- **Extreme gradient boosting:** is also an ensemble learning method and a decision tree-based algorithm where gradient descent optimisation is used for minimising errors to optimise parallel processing, tree pruning, and the model's over-fitting.

A three-Layered feed-forward ANN is also built to predict the fall. Here, the neural network can help us to learn the latent features among the data which

4. A Cloud-based FaaS in for Human Activity Recognition

may not be captured using discussed ML models. This neural network contains the following layers:

- **Input Layer:** It has the same dimension as input data. In our case, we have six features where three features represent linear accelerometers, and the remaining three represent the gyroscope data.
- **Hidden Layer:** There are 2 hidden layers, each having 600 neurons. The output shape for these hidden layers is (None, 600). Rectified Linear Unit (ReLU) is used as a non-linear activation function.
- **Output Layer:** As this is a multi-class classification problem and it is aimed to predict five classes (backside fall, forward fall, side fall, normal walk, running), there will be five neurons in the output layer. Each neuron represents one class, and at the end, the final prediction will be whichever neuron gets the highest probability.
- **The Softmax regression** is a type of logistic regression that normalises an input value into a value vector that follows a distribution of probabilities with a total amount of up to 1.

$$S(y = j | z) = \frac{e^z}{\sum_{j=0}^k e^{z_k}} \quad (4.4)$$

where we define the net input Z as

$$z = \sum_{i=1}^m w_i x_i = W^T X \quad (4.5)$$

Here, W is the weight vector, and X is the feature vector [94]. Softmax function computes the probability that training sample X belongs to class j as forward fall, side fall, backside fall, normal walk, and running, given the weight w and net input z .

4. A Cloud-based FaaS in for Human Activity Recognition

Table 4.2: A sample part of the run/walk dataset used in this experiment for both training and testing.

date	time	activity	acceleration_x	acceleration_y	acceleration_z	gyro_x	gyro_y	gyro_z
30/06/2017	13:55:45	0	0.2927	-0.8271	0.0291	-0.5193	0.5369	-1.3963
30/06/2017	13:55:45	0	0.4243	-1.0259	-0.2221	0.8334	-0.0653	1.7977
30/06/2017	13:55:46	0	0.3722	-0.6749	-0.1641	0.7345	0.238	1.2111
30/06/2017	13:55:46	0	0.5202	-1.2726	-0.3422	-0.3252	-0.2746	0.501
30/06/2017	13:55:46	0	0.5255	-1.0243	-0.1791	-0.7732	0.2769	-1.2629
30/06/2017	13:55:46	0	0.2717	-0.7515	0.0187	-0.544	-0.1414	-1.817
30/06/2017	20:33:44	1	1.2842	0.0526	-0.2362	-1.9543	2.4977	1.3038
30/06/2017	20:33:44	1	0.3057	0.4504	-0.1	1.473	-0.5391	-2.603
30/06/2017	20:33:44	1	1.0601	-0.9857	-0.0732	1.43	-0.9917	-2.3499
30/06/2017	20:33:44	1	-0.1065	-0.7203	0.2003	-0.4013	0.424	3.0591
30/06/2017	20:33:45	1	1.0069	0.3441	-0.276	-1.6086	2.3222	1.4736
30/06/2017	20:33:45	1	0.0997	0.4844	-0.0588	1.0809	-0.6441	-2.7383
30/06/2017	20:33:45	1	1.5466	-1.1082	-0.2509	0.8609	-0.4994	-1.6458

4.3 Experiment Methodology for HAR

In this section, the methodology adopted for implementing FaaS suitable for HAR is discussed with particular emphasis on Data sets used for performing the experiments. The design of the proposed FLS and the rule-based training required are discussed.

4.3.1 Dataset Used

To evaluate the performance of the proposed system, a dataset taken from Kaggle.com [<https://www.kaggle.com>] open repository called “Run or Walk.” The dataset contains 88588 sensor data samples from accelerometer and gyroscope collected from smartphones located on a person’s wrist and an average of about 5 seconds frequency. This dataset contains six attributes, including accelerometer and gyroscope sensor data, each in 3 dimensions, along with their timestamps. The dataset is also labelled with the actual status, where 1 indicates that the person is running and 0 indicates that the person is walking. The dataset includes about 90K data samples, where about 60K samples were used for training, and the remaining data were used for testing. A sample part of the Run or Walk dataset is shown in Table 4.2.

4. A Cloud-based FaaS in for Human Activity Recognition

```

<?xml version="1.0" encoding="UTF-8" ?>
<fuzzyController type="request" service="createFLS">
  <URI>FLS7</URI>
  <service>createFLS</service>
  <fuzzySystem name="Bhavesh">
    <knowledgeBase>
      <fuzzyVariable name="ax" scale="" domainleft="0.0" domainright="2.5" type="input">
        <fuzzyTerm name="1" complement="false">
          </fuzzyVariable>
          .....
        <fuzzyVariable name="gz" scale="" domainleft="0.0" domainright="2.5" type="input">
          <fuzzyTerm name="1" complement="false">
            <triangularShape param1="-0.625" param2="0.0" param3="0.625"/>
            </fuzzyTerm>
          </fuzzyVariable>
          <fuzzyVariable name="y" scale="" domainleft="-5.0" domainright="5.0" type="output" accumulation="MAX"
          defuzzifier="COG" defaultValue="0.0">
            <fuzzyTerm name="0" complement="false"> </fuzzyTerm>
            <fuzzyTerm name="1" complement="false"> </fuzzyTerm>
          </fuzzyVariable>
        </knowledgeBase>
      <mamdaniRuleBase activationMethod="MIN" andMethod="MIN" orMethod="MAX" networkAddress="127.0.0.1">
        .....
        <rule name="1" andMethod="MIN" orMethod="MAX"
        connector="AND" weight="1.0"
        networkAddress="127.0.0.1">
          <antecedent>
            <clause>
              <variable>ax</variable>
              <term>1</term>
            </clause>
            .....
            <clause>
              <variable>gz</variable>
              <term>1</term>
            </clause>
          </antecedent>
          <consequent>
            <then>
              <clause>
                <variable>y</variable>
                <term>1</term>
              </clause>
            </then>
          </consequent>
        </rule>
        .....
        <rule name="247" andMethod="MIN" orMethod="MAX"
        connector="AND" weight="1.0"
        networkAddress="127.0.0.1">
          <antecedent>
            <clause>
              <variable>ax</variable>
              <term>4</term>
            </clause>
            .....
            <clause>
              <variable>gz</variable>
              <term>1</term>
            </clause>
          </antecedent>
          <consequent>
            <then>
              <clause>
                <variable>y</variable>
                <term>1</term>
              </clause>
            </then>
          </consequent>
        </rule>
      </mamdaniRuleBase>
    </fuzzySystem>
  </fuzzyController>

```

Figure 4.8: Partial KB and RB of the adopted FML.

4.3.2 FLS Design

A fuzzy logic system defined in IEEE-1855 can be processed in numerous programming languages, such as Java [95], using an extensible style sheet language translator. So, minimal attempts from the server are required to encode a fuzzy logic system's description into a local program logic. IEEE-1855 also authorises various agents to monitor the same fuzzy logic system that communicates with the environment from distinct locations. This study aimed to focus on FML's known capabilities in defining an FLS by considering essential parameters such as fuzzy input sets, rule base, inference method, output fuzzy sets, and defuzzification [96].

The training outcome is used to compose the FLS description in FML format. Triangular fuzzy sets for inputs and singleton fuzzy sets for the output were embedded in the FLS description. Other parameters set for the FLS include centroid defuzzification and Mamdani inference with min and max operators for the t-norm and t-conorm, respectively. Finally, the FLS description was composed in FML format and was sent to the server via a single API call from a simulated "FLS designing station" client machine. Lists of Partial KB and rule-base of the adopted FML are shown in Figure 4.8.

It is noticeable that the same client machine can later modify the FLS parameters dynamically. It can even intervene in the middle of an experiment by resending an updated FLS description via another API call. The FLS outputs will be automatically adjusted in real-time according to the new FLS parameters. This is an important feature to be considered in adaptable and dynamically trained FLSs.

4.3.3 Rule-base Training

Rule-base training was carried out using the training set part of the dataset, based on the known Wang-Mendel's "learning from example" algorithm [97] and [13]. This algorithm is able to build fuzzy rules given the number of required fuzzy sets for each input-output. The algorithm was independently implemented prior to the main experiment. Five evenly distributed fuzzy triangular sets were used for each input data, whereas the binary output data was represented as two

4. A Cloud-based FaaS in for Human Activity Recognition

Rule Number	Input						Output then Activity Y is
	if ax is	and ay is	and az is	and gx is	and gy is	and gz is	
1	1	2	3	3	2	1	1
2	1	3	2	3	2	1	1
3	1	3	2	3	5	5	1
.....							
55	3	2	3	2	2	4	0
56	3	2	3	2	3	2	0
57	3	2	3	2	3	3	1
58	3	2	3	2	3	4	0
.....							
116	3	3	3	2	4	2	0
117	3	3	3	2	4	3	1
118	3	3	3	2	4	4	0
119	3	3	3	2	4	5	1
120	3	3	3	2	5	3	1
.....							
151	3	3	3	4	4	3	1
152	3	3	3	4	4	4	0
153	3	3	3	4	4	5	1
154	3	3	3	4	5	2	0
.....							
245	5	3	3	4	3	2	1
246	5	3	3	4	3	3	1
247	5	3	3	4	5	2	1

Figure 4.9: A sample list of rules created by the rule-base training algorithm.

singleton fuzzy sets located at 0 and 1.

As a result, 247 rules were created on six inputs and one output, samples of which are shown in Figure 4.9. The number of rules is relatively high, leading to a computationally intense FLS. Achieving real-time data processing with such a system might become slow to respond in real-time if implemented locally in the environment using the low processing power of the embedded devices. Therefore, this could be a suitable case for checking if the Web service can provide real-time processing.

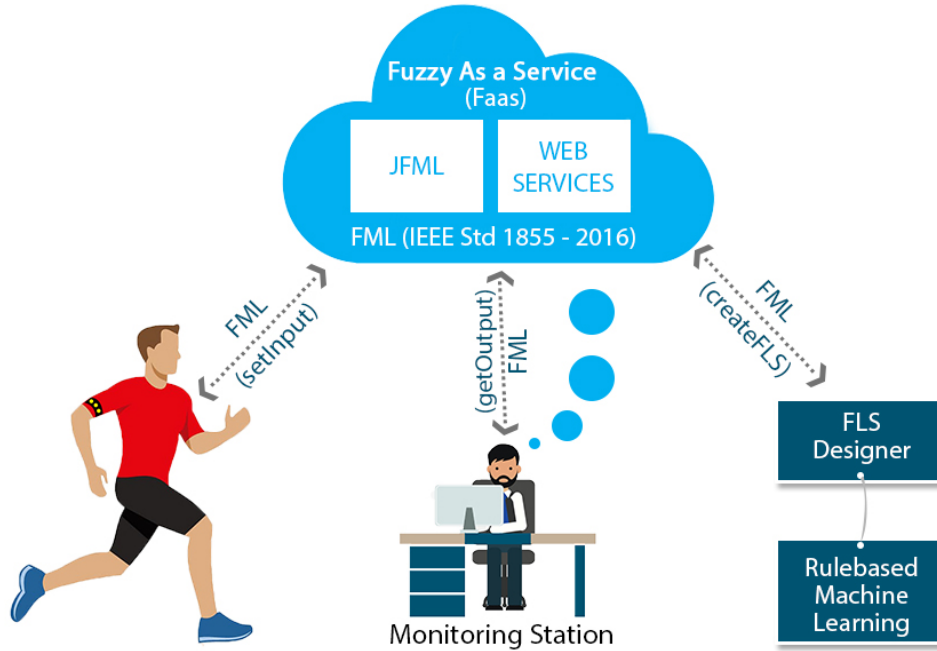


Figure 4.10: The elements of the carried-out experiment for the HAR scenario.

4.4 Experimental Settings

A FaaS solution for a human activity classification scenario is developed to demonstrate the utility of the described architecture. The components of the FaaS designed for this HAR scenario are illustrated in Figure 4.10.

A dataset of accelerometer/gyroscope measurements acquired from body sensors of individuals labelled with their walking/running status is used to validate the system. A fuzzy rule-based system is designed in which sample input/output pairs train the rules. Then the designed FLS is set up on the Web server to classify the individual’s current status as running or walking. All the FLS definitions, input data collection, data processing, and output classification (running/walking) data are being carried out in real-time purely via API calls. IEEE standard 1855-2016 and its extension are used as the core schema in the API data exchange. The extension is added to the schema to support exchanging input/output values between the clients and the servers. Briefly, the “type” attribute determines if the HTTP packet is a request or a response. In contrast, the name of the requested functionality is encoded in the “service”

4. A Cloud-based FaaS in for Human Activity Recognition

attribute of both request and response packets, as explained in Chapter 3.

In a real-world scenario, the `setInput` API calls should be sent by Web-connected sensors or through some Web-connected interfaces. In our simulation, a client-side computer program is developed that reads the sensor data from the dataset and sends them back-to-back to the Web server at the same rate that the sensors originally produced them. Another program (running on a different client PC) is also written that simulates a “monitoring station”, in which the runner/walker status is requested from the Web server at some different rate from that of the data collection (by sending back-to-back `getOutput` API calls). On each `getOutput` request, the server runs the necessary FLS calculation and delivers the real-time running/walking status back to the monitoring station. As a result, human activity is detected in real-time.

The experiment and its results were limited to a specific sample scenario. They were presented here as a proof of concept for the suggested approach, which will be applied in subsequent works in a real-world problem where more in-depth research will be done for real-world scenarios, and actual sensors/output devices are established.

4.5 Results and Discussions of experiment performed for fall detection

Data collection was done for the fall detection in actual settings with human subjects. In this study, 30 sample datasets were collected from 6 participants, 18 of whom were enrolled. In the fall operation, 12 were held, and 12 were detected for fall for non-fall behaviour. The data set was divided into 70:30 of the train-test ratios. A total of 97,320 samples were considered in our study, where 68,124 samples were used as a training data set, and 29,196 samples were used as a test data set. In order to test the designed fall detection algorithms, there are four specific cases for each prediction: True Positive (TP) shows that fall is accurately recognised, True Negative (TN) means that fall is not noted, mainly where fall occurs in a very slow motion, False Negative (FN) indicates that a non-fall event is categorised correctly, and False Positive (FP) when a non-fall event identified

4. A Cloud-based FaaS in for Human Activity Recognition



```
GetOutputs [Java Application] C:\Program Files\Java\jre
09:27:29.508 [main] DEBUG org.spring
09:27:29.508 [main] DEBUG org.spring
09:27:29.508 [main] DEBUG org.spring
<URI>FLS1</URI>
<variable name="ax">
  <value>0.89272</value>
</variable>
<variable name="ay">
  <value>0.43052</value>
</variable>
<variable name="az">
  <value>0.85505</value>
</variable>
<variable name="gx">
  <value>0.09531</value>
</variable>
<variable name="gy">
  <value>0.28577</value>
</variable>
<variable name="gz">
  <value>0.38322</value>
</variable>
<uri>FLS1</uri>
</FuzzyControllerType>
] as "application/xml" using [org.sp
09:27:29.524 [main] DEBUG org.spring
```

Figure 4.11: Real-time human activity classification process using HTTP request/responses to the developed Web Services. A sensor client console view: sending back-to-back XML requests to the server to set the input variables (i.e., the accelerometer and gyroscope data).

as a fall.

The results of testing FaaS with the collected datasets is shown in Table 4.3 and a screenshot of real-time fall detection in FaaS is shown in Figure 4.13. In FaaS, the sensitivity outcome for fall activity is 88.89%, the specificity result could reach 91.67%, and 90% is the total accuracy of fall and non-fall activities. The FaaS could successfully respond in real-time and tolerate the actual sample collection rate.

After testing with FaaS, the same datasets were also fed into the other developed ML methods. Table 4.4 and Table 4.5 shows the results of the feed-forward ANN, which provides an average of 92% accuracy. Table 4.6 compares different machine learning algorithms in terms of their average accuracy and standard deviation, and Table 4.7 shows the result analysis of Random Forest. Though the average accuracy of ANN is low compared with the RF method, it proves that ANN can also be a good alternative if data

4. A Cloud-based FaaS in for Human Activity Recognition

Table 4.3: Fall detection result using FaaS

TP	16	TP + FN	18
FN	02		
TN	11	TN + FP	12
FP	01		
Sensitivity	88.89%	Specificity	91.67%
Accuracy	90%		

Table 4.4: Training and testing of a dataset with respective time using of ANN.

EPOCH	Training	Training	Testing	Testing	Time (sec)
	Accuracy (%)	Loss	Accuracy (%)	Loss	
1	48.96%	1.2248	60.04%	0.9999	63
2	65.81%	0.8732	71.47%	0.7354	64
3	74.47%	0.6733	75.74%	0.6421	69
4	79.51%	0.5595	79.81%	0.5469	69
5	82.63%	0.4776	82.66%	0.4945	69
6	84.78%	0.429	83.92%	0.4504	71
7	86.45%	0.3886	86.10%	0.3956	70
8	87.70%	0.3595	88.22%	0.3526	70
9	88.62%	0.3411	87.41%	0.3768	69
10	89.56%	0.3181	88.38%	0.3335	69
11	90.18%	0.3037	89.44%	0.3251	70
12	90.61%	0.2918	90.94%	0.2795	72
13	91.22%	0.2834	90.15%	0.3195	68
14	91.51%	0.2815	91.33%	0.2997	71
15	91.81%	0.2772	91.12%	0.3189	69
16	92.30%	0.2616	91.28%	0.2996	72
17	92.67%	0.2511	91.96%	0.2839	69
18	92.88%	0.2447	91.30%	0.3092	70
19	92.98%	0.2472	92.01%	0.2851	71
20	93.28%	0.2386	92.82%	0.2514	71

4. A Cloud-based FaaS in for Human Activity Recognition

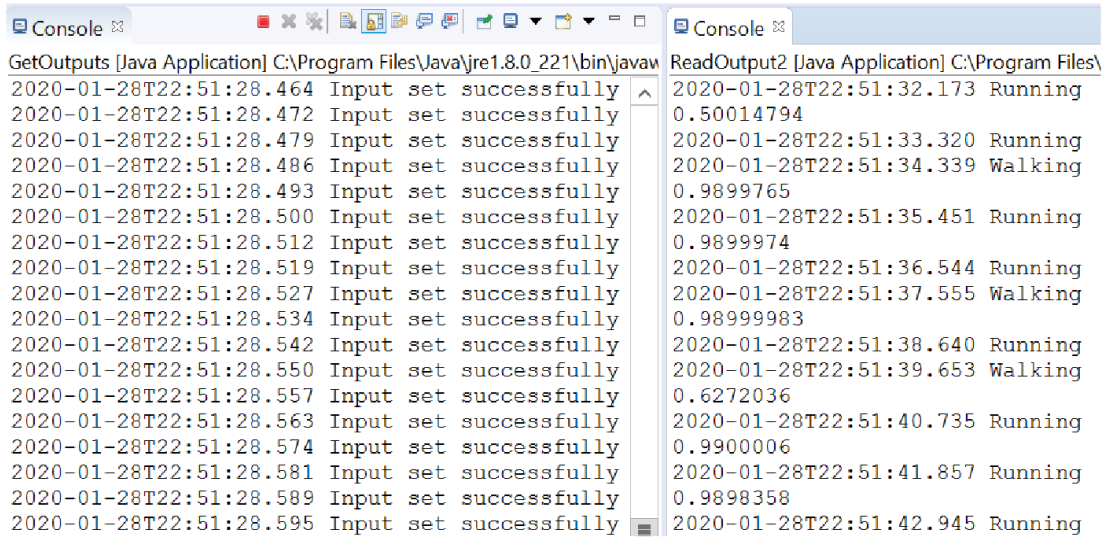
Table 4.5: Result analysis of ANN.

Artificial Neural Network							
No	Action	True Postive	True Negative	False Postive	False Negative	Sensitivity	Specificity
1	Backside Fall	5326	23030	497	343	0.9394955	0.97887533
2	Forward Fall	5307	22705	748	436	0.92408149	0.96810643
3	Normal Walk	5641	22860	287	408	0.93255083	0.98760099
4	Running	5437	23125	184	450	0.92356039	0.99210605
5	Side Fall	5390	22969	379	458	0.92168263	0.98376735
	Average	5420.2	22937.8	419	419	92.83%	98.21%

Table 4.6: Comparison of different machine learning algorithms.

Model Selection		
Algorithm	Average Accuracy	Standard Deviation
KNN	96.85%	0.20%
Decision Tree classifier	96.18%	0.14%
Random Forest	99.19%	0.10%
XGBoost	84.18%	0.53%

4. A Cloud-based FaaS in for Human Activity Recognition



The image shows two side-by-side console windows. The left window, titled 'GetOutputs [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw', displays a series of log entries: '2020-01-28T22:51:28.464 Input set successfully' through '2020-01-28T22:51:28.595 Input set successfully'. The right window, titled 'ReadOutput2 [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw', displays log entries showing activity classifications and their corresponding response times: '2020-01-28T22:51:32.173 Running 0.50014794', '2020-01-28T22:51:33.320 Running 0.9899765', '2020-01-28T22:51:34.339 Walking 0.9899974', '2020-01-28T22:51:35.451 Running 0.9899974', '2020-01-28T22:51:36.544 Running 0.9899983', '2020-01-28T22:51:37.555 Walking 0.6272036', '2020-01-28T22:51:38.640 Running 0.9900006', '2020-01-28T22:51:39.653 Walking 0.9898358', '2020-01-28T22:51:40.735 Running 0.9898358', '2020-01-28T22:51:41.857 Running 0.9898358', and '2020-01-28T22:51:42.945 Running 0.9898358'.

Figure 4.12: Consoles views of two monitoring clients: On the right side the server acknowledgements per request are received; On the right side, the output values (i.e., the activity classification) per request are coming from the server (the client's requests for getting the output values are not shown here).

becomes more complex. It is shown that the RF method offers a prediction accuracy of 98.53%. Moreover, the model can accurately predict the response variable because it achieves an overall sensitivity and specificity of 98.53% and 99.63%, respectively.

4.6 Results and Discussions of experiment performed for HAR

Once the FLS is created on the server side, the FaaS system is ready to serve setInput and getOutput requests coming from different clients simultaneously. Two client programs independently started to send back-to-back setInput and getOutput requests to the server in different frequencies. The highest testing rate was two simultaneous requests every one second, whereas the original data collection frequency was about 5 seconds. In this case, the server could calculate the results based on the latest coming input and send the defuzzified output back within the 1-second window without missing any of the coming requests.

4. A Cloud-based FaaS in for Human Activity Recognition

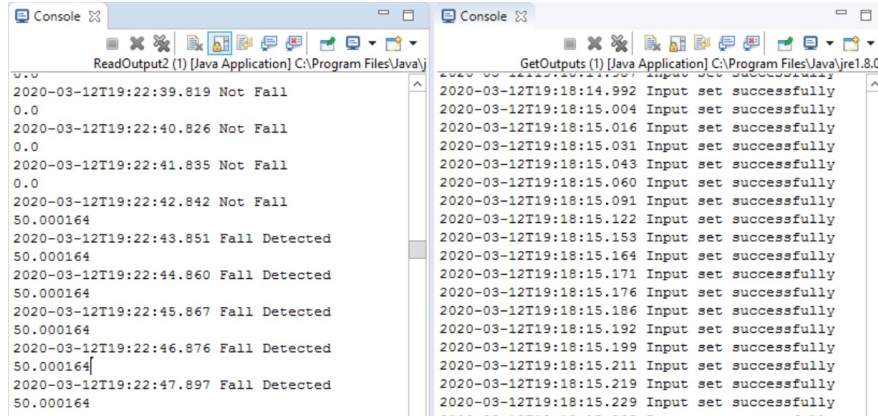


Figure 4.13: A screenshot of fall detection request/response in FaaS

Table 4.7: Result analysis of Random Forest.

Random Forest							
No	Action	TP	TN	FP	FN	Sensitivity	Specificity
1	Backside Fall	5645	23482	45	24	99.25%	99.48%
2	Forward Fall	5678	23408	45	65	97.99%	99.43%
3	Side Fall	6003	23114	33	46	98.33%	99.73%
4	Normal Walk	5866	23270	39	21	98.94%	99.74%
5	Running	5806	23312	36	42	98.13%	99.76%
-	Average	5799.6	23317.2	39.6	39.6	98.53%	99.63%

This shows the ability to process data in a computationally intense FLS over the developed Web services at a more sampling rate that would be necessary for the real-world scenario. Figures 4.11 and 4.12 shows a sample running of the system where it serves the requests of both clients in parallel.

The accuracy of the activity classification performed by this FLS Web Service is shown in Table 4.8. This shows a noticeably high accuracy processed in real-time. Although the classification performance of the FLS is provided here, it is noticeable that the purpose of this experiment is not to check if the designed FLS can do any more or less accurate than other classifiers. Instead, this should be considered as a proof-of-concept for achieving a real-time FLS execution over using Web services, which has yet to have any similar implementation as far as we are aware.

4. A Cloud-based FaaS in for Human Activity Recognition

Table 4.8: Real-time Human Activity Monitoring result.

Data samples	Type	Accuracy
59058	Training	97.23%
29530	Testing	97.42%
88588	Overall	97.30%

4.7 Chapter Summary

The study under this chapter presents an analysis of real-time fall detection using a web-based service-oriented FLS architecture and machine learning-based methods. It is shown that FaaS can keep up real-time processing of the sensory data with the actual sample rate necessary for fall detection, where it can detect a fall and a non-fall with an accuracy of 90%. In contrast, machine learning techniques can detect five classes of a fall: side fall, back fall, forward fall, walking, and running, with a maximum accuracy of 99.19% offered by the random forest technique shown in Fig.4.7.

It is observed that FaaS can detect falls from real-time data as it requires minimum hardware and software specifications. Hence, it is concluded that a cloud-based FaaS to generate real-time results is capable for such applications. The main purpose of using FaaS is to permit multipurpose delivery from clients to dedicated servers that perform complex computations required for FLSs. Unequivocally, virtualised cloud services provide the proposed system with elasticity. Reusing existing data, balancing load amongst FLS devices, and cost-efficiency are some advantages of FLS architecture.

A comparison between the performances of a standalone hardware-dependent solution and a cloud-based FaaS is discussed in the next chapter. Moreover, the analysis and evaluation are performed on a human fall detection scenario involving wearable sensors.

This chapter has presented an example of implementing the recently accepted IEEE-1855-2016 standard. It is shown through experimenting with human activity monitoring datasets that the proposed service-oriented architecture can undertake real-time data processing using a complex fuzzy rule-based system. The accuracy and the response time of the developed system

4. A Cloud-based FaaS in for Human Activity Recognition

are found to be relatively high. The experiment outcome offers a novel approach for a web-based service-oriented FLS architecture. Although the architecture is viewed in AmI environments, it can be extended to a much broader application field, i.e., wherever an FLS' storage logic requires to be abstracted from its logic of information and presentation. The primary motivation is to allow the versatile delivery from the clients to dedicated servers of potentially complex computations needed for FLSs. Unambiguously, virtualised cloud services provide the system with elasticity, essentially allowing universally accessible FaaS. The other benefits of such an architecture are network share, hardware/software autonomy, reuse of existing data, balancing the load between FLS devices, and cost-efficiency.

One notable limitation is the potential complexity associated with implementing the IEEE 1855-2016 standard in real-world applications, which may pose challenges in terms of compatibility and interoperability with existing systems. Additionally, the study may not fully explore the scalability and efficiency of the proposed web services architecture, particularly in terms of handling large volumes of sensor data and concurrent user requests. Furthermore, the generalizability of the findings may be limited by the specific datasets and experimental setups used in the study. Addressing these gaps and limitations would contribute to the advancement and practical implementation of fuzzy logic-based HAR systems using the IEEE 1855-2016 standard.

The subsequent chapter will explain the Framework, implementation, and performance evaluation of a system for monitoring SpO_2 and heart rate collected from wearable sensors via mobile phone and classification of the resulting data using a cloud-based FLS.

Chapter 5

A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

5.1 Introduction

Over time, it has been observed that people's living standards have improved, specifically regarding health-related qualities. The evolution of cloud computing and cloud-based services offer a pervasive solution to the healthcare monitoring system. The system users prefer using wearable devices with sensors, which help monitor individual health [98, 99]. The application of lightweight sensors in measuring physiological parameters supports monitoring human health conditions regularly without visiting health centres. This has led to better living conditions for human beings, especially for older adults, in improving their life expectancy. A wearable pulse oximeter is an example of a non-invasive device utilised to analyse oxygen saturation (SpO_2), and heart rate [100, 101].

Moreover, the applications of the concerned technologies can further be extended to monitor a person under observation remotely at any time from anywhere [102, 103]. Moreover, healthcare service-associated cost rises because of the rising cost of pharmaceutical drugs and devices [104]. Thus, it is necessary to acquire and realise new tactics and know-how for offering superior

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

quality healthcare services at a cheaper rate to the ageing population, thereby ensuring maximum comfort.

Recently, by means of IoT and signal processing techniques, wearable non-invasive plasma O_2 monitoring has become possible[105, 106, 107]. People can check their plasma O_2 saturation, pulse rate, etc., at their residence and acquire data about the changes in their breathing and arterial oxygen saturation [99, 108]. Many Machine Learning tools and techniques are employed to process the collected data from sensors.

This chapter aims to develop a system that remotely monitors two physiological aspects of individuals, namely blood oxygen saturation (in %) and pulse rate (in bpm), using wearable sensors within the proposed new cloud-based fuzzy logic system architecture (FaaS). Monitoring health remotely based on wearable and non-invasive sensors provides a cost-effective solution, which in turn permits the elderly to continue to live in his/her home environment instead of spending on expensive healthcare facilities.

Implementing the monitoring application under the FaaS architecture allows real-time monitoring of the individual's health, not only by the individuals, but also by healthcare personnel in different geographical locations. Moreover, the developed rule-based fuzzy system can infer an individual's health condition based on the fuzzy rules. The acquired results are accessible through a developed mobile application for both individuals and healthcare personnel. The cloud-based inference engine can also offer scalability, so that many individuals can be monitored simultaneously.

Developing this monitoring system was in the era of the COVID-19 pandemic. An application of the system was understood to be detecting early signs of COVID-19 in the monitored individuals, since the abnormal changes in both monitored physiological aspects could contribute to early COVID-19 detection, leading to an early treatment for the patient as well as reducing the infection risk for the other individuals in the environment.

The remainder of this chapter is organised in the following manner. Section 5.2 delves into the architecture of the health monitoring system and the methodology, which includes the use of fuzzy logic web services and a modelling approach. Section 5.3 emphasises the results and discussion. The summary of this chapter

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

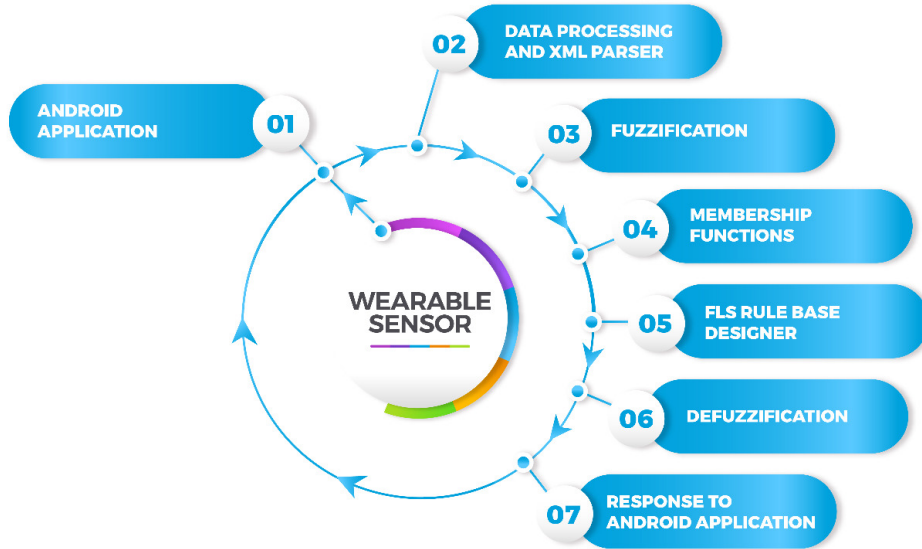


Figure 5.1: Schematic illustration of the proposed system.

is discussed in Section 5.6 with a discussion of future scope.

5.2 Methodology

The proposed system is developed to monitor an individual's health conditions by means of values acquired from heart rate and SpO_2 , which act as the measurement parameters for managing health conditions. The FaaS architecture explained in Chapter 4 is employed using Microsoft Azure and AWS cloud platforms. In the FLS server, fuzzy variables and membership functions are defined and processed by a rule-based fuzzy logic system, and its output is displayed on an Android application on the client side(s). The outline of the data flow in the proposed system is illustrated in Figure 5.1. The framework of the proposed architecture, including data collection point, data processing, monitoring point, design point, and fuzzy logic system is shown in Figure 5.2.

For data collection, BM2000A Smart Care Oximeter wearable wrist pulse oximeter sensor was used¹. The sensor was used to collect the physiological

¹More details are available from: <http://devices.smartcareanalytics.co.uk/>

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

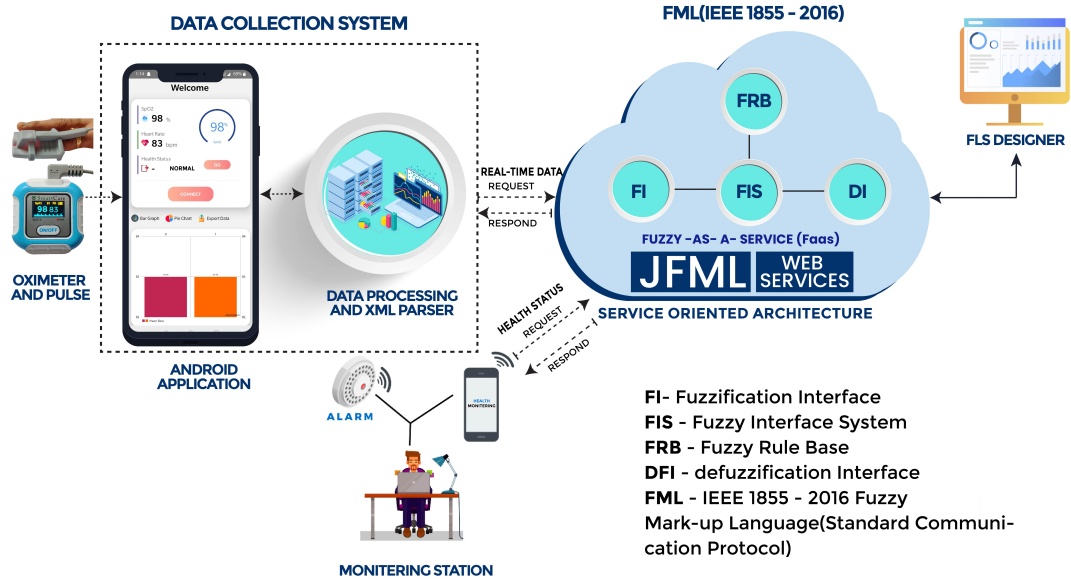


Figure 5.2: Framework of the proposed architecture including data collection point, data processing, monitoring point, design point, and fuzzy logic system.

Table 5.1: Dataset with a few instances of feature-extracted data.

TIMESTAMP (MilliSeconds(MS))	PULSE BPM	SpO ₂ PCT	SpO ₂ STATUS	PLETH	RED_ADC	IR_ADC	PERFUSION INDEX
0	87	97.7	0	25684	193775	250301	7.8
10	87	97.7	0	27597	193872	249747	7.8
20	87	97.7	0	29524	194064	250321	7.8
30	87	97.7	0	31383	194311	249979	7.8
40	87	97.7	0	33099	194559	251033	7.8
50	87	97.7	0	34598	195154	252450	7.8
60	87	97.7	0	35836	195384	252363	7.8
70	87	97.7	0	36778	195897	253078	7.8
80	87	97.7	0	37403	196275	253831	7.8
.
.
.
124490	80	99.4	0	35096	273595	237903	1.4
124500	80	99.4	0	36308	273428	237995	1.4
124510	80	99.4	0	37626	273529	238138	1.4
124520	80	99.4	0	39009	273432	237897	1.4
124530	80	99.4	0	40407	273272	237526	1.4
124540	80	99.4	0	41766	273303	237676	1.4
124550	80	99.4	0	43067	273246	237656	1.4
124560	80	99.4	0	44256	273398	237882	1.4
124570	80	99.4	0	45324	273273	237634	1.4
124580	80	99.4	0	46227	273077	237433	1.4

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

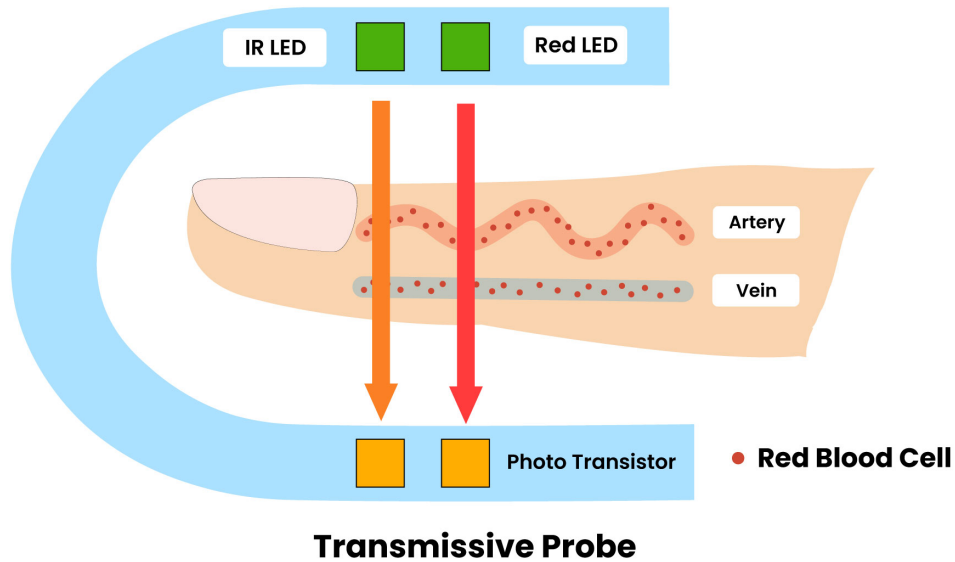


Figure 5.3: Transmittance oxygen saturation monitoring principle.

data, namely, heart rate and SpO_2 shown in Table 5.1. Sensor details with the respective error code are as follows:

1. SpO_2 status is 0 if everything is ok.
2. 0x01: SpO_2 sensor not connected
3. 0x02: No finger detected in sensor
4. 0x04: Could not detect pulse
5. 0x08: Indicates the pulse beats

Pleth is the plethysmograph signal shown on the screen of the oximeter and the app. As shown in Figure 5.3, the RED/IR ADC values are the raw values from the analog-to-digital converters for the red and infrared sensors in the probe. These are used to calculate the SpO_2 value. Perfusion index is the ratio of the pulsatile blood flow to non-pulsatile. After receiving data from an Android-based wearable device, SpO_2 and heart rate data are transferred to the cloud server in the extended FML format, where the FLS is created and maintained. It is

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

required to set the sensor data as input values in the fuzzy system for different variables so that the system can return the desired output. As discussed in Section 3.4.2, API function calls are designed for data exchange.

5.3 Fuzzy System Design

The system architecture is designed to track health conditions using values extracted from heart rate and oxygen saturation, which can be used as a standard for condition-based monitoring of the person's health under observation. The wearable device's output value in the application should be processed using fuzzy logic. The design of the fuzzy sets and rules are as follows:

- Fuzzy Input Sets: SpO_2 is the first input that is fuzzified using four fuzzy sets for critical, alert, low, and normal states. The broad range for SpO_2 pressed in % is from 0 to 100. Pulse rate is the second input containing four sets: critical, normal, low, and alert. The range for pulse rate input is 0 to 180 beats per minute (bpm). There are four fuzzy sets for pulse rate: critical is from 0 to 60bpm, normal is between 60bpm to 90bpm, low is between 90bpm to 100bpm, and alert is above 100bpm. Through fuzzification, input signals are mapped into membership grades defined for critical, alert, low, and normal states. The system is a dual input system that accepts SpO_2 and heart rate as inputs so that the fuzzification is repeated for both inputs. The input membership functions in each group are based on having three turning points. For example, the Oxygen saturation value is usually known to be critical if it is less than 30, alert between 30 and 60, and normal if more than 60. Trapezoidal membership functions are used to convert the crisp classifications to fuzzy grades, as shown in Figure 5.4 (a and b).
- Compounding: As shown in Figure 5.4(a), there are four fuzzy sets associated with oxygen saturation membership, critical which is less than 85%, alert which is between 86% to 90%, low is 91% to 94% and normal is above 95%.
- Rule Base: A total of 16 rules are created to identify humans' health status.

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

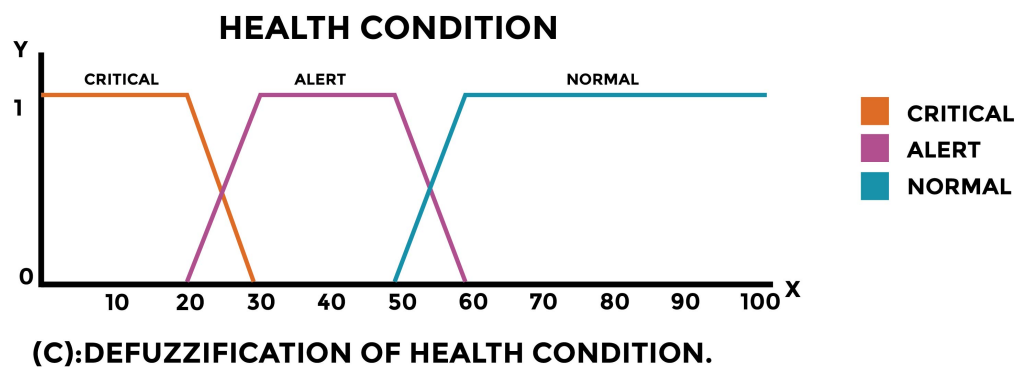
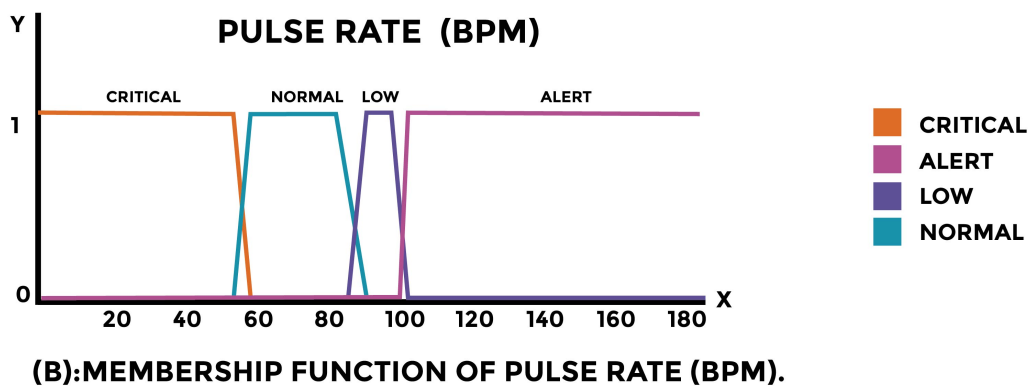
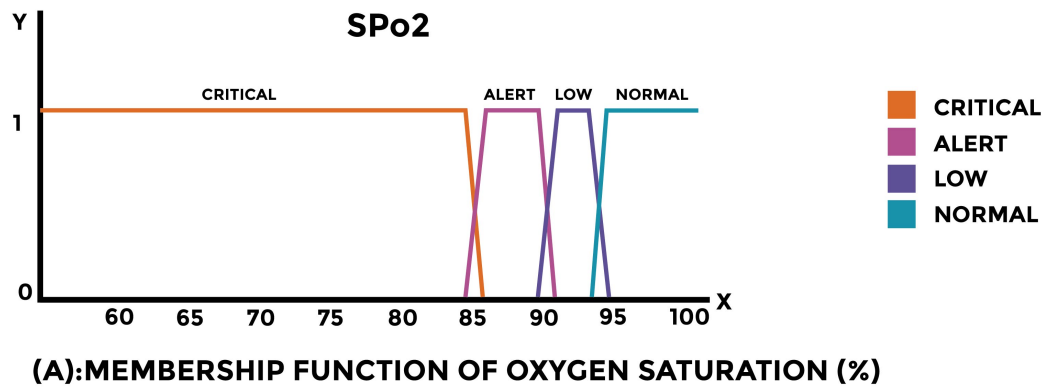


Figure 5.4: Membership functions for fuzzy inputs:(a) SpO_2 (b) Pulse rate and membership for output (c) Health Condition.

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

Table 5.2: FLS rules for calculating health status.

No	SP_{O_2}	Pulse	Health Status
1	Critical	Critical	Critical
2	Critical	Normal	Alert
3	Critical	Low	Alert
4	Critical	Alert	Critical
5	Alert	Critical	Critical
6	Alert	Normal	Normal
7	Alert	Low	Normal
8	Alert	Alert	Alert
9	Low	Critical	Alert
10	Low	Normal	Normal
11	Low	Low	Normal
12	Low	Alert	Normal
13	Normal	Critical	Alert
14	Normal	Normal	Normal
15	Normal	Low	Normal
16	Normal	Alert	Normal

The fuzzy logic processing rules, which can classify a situation as critical, alert, or normal, are shown in Table 5.2.

- Fuzzy Output Sets: According to the inputs, input fuzzy sets, and the rulebase, the output of the fuzzy system offers three classes, i.e., critical, alert, and normal. Figure 5.4(c) shows the membership grades for these three output fuzzy sets. The membership grades of these fuzzy sets are defined over a range of 0 to 100. Any output below 30 is deemed critical, 30 to 60 is called alert, and anything above 60 is considered normal. Through defuzzification, the well-known centroid defuzzification method converts the fuzzy output to crisp values.

5.4 Software Components

A Web application server is created according to the FaaS idea to provide essential HTTP communication to and from clients. Python, Android Studio, AWS and Microsoft Azure Services are the pieces of software and services to develop the

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

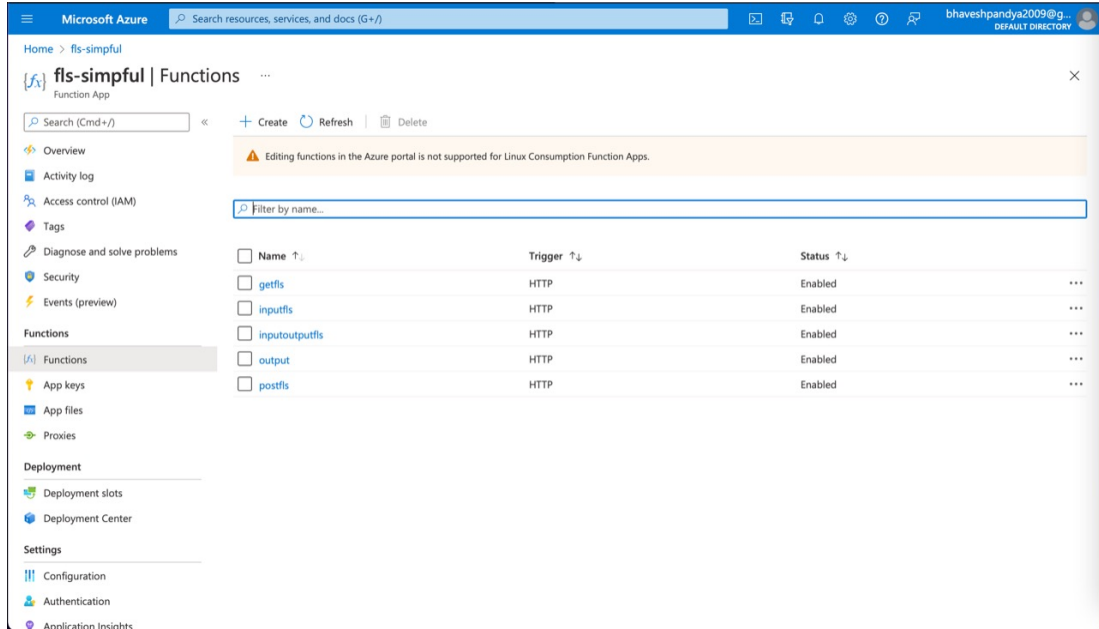


Figure 5.5: List of methods created over Microsoft Azure.

final system. Simplful library is adapted as an Azure service on the cloud. Sample adapted methods created in Microsoft Azure are shown in Figure 5.5. As an alternative arrangement, AWS was used as the cloud platform and JFML was used as the FLS software library, and the results were compared.

The proposed extended FML is used not only as the input/output format for this system, but also for exchanging the FLS design details. Therefore, an application called FLS designer is developed for this purpose. Because the developed app is not restricted to a single fuzzy logic system, it is possible to manage the design of many FLSs with separate knowledge bases that might be located on different servers, even by different cloud service providers. This is possible through identifying each FLS by a URI, as predicted in the FML schema. The same system can also be used to manually send/receive input/output data to/from the server in FML format. An example of the FLS Designer's user interface is shown in Figure 5.6, in which an FLS identified by its URI is being designed and transferred to the AWS cloud platform in FML format.

Finally, an Android app is developed using Android Studio for monitoring

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring



Figure 5.6: The User interface of the FLS Designer application. Under the “Create FLS” tab, an FLS is being designed on the client side for being sent to the server side. Other tabs are responsible for other input/output tasks as defined in the proposed extended FML.

the health status of individuals to serve as a user interface on the client-side. Sample screenshots of the developed app are shown in Figure 5.7 for three states of critical, normal, and alert.

5.5 Experimental Results

In this investigation, real-time data were collected using the wearable sensor BM2000A wrist pulse oximeter. The captured real-time data is sent to the server through an android application that assesses the outcome using the defined fuzzy rules. For example, Figure 5.7(a) indicates that the health condition is critical when the SpO_2 value is 88 percent and the heart rate is 55 bpm, which is very low according to FLS calculations. Figure 5.7(b) indicates that the health status is normal when the SpO_2 value is 98% and the heart rate is 90 beats per minute. Finally, Figure 5.7(c) illustrates how an alert for a health condition is shown when the SpO_2 value is 91 percent and the heart rate is 108 beats per minute, as determined via FLS computation.

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

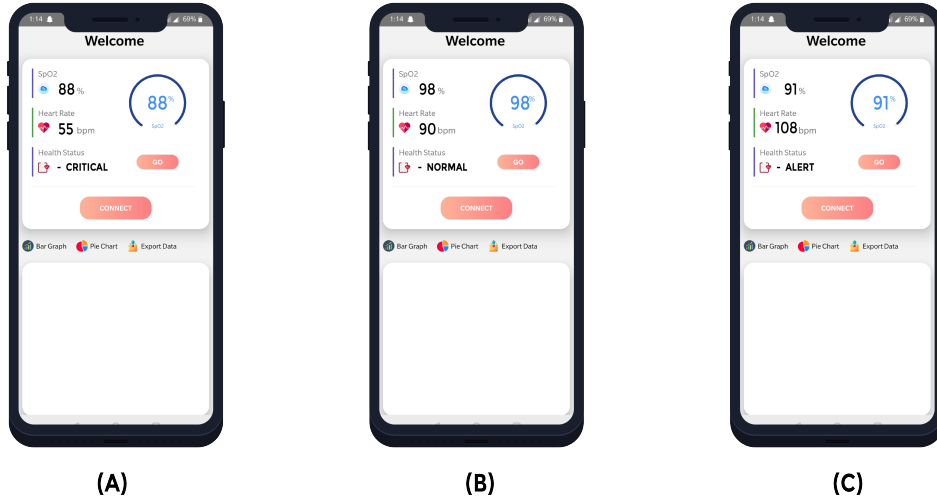


Figure 5.7: Real-time human health monitoring and processing result a) Critical b) Normal and c) Alert.

The main outcome of this experiment is the ability of FaaS to serve multiple users in a distributed and cross-platform health monitoring system. Besides that, this study aims to perform a comparative performance analysis of employing a different arrangement of cloud providers and software libraries in realising FaaS. In particular, it includes the use of two algorithms, real-time fuzzy with AWS and JFML and real-time fuzzy with Azure and Simplful libraries, on a dataset whose details are provided in Table 5.3. The experiments were performed in a real-time environment with time intervals of 30 seconds to 5 seconds in decreasing manner of 5 seconds. The result was very satisfactory with these intervals; hence started from 5 seconds down to 1 second in intervals of 1 second to check the processing time and effect of data rate on the application's performance. A data set with the size of 10000 samples has been used to evaluate the performance of AWS and Azure cloud-based systems. Table 5.3 shows the comparison between the processing time. The table interpretation indicates that AWS's performance with JFML requires less processing time of 19.65% compared to Azure with Simplful. It is noticeable that this experiment does not compare the FaaS performance results with those of any stand-alone FLS solution. A full comparative study between FaaS and non-FaaS solutions for HAR will be presented in Chapter 7.

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

Table 5.3: The analysis of cloud-based FLS on the provided dataset.

Cloud-based services	AWS with JFML	Azure with Simplful
Total records	10,000	10,000
Processing time	51 minutes	76 minutes
Processing time per record	0.306 seconds	0.456 seconds

Besides, real-time fuzzy systems with AWS and JFML support more inference systems such as Mamdani and Basilian, Takagi–Sugeno–Kang (TSK), Tsukamoto, and Anya. However, real-time fuzzy systems with Azure and Simplful libraries have less compatibility as they only support Mamdani, Basilian and Takagi–Sugeno–Kang (TSK) type-1 fuzzy inference systems. FaaS decision-making can be improved by using a larger dataset for training and testing. As compared to current systems, the proposed technique consumes the least resources in terms of hardware and software. Consequently, this study’s proposed designing systems that consume much less computational power in the local devices than the previous efforts. This increases the usability and deployability of the proposed technique in real-world circumstances.

5.6 Chapter Summary

The proposed service-oriented architecture analyses data in real-time by using a complex fuzzy rule-based system and datasets tracking human activities. Based on the above experiments, it is concluded that the proposed SOA can perform real-time data processing by means of cloud computing with human activity monitoring datasets using a complex fuzzy rule-based framework through FML. The efficiency and response time of the developed system were considered comparatively high. Even though the architecture is specified in Aml environments, it can be expanded to a broader area. Fuzzy-as-a-service is now more widely available because of the utilisation of visualised cloud services, increasing system robustness. Network sharing, hardware/software independence, data reuse, load balancing amongst FLS devices, and cost-efficiency are only a few of the advantages of this design. Technically, the suggested system is extendable in a variety of ways. The FLS community is

5. A Cloud-Based FaaS for Pervasive Health Conditions Monitoring

invited to participate and contribute to the collaborative app development effort by providing design input on feature prioritisation and contributing to the collaborative app development effort. This enhances the proposed API structure and/or invocation forms while giving more specialised feedback for specific architectural implementations. The results indicate that AWS processing time is 19.65% less than the time required by Microsoft Azure. Experiments were performed with different time intervals and monitored processing time where AWS processing time was 0.306 seconds and 0.456 seconds with Microsoft Azure. Future work could be conducted to empirically compare cloud-based web services versus standalone non-real-time fuzzy systems and non-fuzzy-based learning systems. A novel fuzzy logic algorithm has been implemented to detect a human's medical condition with the help of a real-time wearable sensor and cloud-based web services.

However, certain gaps and limitations exist in the research that warrant further investigation. Firstly, while the use of fuzzy-as-a-service shows promise for enhancing the accuracy and efficiency of health monitoring systems, there may be challenges in ensuring the reliability and security of data transmission and processing in a cloud-based environment. Additionally, the study may not fully address the potential privacy concerns and regulatory compliance issues associated with storing and analyzing sensitive health data in the cloud. Furthermore, the scalability and interoperability of the proposed pervasive application across different healthcare settings and devices may need to be further explored and evaluated. Future research could focus on addressing these gaps by conducting comprehensive usability and security assessments, as well as exploring strategies to enhance the interoperability and scalability of cloud-based health monitoring solutions. By addressing these limitations, the potential benefits of fuzzy-as-a-service for healthcare applications could be more effectively realized.

The subsequent chapter will explain the design and implementation of the system for identifying cloud-based FLS for real-time fall detection using wearable accelerometers and gyroscope sensors as a HAR, and the acquired results are displayed on a mobile application via mobile phone and classification of the resulting data using a cloud-based FLS.

Chapter 6

A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

6.1 Introduction

In the previous chapters, the feasibility and applications of FaaS in a number of HAR scenarios have been studied. Particularly in the previous chapter, a comparison between FaaS performance and other machine learning methods has been provided, however, it is still an outstanding research question, as to whether a distributed fuzzy logic system provides any advantage in processing uncertain data in HAR over the stand-alone systems? This chapter addresses this question by comparing between the performances of cloud-based FaaS solutions and a number of standalone hardware-dependent solutions for a single HAR scenario, namely, fall detection. The performance metric in this chapter is not the classification accuracy, but it is the systems' responsiveness for different sample rates in real-time processing.

The analysis and evaluation are performed on a human fall detection scenario involving wearable sensors. The proposed algorithm can identify between fall

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

and non-fall events, in a similar experimental setting to that of the previous chapter. The analysis is also carried out on two different cloud service providers and software libraries (Amazon Web Services using JFML as a Java-based library and Azure Web Services using Simpful [63] as a python-based library).

The primary contributions of this research are as follows:

- Showcasing a group of cloud-based and real-time fuzzy systems for human fall detection application areas that would not be practical to implement as standalone systems - therefore removing the design barriers that would stop implementing FLSs for such applications. A wearable smartwatch is used in this research to collect data.
- Comparing the performances of classical standalone architectures with the proposed FaaS architecture for the above case studies on the cloud and real-time high sample rate scenarios.

The remainder of this chapter is organised as follows: Section 6.2 explicates the procedures for carrying out the study and describes the various approaches used. Section 6.3 describes the experimental setup, followed by Section 6.4 that elaborates on the basis of the performed experiments. Finally, Section 6.5 discusses the summary of the chapter with future scope.

6.2 Methodology

This section describes the experimental settings for the human fall detection system, including data gathering, the non-fuzzy machine learning model, and the FaaS. Then, the performance measure and comparison methodology will be presented. In order to implement FLS for fall detection, the following approaches were used.

6.2.1 Data Collection

For this research, an experimental dataset was compiled using a linear accelerometer and gyroscope sensors connected to a smartwatch and a

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

Table 6.1: Dataset with a few instances of feature-extracted data.

Number	Time	wx	wy	wz	ax	ay	az	Gmagnitude	Amagnitude	SVM	DeltaA	Status
0	0.006	0.07	-0.17	0.02	0.34	-0.07	0.22	0.18493243	0.41097447	150	0	Not fall
1	0.007	0.07	-0.14	0.02	0.34	-0.07	0.22	0.15779734	0.41097447	150	0	Not fall
2	0.007	0.07	-0.14	0.02	0.43	-0.09	0.44	0.15779734	0.6217717	150	0	Not fall
3	0.007	0.08	-0.12	0.02	0.43	-0.09	0.44	0.1456022	0.6217717	150	0	Not fall
4	0.007	0.07	-0.1	0.03	0.43	-0.09	0.44	0.12569806	0.6217717	150	0	Not fall
5	0.008	0.07	-0.1	0.03	0.48	-0.1	0.46	0.12569806	0.67230946	150	0	Not fall
6	0.008	0.06	-0.08	0.04	0.48	-0.1	0.46	0.1077033	0.67230946	150	0	Not fall
7	0.008	0.06	-0.06	0.05	0.48	-0.1	0.46	0.09848858	0.67230946	150	0	Not fall
8	0.008	0.06	-0.06	0.05	0.46	-0.14	0.37	0.09848858	0.60671246	150	0	Not fall
9	0.008	0.04	-0.05	0.06	0.46	-0.14	0.37	0.087749645	0.60671246	150	0	Not fall
10	0.009	0.03	-0.03	0.07	0.46	-0.14	0.37	0.081853524	0.60671246	150	0	Not fall
.....												
Number	Time	wx	wy	wz	ax	ay	az	Gmagnitude	Amagnitude	SVM	DeltaA	Status
2695	4.294	-3	4.69	0.1	-8.01	-11.97	-7.57	5.568312	16.271015	400	80.61	Not fall
2696	4.294	-3.2	5.03	-0.03	-8.01	-11.97	-7.57	5.9616947	16.271015	400	81.38	Not fall
2697	4.295	-3.4	5.39	-0.16	-8.01	-11.97	-7.57	6.3747706	16.271015	400	82.27	Fall
2698	4.295	-3.4	5.39	-0.16	-6.57	-10.55	-7.42	6.3747706	14.474937	400	83.22	Fall
2699	4.295	-3.6	5.73	-0.3	-6.57	-10.55	-7.42	6.7736917	14.474937	400	84.15	Fall
2700	4.296	-3.76	6.03	-0.45	-6.57	-10.55	-7.42	7.1204634	14.474937	400	85.17	Fall
2701	4.296	-3.76	6.03	-0.45	-4.77	-9	-6.99	7.1204634	12.353663	400	86.26	Fall
2702	4.296	-3.85	6.26	-0.59	-4.77	-9	-6.99	7.3728013	12.353663	400	87.35	Fall
2703	4.297	-3.87	6.42	-0.73	-4.77	-9	-6.99	7.5316796	12.353663	400	88.41	Fall
2704	4.297	-3.87	6.42	-0.73	-3.14	-8.21	-4.97	7.5316796	10.097753	400	89.43	Fall
2705	4.297	-3.85	6.48	-0.87	-3.14	-8.21	-4.97	7.5874763	10.097753	400	90.36	Fall
2706	4.298	-3.85	6.48	-0.87	-1.54	-7.7	-3.01	7.5874763	8.409619	400	91.18	Fall

smartphone. A Fossil generation 5 Wear OS smartwatch¹ and One plus 7 smartphones² with a three-axis linear accelerometer and gyroscope were used for the data collection. A_x , A_y , A_z , W_x , W_y , and W_z are the six data columns shown in Table 6.1. In order to collect data, the time series approach was used, and the data capture rate was 419 Hz. Orientation and angular rotation were determined by gyroscope sensors, while linear accelerometer sensors identified fall or non-fall states. In addition to labels for fall and non-fall, the dataset contained information on the time periods and human movement positioning along three axes. Data processing, fuzzy sets, and rules are the same as what is explained in Chapter 6 Sections 4.2.3 and 4.2.4.

¹More details are available from:<https://www.fossil.com/en-gb/watches/learn-more/gen-6-smartwatches/>

²More details are available from:<https://www.oneplus.com/uk/7pro/>

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

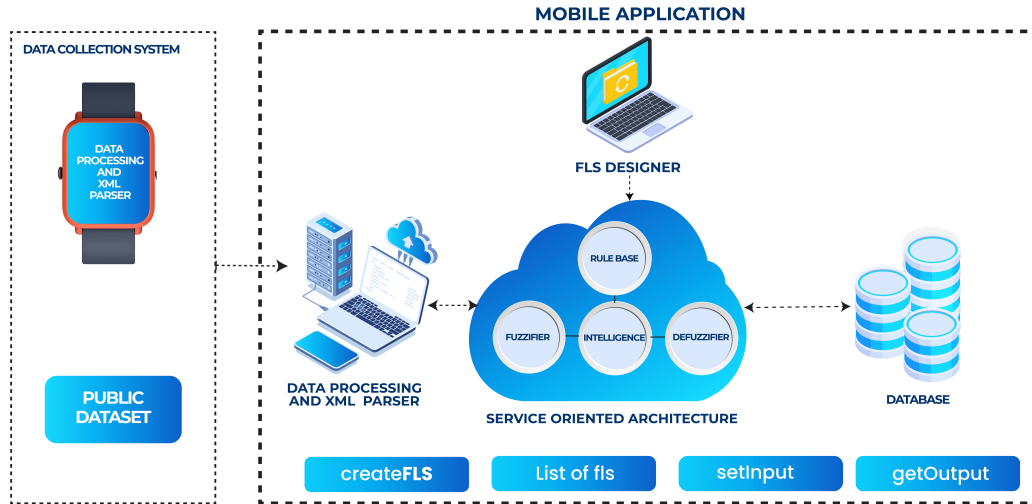


Figure 6.1: System architecture of a standalone fuzzy system using the Mobile application.

6.2.2 Fuzzy Logic System Design

In this subsection, two alternative architectures of implementing the designed FLS for fall detection are described: the standalone and the distributed FaaS architectures. The time required for processing data locally by the smartwatch and the smart phone is to be compared to the processing time required by the cloud-based FaaS.

6.2.2.1 Stand-alone Architecture

The system architecture of a standalone fuzzy system using a mobile application is presented in Figure 6.1. This architecture uses smartwatch and mobile applications to create output based on real-time input supplied to the system. The stand-alone system uses all the FaaS architectural components but instead of being distributed, they are locally implemented on the smartwatch and/or the smartphone, so that the FLS processing is to be handled by the local devices. As shown in Figure 6.1, this architecture takes input directly from wearable devices or input data from any public data set.

When constructing fuzzy logic systems, the fuzzy rules and the member

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

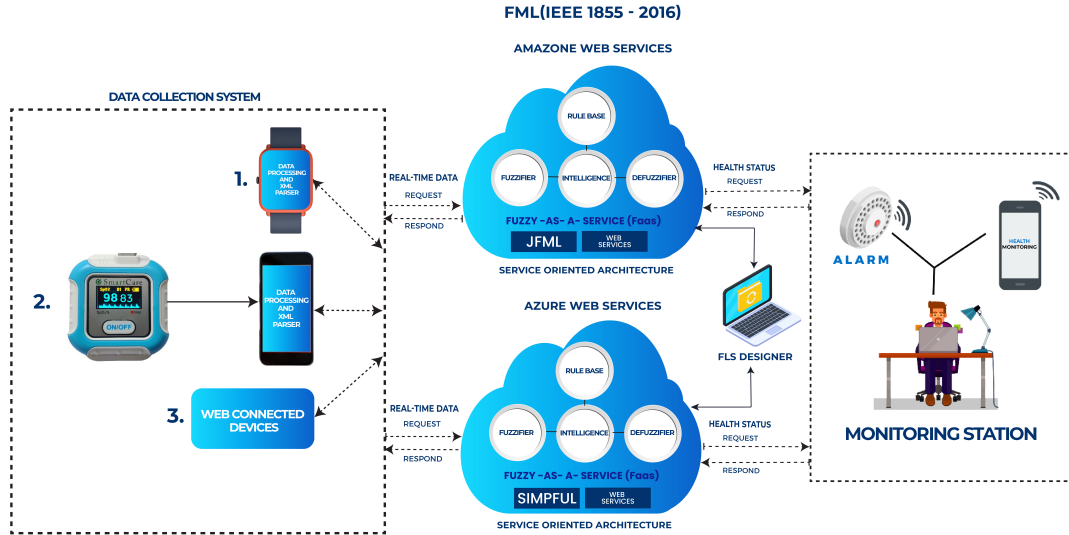


Figure 6.2: The suggested architecture includes data collecting, data processing, feature extraction, and a fuzzy logic system.

variables were specified and exchanged between components in FML format. The program allows a user to enter data in one of three ways: XML, CSV, and sensor data-based. The raw data will be processed and parsed in XML or CSV format or be taken directly through the device sensors at fixed intervals. The FLS contains all of the necessary information and values and a unique URI. The FLS comprises two major components: a knowledge base and a rule base. Once the FLS is formed and listed, the XML data provided by the user can be configured using the `setInput()` - the API function of the FaaS as described in Chapter 3. Similarly, the `getOutput()` returns the desired result based on the FLS and the input provided using the `setInput()`. It creates outcomes based on the rules supplied to the fuzzy system.

6.2.2.2 Cloud-based Architecture

The proposed cloud-based FaaS architecture for fall detection includes distributed blocks for data acquisition, data processing, and fuzzy logic system shown in Figure 6.2. This architecture takes input directly from wearable devices connected to the Web. Alternatively, if the wearable devices have limited connectivity to

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

the Web, data from wearable devices can be passed on to mobile applications or web-connected devices (e.g., via Bluetooth). This real-time data is transmitted over the cloud using the IEEE 1855-2016 standard. This system architecture is implemented over the cloud using the Amazon Web Service with JFML as Java library, or alternatively using Azure Web Service with Simplful as Python library. The real-time data will be passed through either of these modules. The real-time data will be fuzzified, rule-based decisions will be taken, and later the data will be defuzzified, and the fall/non-fall status will be passed on to the monitoring station(s). If required, the monitoring station(s) will raise the alarm to other connected devices based on the monitored status.

6.2.3 Non-FLS Machine Learning Design

In addition to comparing the performance between standalone and distributed architectures, the performances of both FLS architectures can also be compared with the performance of a non-FLS machine learning method (in a standalone mode). This comparison can potentially identify the limitations of stand-alone architectures in high sample rates for both FLS or non-FLS. For this purpose, similar experimental settings to the machine learning approach explained in Chapter 6 is designed. The Random Forest classification method showed its superior accuracy in Chapter 6, therefore it is chosen in this chapter as the non-FLS method for fall detection. Similarly, the dataset was arbitrarily divided into a 70%:30% split and randomly assigned to training and testing.

Non-FLS machine learning is a crucial alternative for finding intrinsic or internal data properties and predictions. On the other hand, non-FLS machine learning is not well suited for dynamic factors in a real-time situation. It can only forecast using previously taught factors, and re-learning is required every time a new element is added. In contrast, an FLS may learn incrementally and interactively in real-time. Another disadvantage of most non-FLS methods is the lack of interpretability, exacerbated by the difficulty in explaining how they work. Furthermore, having sufficient data for training is a significant challenge in machine learning. FLSs can be created with a relatively smaller dataset than non-FLS, because non-FLS machine learning is susceptible to over-fitting and

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

under-fitting, and accuracy testing may misguide the model's performance with unseen data. On the other hand, FLS is unaffected by under-fitting or over-fitting issues. In the case of an enormous amount of data, non-FLS machine learning methods may necessitate a higher computational power during the learning process in contrast to FLSs.

6.2.4 UI Design

The UI screen for the Android mobile application is shown in Figure 6.3 and various methods of interaction with the system such as Create FLS, Set Input, and Get Output is shown in Figure 6.4. This system is able to generate output either based on non-real-time input supplied to the system in the form of an XML or CSV file or based on real-time input supplied to the system directly from device sensors such as accelerometer and gyroscope etc. The significant advantage of this feature for the purpose of this experiment is that it can work in offline mode with different simulated sample rates.

6.3 Experimental Results

This section provides the designed systems' responsiveness results for different combinations of FLS/non-FLS and distributed/non-distributed solutions tested by data processing in different sample rates.

6.3.1 Offline Standalone FLS

The system was tested with a dataset containing 10,000 samples collected using the smartwatch and the mobile phone. It was observed that the application cannot handle more than 5000 samples using the smartwatch. The time required by the smartwatch to process 5000 samples is 10 minutes and 1 second. It took 2 minutes and 59 seconds using a mobile phone to process 10,000 samples. The result might vary depending on the device and Android specifications.

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

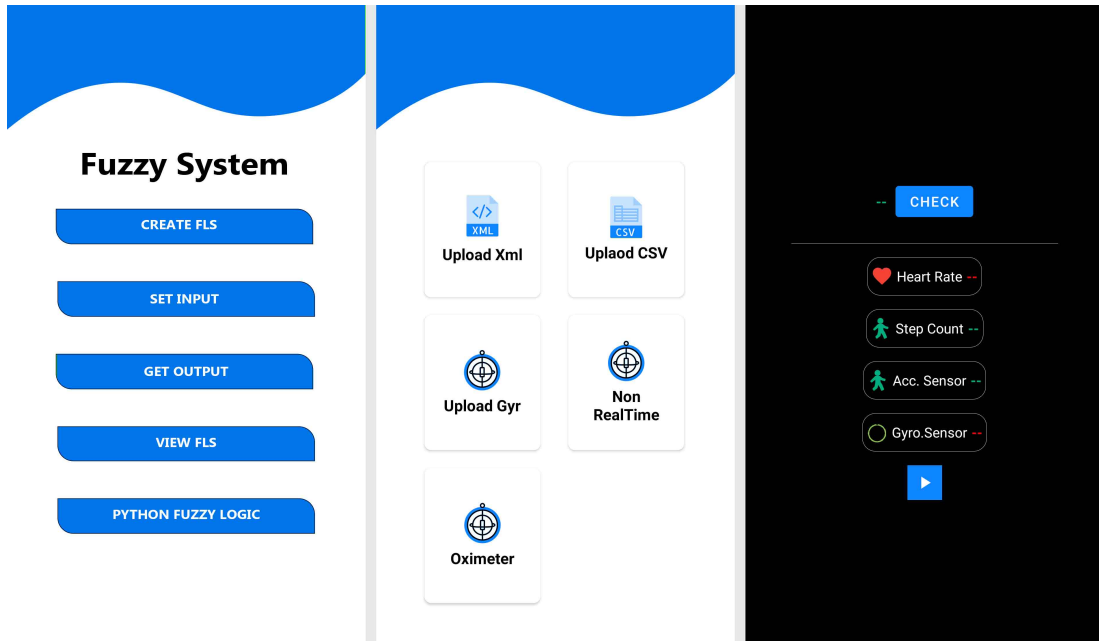


Figure 6.3: UI screen for Android mobile application.

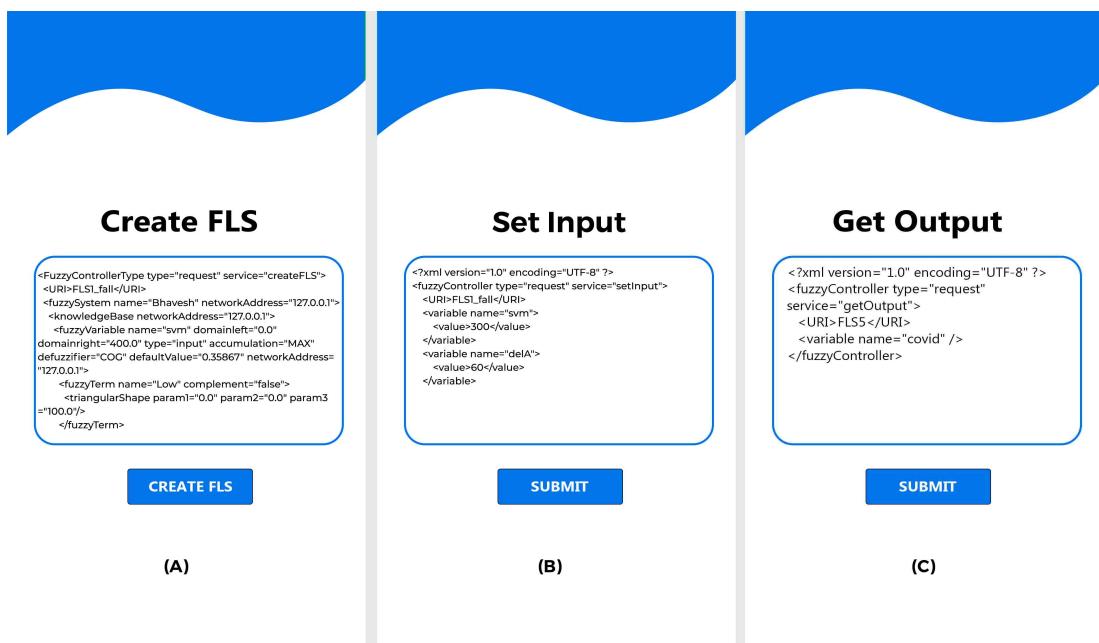


Figure 6.4: Real-time human health monitoring and processing result using A) Create FLS B) Set Input C) Get Output methods.

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

6.3.2 Offline Non-FLS Machine Learning

The system was tested with a file containing 10,000 samples using the smartwatch and mobile phone. It was observed that the time required to process and generate results is 37 minutes and 02 seconds on the smartwatch and 1 minute 41 seconds on the mobile phone using the Random Forest classifier.

6.3.3 Real-time Cloud-based FLS

The accelerometer and gyroscope sensors were used to directly fetch data in real-time at certain time intervals. Testing was performed in a real-time environment with various sample rates in an increasing manner. The obtained data (i.e., 10,000 samples) was fed to AWS and Microsoft Azure. Figure 6.5 shows the snapshot of the smartwatch for different methods, namely Non-FLS machine learning Real-time FLS with Azure/Simpful, Real-time FLS with AWS/JFML, and standalone FLS.

The observed results are as follows:

1. AWS: It is possible to construct virtual computers and operate them on one of Amazon's data centres using the Elastic Compute Cloud (EC2) service. Through AWS with JFML, 10,000 samples were processed, and the desired results were generated using the smartwatch in 18 minutes 31 seconds, and 45 seconds using the mobile phone.
2. Microsoft Azure: Through Microsoft Azure with Simpful, 10,000 samples were processed, and the desired results were generated within 20 minutes, 50 seconds using the smartwatch, and 1 minute 37 seconds using the mobile phone.

6.4 Discussion

A comparison of detection efficiencies has been made among the existing fuzzy logic-based standalone architectures and non-fuzzy ML-based decision-making methods with the implemented architecture. The study included the use of four

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection



Figure 6.5: Snapshot of fall detection output using smartwatch A) Non-fuzzy machine learning using cloud B) Fuzzy with simplful libraries and Azure C) Fuzzy with JFML libraries and AWS D) Fuzzy standalone.

approaches which are elaborated in Table 6.2, i.e., machine learning (non-fuzzy), fuzzy standalone, real-time fuzzy with AWS/JFML, and real-time fuzzy with Azure/Simplful.

The summary of analysis, advantages, and disadvantages of the four approaches are shown in Figure 6.6 and Table 6.3. It can be seen that the cloud-based FaaS solution (based on AWS/JFML) has achieved a significantly lower average processing time (45 seconds per 10K samples) compared to the other three approaches.

A more detailed comparison between the processing time taken by the non-FLS and FLS approaches using the smartwatch is shown in Figure 6.7(a) and Table 6.4. Fig. 6.7(A) shows the difference between the processing time that is taken for the collected data and given as an input to the ML model (Non-fuzzy) and Fuzzy model. It was observed that for average processing times of 1000 records, 2000 records, 3000 records, 4000 records, 5000 records, 7500 records and 10000 records, the time taken was 1428 seconds, 1538 seconds, 1497 seconds, 1481 seconds, 1622 seconds, 2027 seconds and 2220 seconds, respectively for ML (Non-fuzzy) model, whereas for Fuzzy (Standalone) model, average processing times were 192 seconds, 333 seconds, 441 seconds, 533 seconds, 600 seconds. The evidence shows that application were crash for above 5000 records using standalone system, whereas Fuzzy model (AWS with JFML), average processing

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

Table 6.2: Comparative analysis of real-time and non-real-time human fall detection.

Sr. No.	Model	File format	Advantages	Disadvantages	Type of user	Algorithm used
1.	Machine learning model (Non-fuzzy)	.csv and sensor data	Model training is performed over cloud and the five classes of a fall detection can be possible.	Model training is essential if user requirements change.	Single user	Random Forest Classifier
2.	Stand-alone Fuzzy model	.csv and sensor data	Processing time taken is comparatively less when data sending rate is increased.	Since it is a non-real-time system, only a single user can view results.	Single user	Mamdani, TSK, Tsukamoto and AnYa
3.	Real-time fuzzy model with AWS and JFML	xml, .csv and sensor data	Processing time is less as compare to other methods. Four different types of inferences are used, namely Mamdani and Assilian, Takagi - Sugeno - Kang (TSK), Tsukamoto and AnYa.	Dependency over cloud.	Multi user	Mamdani, TSK, Tsukamoto and AnYa
4.	Real-time fuzzy model with Microsoft Azure and Simpful library	xml, .csv and sensor data	It offers results in real-time with the help of Azure and a Simpful library with a tolerable amount of processing time per data.	Processing time taken is more as compared to AWS with JFML models.	Multi user	Mamdani and TSK

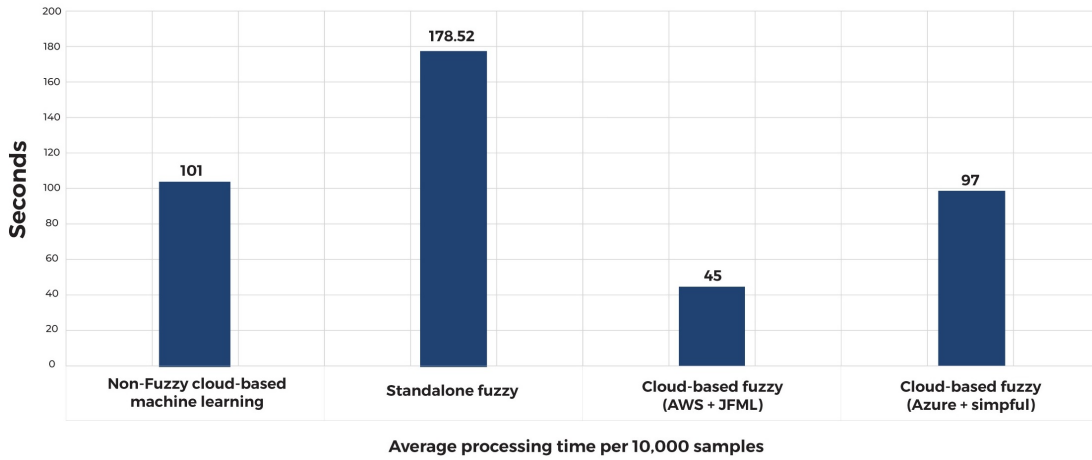


Figure 6.6: Comparative analysis of human fall detection.

Table 6.3: Summary of real-time and non-real-time human fall detection system.

Sr. No.	Model	Average Processing time per 10,000 data using smartwatch	Average Processing time per 10,000 data using smartphone	Number of samples processed per second using smartwatch	Number of samples processed per second using smartphone
1	Machine learning model (Non-fuzzy)	2220	101	4.5	99
2	Stand-alone Fuzzy model	∞	179.42	∞	56.68
3	Real-time fuzzy model with AWS and JFML	1111	45	9	222
4	Real-time fuzzy model with Microsoft azure and Simpful Library	1250	97	8	103

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

Table 6.4: Number of samples processed per second using the smartwatch.

Sr. No	Model	1000 samples	2000 samples	3000 samples	4000 samples	5000 samples	7500 samples	10000 samples
1.	Machine learning model (Non-fuzzy)	0.7	1.3	2.1	2.7	3.07	3.7	4.5
2.	Stand-alone Fuzzy model	5.2	6	6.8	7.5	8.33	Appl crash	Appl crash
3.	Real-time fuzzy model with AWS and JFML	4.7	5.3	5.8	6.25	6.7	7.67	9
4.	Real-time fuzzy model with Microsoft Azure and Simplful	1.96	3.38	4.47	5.55	6.25	7.5	8

times were 212 seconds, 377 seconds, 517 seconds, 640 seconds, 746 seconds, 977 seconds, and 1111 seconds and for Fuzzy model (Azure with Simplful), average processing times were 510 seconds, 590 seconds, 670 seconds, 720 seconds, 800 seconds, 1000 seconds and 1250 seconds, respectively. Similarly, Figure 6.7(b) and Table 6.5 shows the number of samples that can be processed per second by the smartwatch, when the number of samples is gradually changed from 1000 to 10000 samples. Fig. 6.7(B), the number of records processed per second was calculated by supplying 1000 records, 2000 records, 3000 records, 4000 records, 5000 records, 7500 records and 10000 records. As shown in chart, 0.7 records /sec, 1.3 records /sec, 2.1 records /sec, 2.7 records /sec, 3.07 records /sec, 3.7 records /sec, and 4.5 records /sec respectively for ML (Non-fuzzy) model, whereas for Fuzzy (Standalone) model, 5.2 records /sec, 6 records/sec, 6.8 records/sec, 7.5 records/sec and 8.33 records/sec whereas Fuzzy model (AWS with JFML), 4.7 records/sec, 5.3 records/sec, 5.8 records/sec, 6.25 records/sec, 6.7 records/sec, 7.67 records/sec and 9 records/sec and for Fuzzy model (Azure with Simplful), 1.96 records/sec 3.38 records/sec, 4.47 records/sec, 5.55 records/sec, 6.25 records/sec, 7.5 records/sec and 8 records/sec.

For the smartphone, Figure 6.8(a) and Table 6.6 show Fig. 7 A) shows the difference between the processing time that is taken for the collected data and given as input to the ML model (Non-fuzzy) and Fuzzy model. It was observed

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

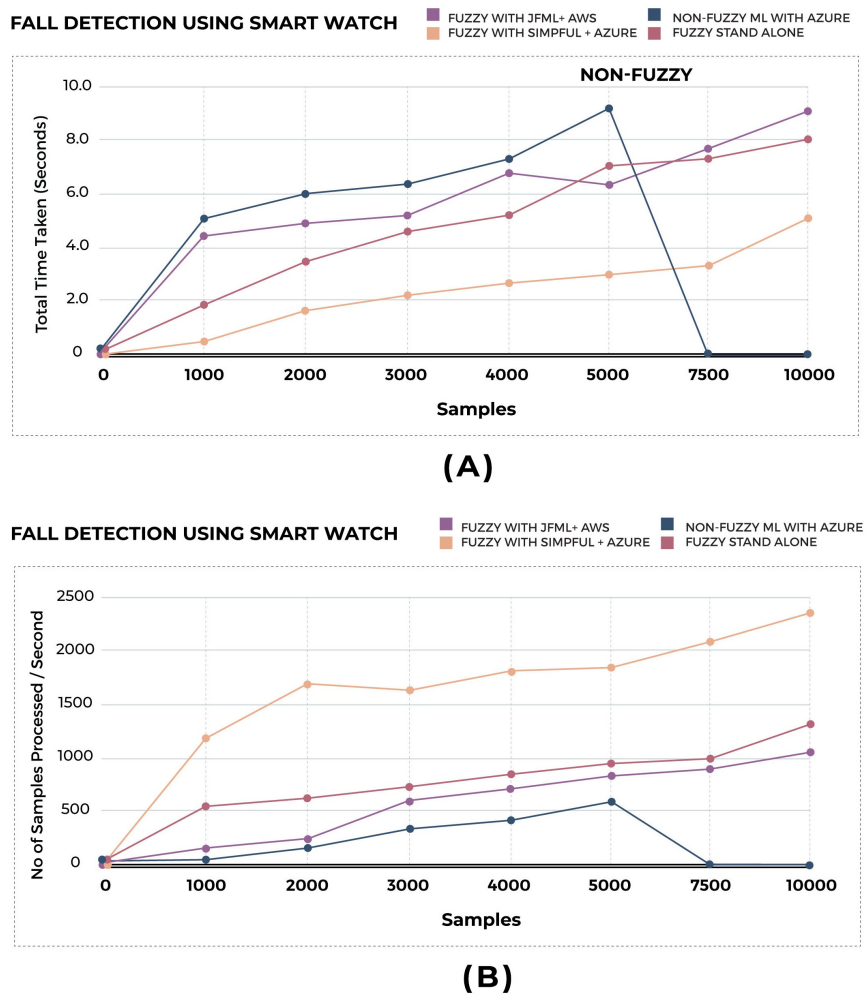


Figure 6.7: Fall detection using smartwatch A) Total time required to process the samples B) Number of samples processed per second.

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

Table 6.5: Average processing time (in seconds) with respect to number of samples using the smartwatch

Sr. No.	Model	1000 samples	2000 samples	3000 samples	4000 samples	5000 samples	7500 samples	10000 samples
1.	Machine learning model (Non-fuzzy)	1428	1538	1497	1481	1622	2027	2220
2.	Stand-alone Fuzzy model	192	333	441	533	600	App crash	App crash
3.	Real-time fuzzy model with AWS and JFML	212	377	517	640	746	977	1111
4.	Real-time fuzzy model with Microsoft Azure and Simplful	510	590	670	720	800	1000	1250

Table 6.6: Average processing time (in seconds) with respect to number of samples using smart phone

Sr. No	Model	1000 samples	2000 samples	3000 samples	4000 samples	5000 samples	7500 samples	10000 samples
1.	Machine learning model (Non-fuzzy)	35.7	50	58.82	67	71.42	89.29	101
2.	Stand-alone Fuzzy model	41.67	74.07	96.77	114	125	159.71	179.42
3.	Real-time fuzzy model with AWS and JFML	15.38	24.69	30.61	32	34.96	39.68	45
4.	Real-time fuzzy model with Microsoft Azure and Simplful	31.25	44.44	52.63	63	65.8	82.41	97

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

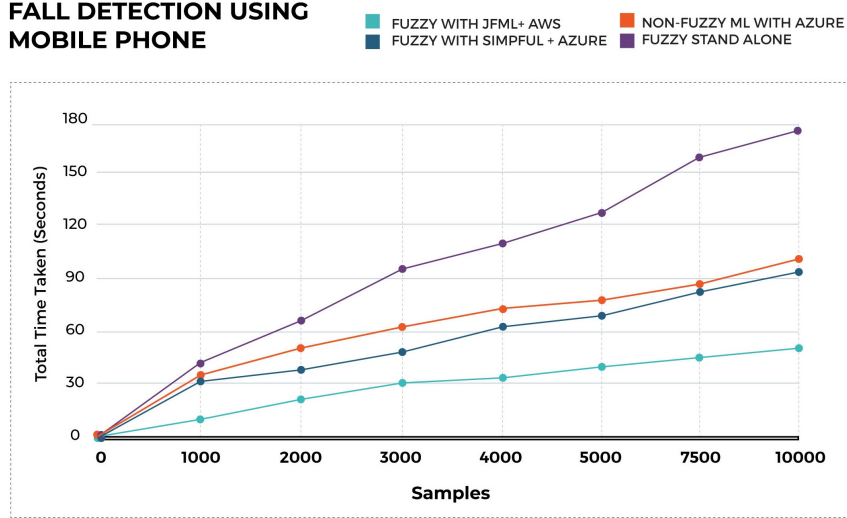
that for average processing times of 1000 records, 2000 records, 3000 records, 4000 records, 5000 records, 7500 records and 10000 records, the time taken was 35.7 seconds, 50 seconds, 58.82 seconds, 67 seconds, 71.42 seconds, 89.29 seconds and 101 seconds, respectively for ML (Non-fuzzy) model, whereas for Fuzzy (Standalone) model, average processing times were 41.67 seconds, 74.07 seconds, 96.77 seconds, 114 seconds, 125 seconds, 159.71 seconds and 179.42 seconds, whereas Fuzzy model (AWS with JFML), average processing times were 15.38 seconds, 24.69 seconds, 30.61 seconds, 32 seconds, 34.96 seconds, 39.68 seconds, and 45 seconds and for Fuzzy model (Azure with Simplful), average processing times were 31.25 seconds, 44.44 seconds, 52.63 seconds, 63 seconds, 65.8 seconds, 82.41 seconds and 97 seconds, respectively. Similarly, the average time taken for processing 1000 to 10000 samples are shown in Figure 6.8(b) and Table 6.7. The number of records processed per second was calculated by supplying 1000 records, 2000 records, 3000 records, 4000 records, 5000 records, 7500 records and 10000 records. As shown in the chart, 28 records /sec, 40 records /sec, 51 records /sec, 59.7 records /sec, 70 records /sec, 84 records /sec, and 99 records /sec respectively for ML (Non-fuzzy) model, whereas for Fuzzy (Standalone) model, 24 records /sec, 27 records/sec, 31 records/sec, 35.08 records/sec, 40 records/sec, 49.9 records/sec, and 56.68 records/sec whereas Fuzzy model (AWS with JFML), 65 records/sec, 81 records/sec, 98 records/sec, 125 records/sec, 143 records/sec, 189 records/sec and 222 records/sec and for Fuzzy model (Azure with Simplful), 32 records/sec 45 records/sec, 57 records/sec, 63.5 records/sec, 76 records/sec, 91 records/sec and 103 records/sec.

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

Table 6.7: Number of samples processed per second using smart phone

Sr. No	Model	1000 samples	2000 samples	3000 samples	4000 samples	5000 samples	7500 samples	10000 samples
1.	Machine learning model (Non-fuzzy)	28	40	51	59.7	70	84	99
2.	Stand-alone Fuzzy model	24	27	31	35.08	40	49.9	56.68
3.	Real-time fuzzy model with AWS and JFML	65	81	98	125	143	189	222
4.	Real-time fuzzy model with Microsoft Azure and Simpful	32	45	57	63.5	76	91	103

FALL DETECTION USING MOBILE PHONE



FALL DETECTION USING MOBILE PHONE

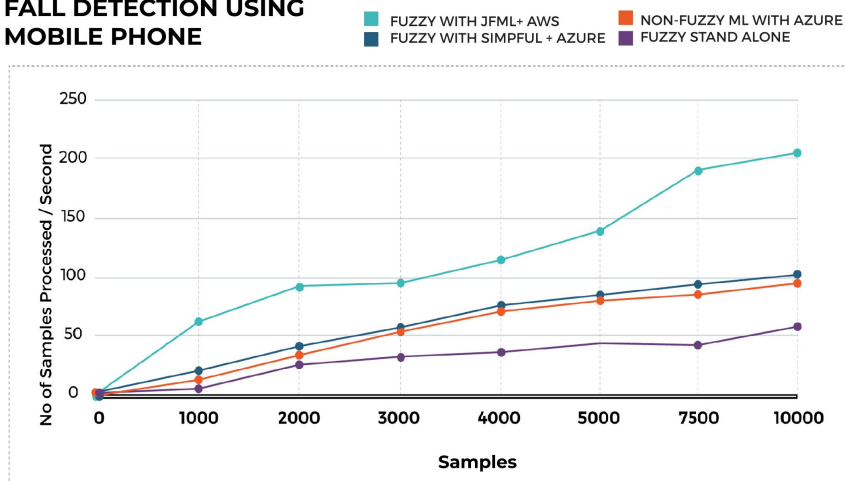


Figure 6.8: Fall detection using mobile phone A) Total time required to process

6.5 Chapter Summary

In this chapter, a comparative analysis between standalone and cloud-based systems was performed wherein different approaches were used. Based on these findings, it can be concluded that the suggested FaaS can effectively manage real-time data using cloud computing with datasets for human fall detection using a fuzzy rule-based system utilising FML. The developed system was regarded as having relatively good efficiency and response time. The architecture may be extended to a broader region even though it was designed with human fall detection. Additionally, this study attempted to make it easier for clients to flexibly transmit relatively complex computational tasks to dedicated servers. Utilising vitalised cloud services offers the devices a remarkable level of flexibility. To demonstrate the value of fuzzy logic and IEEE-1855 in a practical setting, such as forensic science, future work on an artificial intelligence system for investigating crime scenes and the automatic reconstruction of criminal dynamics will build on the recommended architecture. Additionally, fuzzy services like fuzzy ontologies and fuzzy querying of fuzzy databases can be taken into account. A suitable alternative for implementation would be to expand web services to the Semantic Web due to the close relationship between FML and fuzzy ontologies. In addition to the primary advantage of FaaS, this type of architecture offered several other advantages, including the ability to share networks, maintain control over hardware and software, reuse data, provide load balancing among FLS devices, and reduce costs.

The limitation of the research sheds light on the potential of fuzzy logic web services and machine learning techniques in this domain. Firstly, the study may not thoroughly explore the robustness and reliability of the proposed fall detection systems in diverse real-world environments and conditions. Additionally, the comparison between fuzzy logic web services and machine learning methods could be further nuanced to consider factors such as computational efficiency, scalability, and adaptability to varying sensor data characteristics. Future studies could aim to address these gaps by conducting comprehensive performance evaluations across diverse datasets and sensor setups, as well as exploring hybrid approaches

6. A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection

that leverage the strengths of both fuzzy logic and machine learning techniques.

Chapter 7

Conclusions and Future Scopes

7.1 Thesis Summary

The primary objective of the current research work is to develop a novel and standard architecture for designing distributed fuzzy logic systems, with a particular focus on human activity/health monitoring applications. The research was motivated by the limitations and drawbacks of the classical approach in implementing stand-alone complex fuzzy logic systems.

In order to meet the objectives set to reach the target, the following steps were taken, and accordingly, relevant experiments were performed:

- A comprehensive research on existing tools and techniques and related research work is conducted.
- A service-oriented and web/cloud-based architecture for addressing the associated problems with uncertain data processing in HAR exploring the applications and possibly extending the standard web communication protocols for fuzzy logic systems, i.e., IEEE 1855-2016 and its associated software libraries, are developed (called Fuzzy-as-a-Service or FaaS).
- The capabilities of the developed methodology in processing uncertain data for decision-making support scenarios in HAR are investigated and compared with alternative solutions. The results of the experiments are summarised in the next section. In Section 8.3, the project's initial

objectives are revisited in detail. Finally, in Section 8.4., some directions for future research are provided.

7.2 Results and Discussion

Managing data in an intense FLS on top of developed web services with a more sampling rate is essential in real-world situations. Three different experiments were performed and were reported.

Experiment 1 (Chapter 4): A Cloud-based FaaS for Fall Detection to Recognize Human Activities. This experiment comprised wearable sensors (i.e., accelerometers and gyroscopes) that promoted a practical HAR scenario using FaaS. Research findings exhibited that the proposed method could effortlessly differentiate between fall and non-fall occurrences with an accuracy, sensitivity, and specificity of 90%, 88.89%, and 91.67%, respectively. The results were also compared with some other machine learning methods. This experiment demonstrated the effectiveness of using the proposed FaaS in yet another practical and important HAR scenario.

Moreover, the responsiveness and accuracy of the proposed FaaS architecture are examined in an intense HAR scenario. A walking/running classification experiment using an offline public dataset was designed utilising an FLS with 247 rules over 7 fuzzy sets. The system was implemented on the cloud where it had to serve two clients' back-to-back requests in parallel. The results showed that the system was responsive at a more sampling rate that would be necessary for a real-world scenario over a maximum of 1-second turn-around delay without missing any of the coming requests. The developed fuzzy logic web service system also attained an accuracy of 97.42%. This showed achieving high accuracy and responsiveness in a relatively intense real-time HAR data processing scenario.

Experiment 2 (Chapter 5): A Cloud-Based FaaS for Pervasive Health Conditions Monitoring. This experiment used wrist pulse and oximeter sensors to gather real-time data from individuals in order to predict their health status using two alternative cloud-based FaaS architectures. The data was passed to alternative cloud servers through an Android application

and the responses were received on the same device based on a fuzzy rule-base. Although the designed FLS has less complexity than that of Experiment 1, it shows the capabilities of FaaS in a real-world setting where the actual individuals' data is collected and processed in real-time. The results showed a processing time between 0.306 to 0.456 seconds per sample for alternative cloud/library platforms (AWS/JFML vs. Azure/Simpful). A noticeable highlight of the experiment is to demonstrate the cross-platform and openness capabilities of the proposed FaaS, mainly due to the adaptation of IEEE 1855 standard, so that the same local devices can communicate with different FaaS servers regardless of the employed cloud platform and/or software libraries. The performed experiment is a proof-of-concept that aims to achieve an instantaneous FLS execution via cross-platform web services, which to our knowledge, has had no similar implementation record.

Experiment 3 (Chapter 6): A Comparative Study of Standalone and Cloud-based Fuzzy Logic Systems for Human Fall Detection. This experiment extends Experiment 4 in fall detection, by performing a comparative analysis of the time-responsiveness between different combinations of fuzzy/non-fuzzy and cloud-based/stand-alone approaches by sensing the individual's movements through a smartwatch and a mobile phone. Based on the findings, it can be concluded that the suggested cloud-based FaaS can manage real-time data with relatively better efficiency and response time. Particularly, it was shown that while the local solutions struggle to keep up with processing data in high sample rates, the FaaS implemented on the AWS platform has the best response time among the other local solutions (i.e., processing 10K samples takes 45 seconds in FaaS, as opposed to 178 seconds in a local FLS).

7.3 Conclusions

This project presents a novel approach towards a web-based service-oriented FLS architecture (FaaS) as an example of the implementation/extension of the IEEE 1855-2016 standard. Based on the experiments performed, the proposed FaaS can perform real-time data processing by means of cloud computing with human

activity monitoring datasets using a distributed form of fuzzy logic systems. Even though the architecture is specified in terms of Aml environments, it can be expanded to a broader area, e.g., in which local computation power reaches its limits in processing real-time data and/or open access to the shared FLS resources is required.

This study aimed at facilitating the flexible delivery of the relatively complex computing that might be required for complex FLSs from clients to dedicated servers. The efficiency and response time of the developed system were considered high. The use of virtualised cloud services provided distinctive elasticity to the system. Fundamentally allowing universally accessible FaaS, other advantages of such architecture included network sharing, hardware/software control, data reuse, load balancing amongst FLS devices, and cost-efficiency. The unique feature of the study was the use of an IEEE 1855-2016 algorithm in real-time, and a novel extension of the standard so that it can be used not only for defining FLS specifications, but also for data communication between different components of the distributed FLSs.

7.3.1 Revisiting the Objectives

The present research has been initiated by the question: “Can a distributed fuzzy logic system provide any advantage in making decisions based on processing uncertain data in HAR over the current stand-alone and/or non-fuzzy systems?” In order to address the above question, the following objectives are discussed with their respective answers:

1. To conduct comprehensive research on existing tools and techniques and related research work.

Reflection: A comprehensive research on existing tools and techniques in the referred domain has been carried out and is reported in Chapter 2 in a systematic manner.

2. To develop a service-oriented and web/cloud-based architecture for addressing the associated problems with uncertain data processing. This will involve exploring the applications and possibly extending the

standard web communication protocols for fuzzy logic systems, i.e., IEEE 1855 (2016) and its associated software libraries.

Reflection: Uncertain data processing is the main hurdle in handling imprecise data generated by the sensors attached to the person under observation. It has been demonstrated that a fuzzy logic system is the main contender in solving such issues. Thus a standard FLS using a suitable fuzzy inference engine has been designed and tested with such imprecise data. It is observed that the efficiencies of FLS-based decision-making systems are more effective compared to their counterparts which are using other ML techniques. However, it is also observed that the architecture involving standalone systems is not capable of handling such a huge amount of data. Thus cloud-based architecture is proposed instead of standalone architecture. Hence a service-oriented architecture in a cloud platform (Azure as well as AWS) is designed and developed with FLS as the decision-making tool. The main aim is to extend the standard web communication protocols. We ultimately used various libraries that implement IEEE 1855-2016 standards.

3. To investigate the capabilities of the developed methodology in processing uncertain data for uncertain decision-making support scenarios in HAR.

Reflection: In Chapter 4, the capabilities of the developed methodology in processing uncertain data for uncertain decision-making support scenarios in HAR are reported. It is shown through experimenting with human activity monitoring datasets that the proposed service-oriented architecture can undertake real-time data processing using a complex fuzzy rule-based system. The accuracy and the response time of the developed system are found to be relatively high. The outcome of the experiment offers a novel approach for a web-based service-oriented FLS architecture.

4. To apply the methodologies developed for fall detection of the persons under observation using suitable sensors.

Reflection: An in-depth analysis of real-time fall detection using a

web-based service-oriented FLS architecture is carried out. It is observed that the proposed FaaS can detect a fall and a non-fall with a fair amount of accuracy. Based on the experiment performed in Chapter 4, it is concluded that FaaS can detect falls from real-time data as it requires minimum hardware and software specifications.

5. To measure the efficiencies of the desired system in monitoring physiological (non-invasive) data obtained from a person.

Reflection: Blood Oxygen Saturation (SpO₂) and heart rate are the two main physiological parameters obtained from the non-invasive methods that are taken up as the input to measure the efficiencies of the system developed. The novelty that followed in this study is that an extension of the IEEE 1855-2016 standard is implemented in real-time for managing sensor information. Moreover, the study includes the use of FaaS and real-time monitoring of activities. Based on the experiment performed in Chapter 5, it is observed that the efficiencies of the systems developed are as expected.

6. To compare the performances of the developed system using FLS and other standard ML-based decision-making processes.
7. To compare the efficacies of a stand-alone system and cloud-based system in realising the support system for HAR.

Reflections regarding Objective 6 and 7: A comparative analysis between standalone and cloud-based systems is performed wherein different approaches are used. Based on these findings, it is concluded that the suggested FaaS can manage real-time data using cloud computing with datasets for human fall detection using a complex fuzzy rule-based framework utilising FML. Based on the experiment performed in Chapter 7, the developed system has relatively good efficiency and response time. In addition to the primary advantage of making FaaS internationally available, this type of architecture offered several other advantages, including the ability to share networks, maintain control over hardware

and software, reuse data, provide load balancing among FLS devices, and reduce costs.

7.4 Directions for Further Research

The practical perspective of the proposed system has many aspects of expansion. The FLS Group is motivated to participate in the design process and provide feedback on feature prioritisation and collaborative growth activities in deployable applications. This reinforces the proposed API schema and/or invocation formats as well as offers more advanced input for other architecture implementations. However, even though this effort is the first stage of our HAR project, there is still more that has to be done for improvement and extension, as shown below:

- In the future, it is necessary to extend FML to cope with more sophisticated HAR scenarios and situations requiring complex interaction mechanisms between agents.
- The study only looked at one type of FLS (i.e., rule-based systems). Other fuzzy services, such as fuzzy querying of fuzzy databases or fuzzy ontologies, may be employed in the future.
- Furthermore, caused of the close relationship between FML and fuzzy ontologies, expanding web services to semantic web services (e.g., developing cloud-based searchable FLS repositories) will make a significant choice.
- In the future, the proposed architecture can also be used as the main framework to design an artificial intelligence system for the analysis of crime scenes and the automatic reconstruction of crime dynamics to highlight the role of fuzzy logic and IEEE-1855 in a critical real-world scenario, such as that of forensic sciences.
- Efforts can also be made to optimise the distributed FML rules on several hosts to minimise the fuzzy inference time.

Appendix A: Development of AWS and Azure Web Services

The process of deploying an application over the cloud is as follows:

1. The deployment of this project will be handled by the cloud service provider AWS (Amazon Web Services).
2. It offers a variety of implementation options for the project. As a result of this, EC2 service is considered for this research.
3. EC2 is a Linux operating system running on an instance of a VM (Virtual Machine) (Operating System)
4. The virtual machine needs to be prepared, which entails installing Java as well as the MySQL database.
5. It is essential to change the network configuration so that users can access port 22 from the internet. So that we can connect to the VM via SSH, we must install the necessary software. For us to meet security requirements, SSH Public Key Authentication is required.
6. To accomplish this, we will need to produce a key pair. The private key will be held in Amazon Web Services (AWS), and a public key will be issued to users wishing to connect to the Virtual Machine (VM).
7. When it comes to constructing artefacts for the Spring boot project, we make use of Maven.

8. With the "mvn clean build" command, the jar file or artefact for this project would be constructed.
9. The generated artefact has to be moved to the instance running on Amazon Web Services EC2.
10. When it has been transferred, we will need to execute the jar within the VM by using the command "java -jar jar-name."
11. In addition to this, we need to ensure that the jar contains the essential database credentials in order for it to be able to connect to the database that has been set up on the EC2 VM.
12. The application may be accessible through the internet by using the public IP configuration.

The API developed over AWS tested by passing FML code to the respective web page. Users can write or paste the FLS code on the creatFLS API shown in Figure 1.

Users are required to have the ability to set their own individual values for the FLS inputs. In addition to this, it is necessary that any number of inputs can be included in a single invocation made by a particular client system. This distributes the burden of data collection across a multi-input multi-output FLS by enabling the system to collect individual or clustered outputs from a variety of devices. This can be done individually or collectively. In spite of the fact that each collected input has the potential to be stored in the server, only the most recent collection will actually be used when the FLS algorithm is put into action, as demonstrated in Figure 2.

The Getoutput API requires the FLS name and an output variable in order to compute and generate output values, as illustrated in the figure 3.

In the future, when the description of an FLS or its input/output history is no longer required, clients should be able to request that it be removed from the list of specified FLSs in the database. After a predetermined amount of inactivity, the device can be programmed to automatically remove the FLSs. Figure 4 shows that FLs can be deleted by passing a unique FLS name.

Appendix A

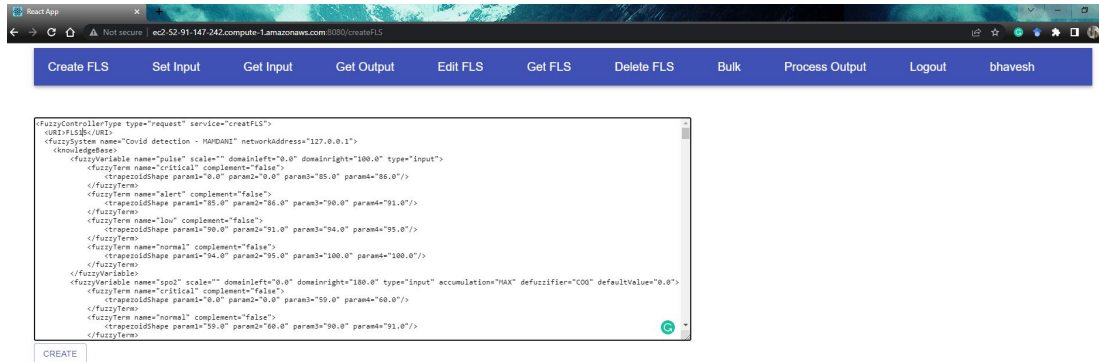


Figure 1: Create FLS.

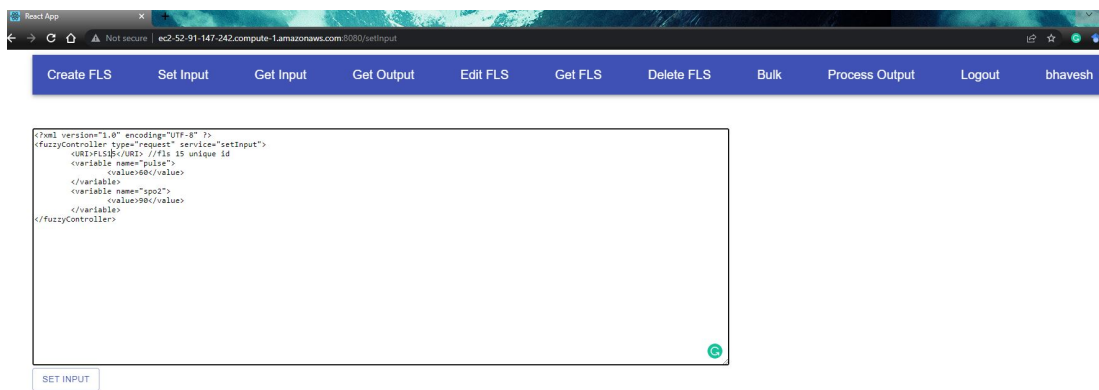


Figure 2: Setinput FLS.

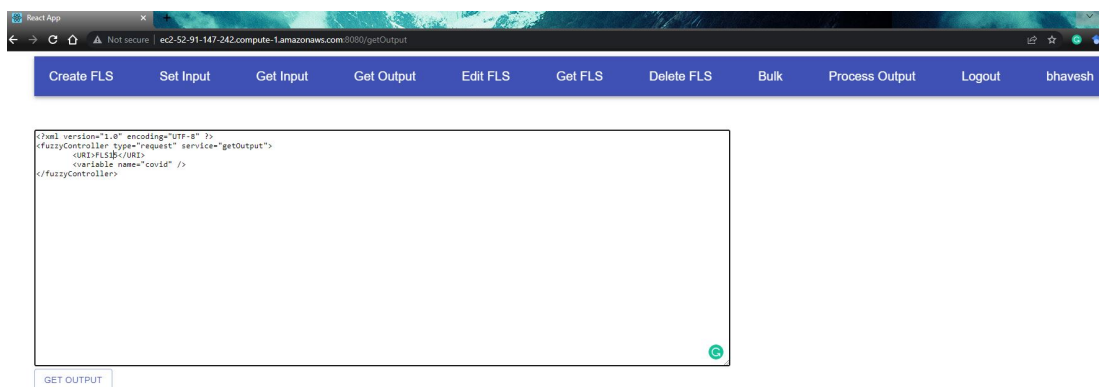


Figure 3: GetOutput FLS.

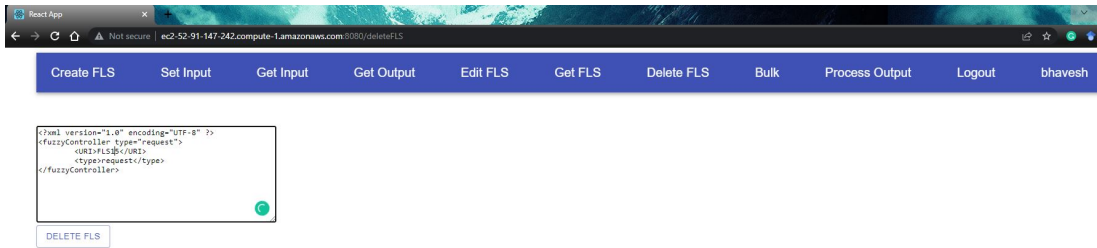


Figure 4: Delete FLS.

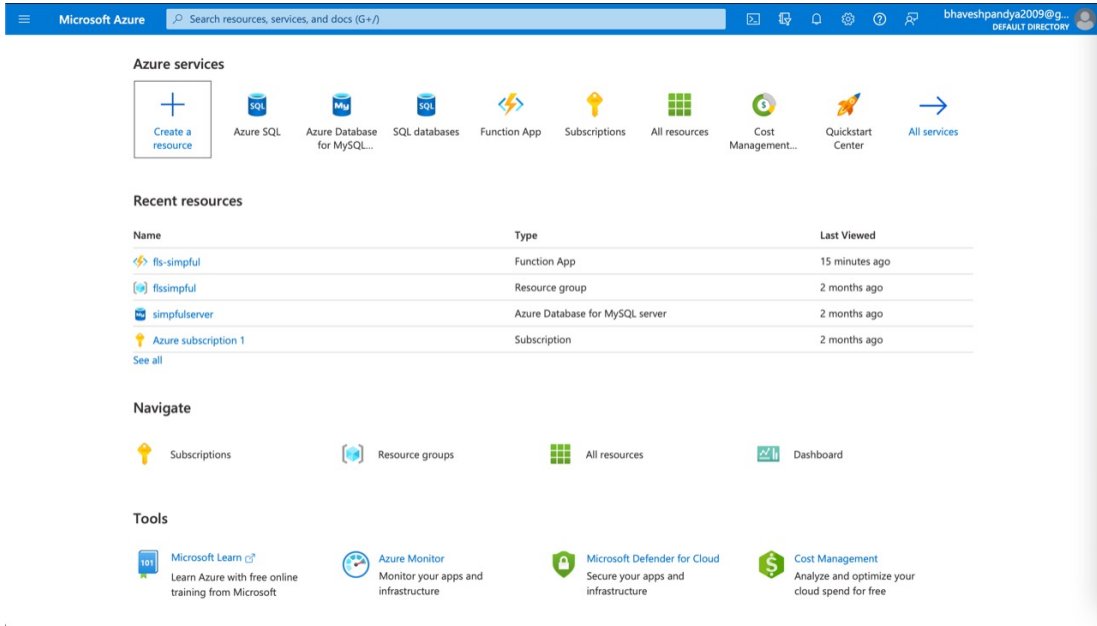


Figure 5: Azure services.

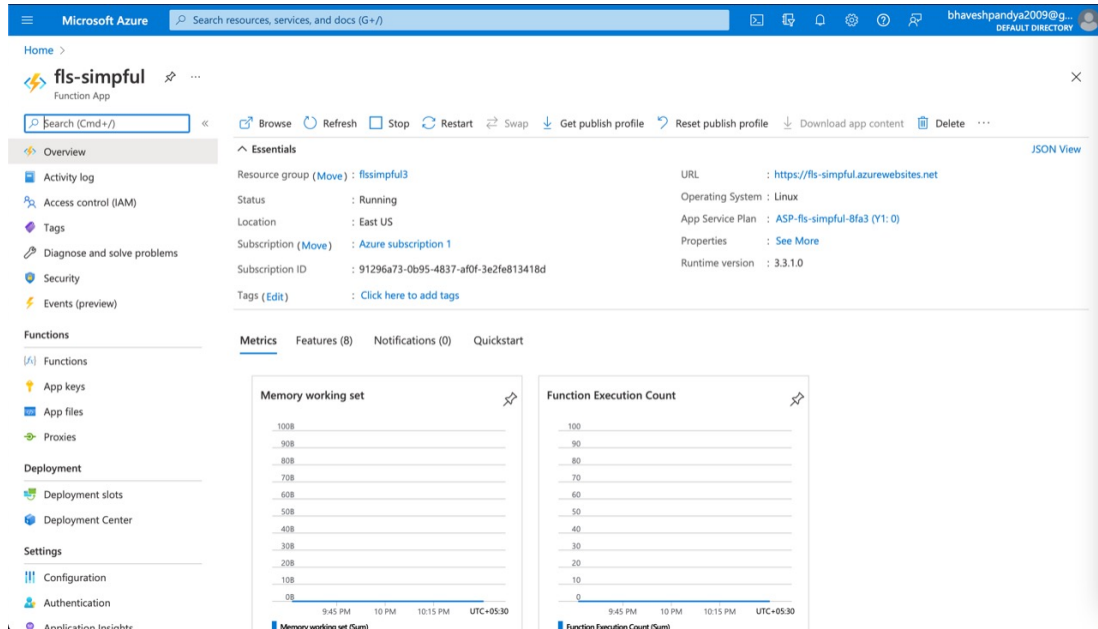


Figure 6: FLS Simplful.

Figure 5 displays the homepage of Azure, which displays the fls-simpful and the Azure subscription. fls-simpful is a function app in which the project’s APIs are present. Simplfulserver stores a MySQL database storing the system’s FLS file.

Figure 6 is the homepage of the fls-simpful function application. The process can be stopped or restarted. After stopping, we will be able to locate the start button. In addition, we have a great deal of additional information about the system, including its operating system, deployment location, URL, etc. Terminating the running process when it is not in use is a good practice.

Figure 7 shows the standalone APIs that were developed for this research. The list of individual functions written for this project is available by clicking on the functions. The APIs developed for this research are as follows:

- postfls: Inserting string XML passed during the "CreateFLS" stage into the database so that it can be used later during evaluation.
- XML inputfls: To save all variables and values.
- getfls: To retrieve the list of FLS that are saved in the system.

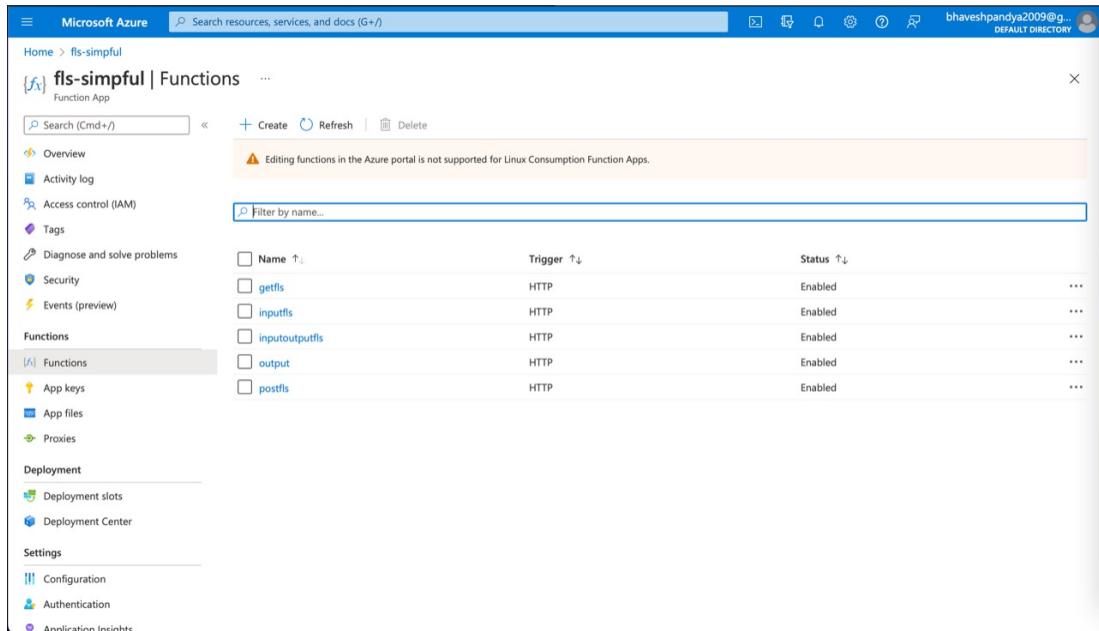


Figure 7: List of functions.

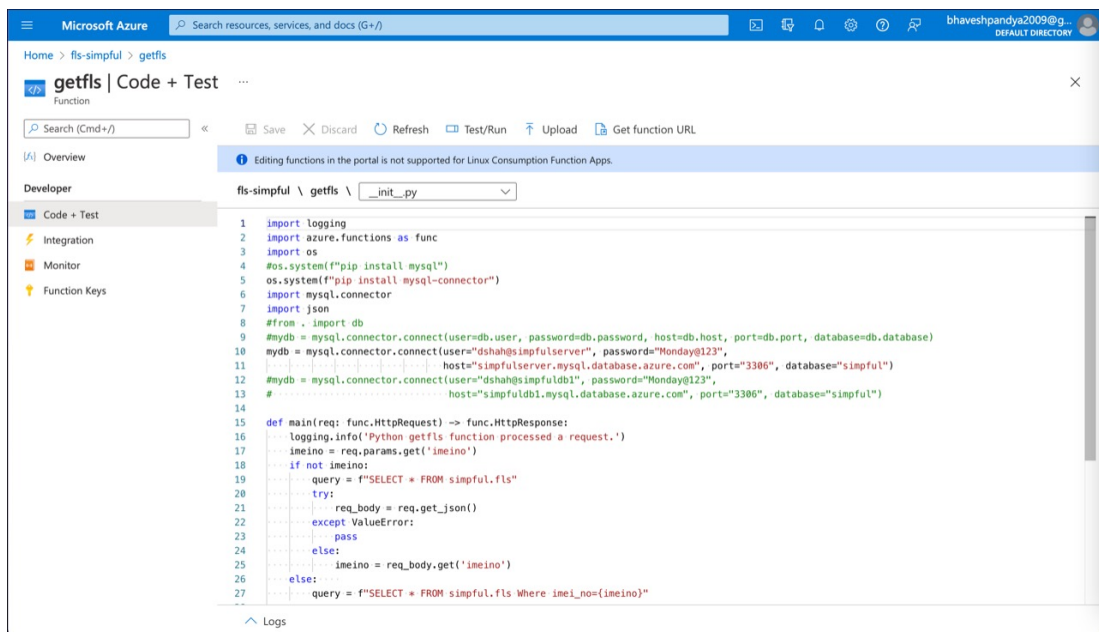
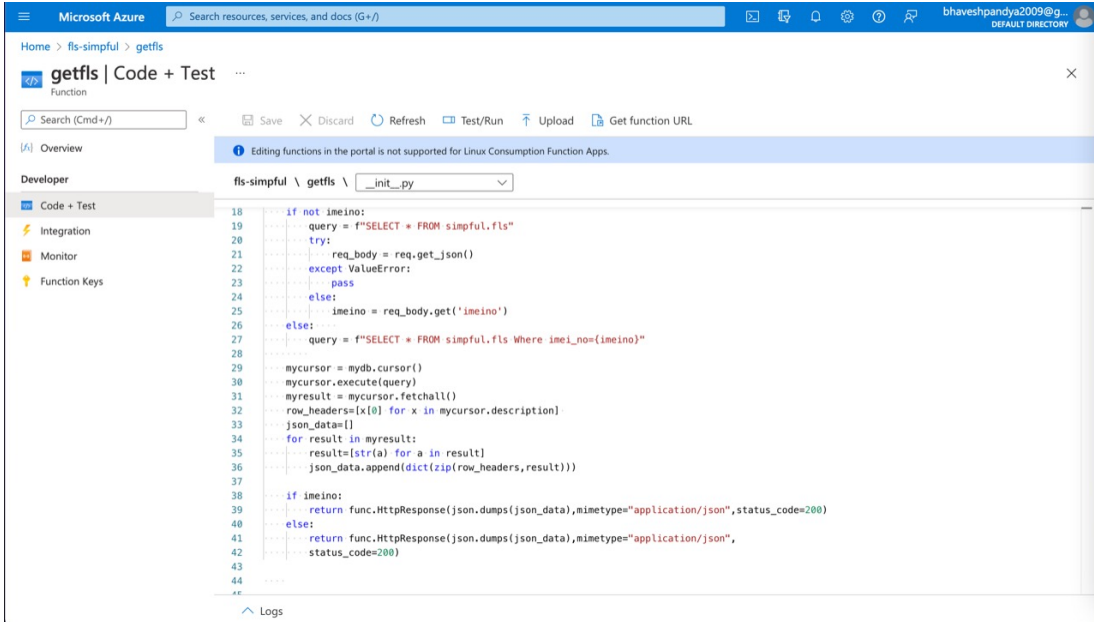


Figure 8: Get FLS.



```

18 if not imeino:
19     query = f"SELECT * FROM simpful.fl's"
20     try:
21         req_body = req.get_json()
22         except ValueError:
23             pass
24     else:
25         imeino = req_body.get('imei_no')
26     else:
27         query = f"SELECT * FROM simpful.fl's Where imei_no={imeino}"
28
29     mycursor = mydb.cursor()
30     mycursor.execute(query)
31     myresult = mycursor.fetchall()
32     row_headers=[x[0] for x in mycursor.description]
33     json_data=[]
34     for result in myresult:
35         result=[str(a) for a in result]
36         json_data.append(dict(zip(row_headers,result)))
37
38     if imeino:
39         return func.HttpResponse(json.dumps(json_data),mimetype="application/json",status_code=200)
40     else:
41         return func.HttpResponse(json.dumps(json_data),mimetype="application/json",
42                                 status_code=200)
43
44
45

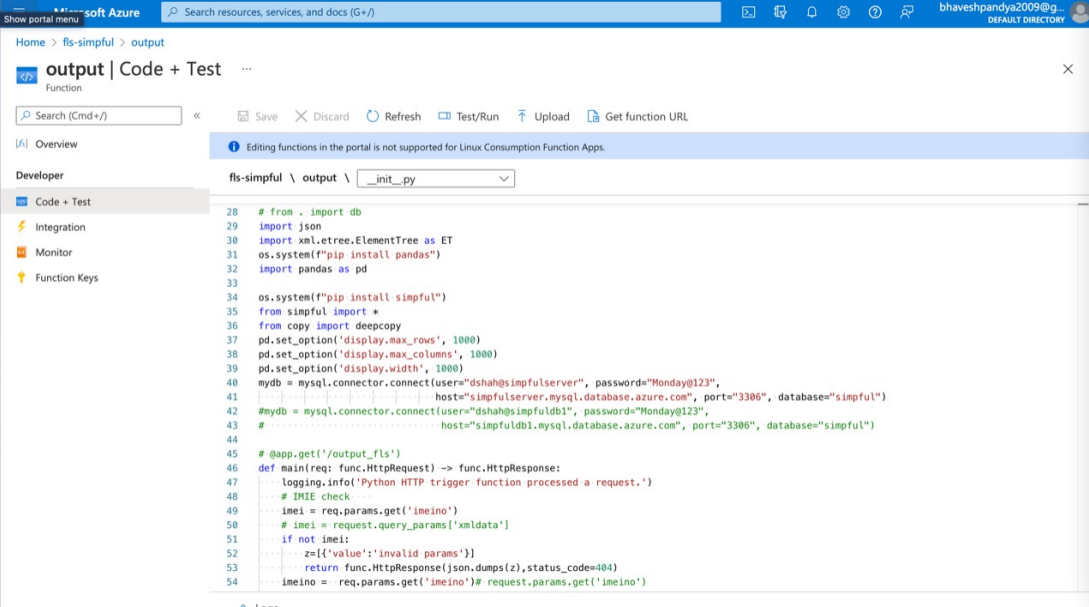
```

Figure 9: Get FLS code.

- inputoutputfls: evaluate the output.
- output: Retrieve the output from the previous functions stored in a database.

A specific sequence must be followed in order to evaluate the function. Postfls must be completed before creating fls. Hit inputoutputfls to evaluate the output, inputfls to set up the parameters, and output API to obtain the output.

The code elaborates the GetFls function, shown in Figure 9. The initial Figure 8 shows a database connectivity query that will instruct the DB to retrieve each and every value from the fls table. If the IMEI number of a device is not present, this provides a list of all of the FLSs that have been created in the system. If it is there, the query for the second snapshot will be altered so that it will only return FLSs that were produced using the IMEI no that was provided in the request if the condition is met. After that, the result is converted into JSON, and that is what is sent back as the output. The Application Programming Interface (API) for posting FLS is created. The code shown in Figure 10 establishes a connection with the database in the usual manner. Then it performs a validation check to determine whether or not the necessary parameters have been passed in the



```
28 # from . import db
29 import json
30 import xml.etree.ElementTree as ET
31 os.system("pip install pandas")
32 import pandas as pd
33
34 os.system("pip install simpful")
35 from simpful import *
36 from copy import deepcopy
37 pd.set_option('display.max_rows', 1000)
38 pd.set_option('display.max_columns', 1000)
39 pd.set_option('display.width', 1000)
40 mydb = mysql.connector.connect(user="dshah@simpfulserver", password="Monday@123",
41                               host="simpfulserver.mysql.database.azure.com", port="3306", database="simpful")
42 #mydb = mysql.connector.connect(user="dshah@simpfuldb1", password="Monday@123",
43                               host="simpfuldb1.mysql.database.azure.com", port="3306", database="simpful")
44
45 # @app.get('/output_fls')
46 def main(req: func.HttpRequest) -> func.HttpResponse:
47     logging.info("Python HTTP trigger function processed a request.")
48     # IMIE check
49     imei = req.params.get('imeino')
50     # imei = request.query_params['xmldata']
51     if not imei:
52         z={'value': 'invalid params'}
53     return func.HttpResponse(json.dumps(z), status_code=404)
54     imeino = req.params.get('imeino')# request.params.get('imeino')
```

Figure 10: Database connect.

request. The IMEI no parameter and the string value of the XML document are required for this API. This includes a query that has been written to store the value that was received in the DB against the IMEI number that was passed.

Appendix B: JFML Library

Details

JFML library [25] adheres to a strong object-oriented methodology and a modular architecture based on the same labelled tree structure that FML uses to describe FLSs, allowing developers to expand FML without modifying the language grammar. This programme is offered as open source under the provisions of the GNU Public License GPLv3 and is hosted on the public hosting platform GitHub, which provides several tools (such as a bug tracker and mailing list) for leveraging the benefits of the open source approach. Additionally, JFML has a web page with extensive documentation and a wide range of examples. Following are the primary properties of JFML: primary classes, FML XSD binding, extensibility, and software compatibility.

The class diagram of the library is already shown in Figure 3.4. The details of JFML classes are explained in this Appendix.

1. Fuzzy System Type Class represents the four FLSs enclosed in the standard scheme: Mamdani, Tsukamoto, TSK, and AnYa. This attribute is also included in other subclasses that have the same aim, and each object of this class possesses a unique name and a network address to define the location of the FLS within a computer network system (this attribute is also included in other subclasses that have the same aim). Additionally, each object of this class consists of one element Knowledge Base and a list of elements Fuzzy System Rule Bases, which represent the KB and the RBs of the system, respectively. It is important to take note that an FLS can accommodate more than one RB, each of which can be of a different

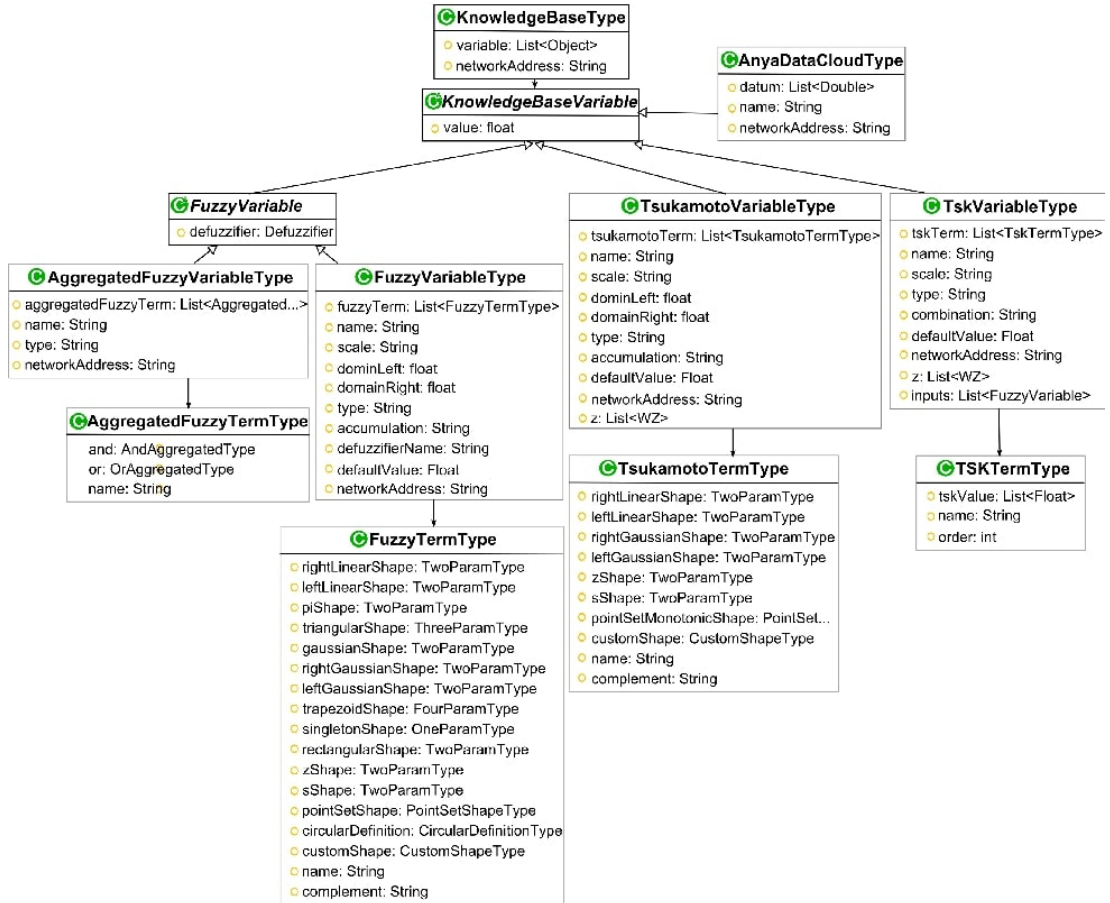


Figure 11: Main Class Diagram for the definitions of KB.

type. By default, the system evaluates RBs in the order they appear in the list, but this can be changed. Accordingly, we can define FLSs in different sub-hierarchies.

2. The Knowledge Base Type Class represents the system’s input and output variables. Each object in this class consists of a list containing one or more members of the class Knowledge Base Variable, which represents a system variable in an abstract manner. Subclasses are responsible for providing detailed information about each variable and the operations that can be performed on it; this base class consists only of the most basic operations that may be performed on any variable (see Figure 11). They are introduced as follows:

3. Fuzzy Variable Type class represents a fuzzy variable that may appear either as an antecedent or a consequent in Mamdani rule bases. It is also possible for it to be a part of the rules in the antecedent of the Tsukamoto and TSK RBs, as well as a part of the rules in the consequent of the AnYa RBs. Each individual object of this class stores information regarding the domain, type (input or output), the scale used to measure the variable, the default value for this variable when no rule has been fired, the accumulation (also known as a combination in the standard), defuzzification methods used when this variable is involved in the consequent of the rules, and a list with the linguistic terms of the variable.
4. Tsukamoto Variable Type class represents an output variable that can be part of the consequent of the rules in Tsukamoto RBs. The objects of this class contain the same information as the objects of the Fuzzy Variable Type class; however, the membership functions of the linguistic terms associated with these objects can only be monotone functions.
5. TSK Variable Type class is used to represent an output variable that may be used as part of the consequent of the rules found in TSK RBs. The consequent of a TSK rule may involve any object of type Tsk Term, which may either represent a constant value (zero-order TSK system) or a linear function of the inputs (one-order TSK system). Each object provides information on its type (input or output), the scale that was used to measure the variable, the default value for this variable when no rule has been fired, the accumulation technique, and a list of other objects that are of the Tsk Term Type.
6. The AnYa Data Cloud Type class is used to describe a data cloud that may be utilised as an antecedent for rules that are contained within AnYa RBs. Each object has its own list of data, which together represents a subset of previous data samples that have common characteristics. It is important to note that each data is specified as only one double value in the standard; nevertheless, it is extensible to consider data objects with more than one value.

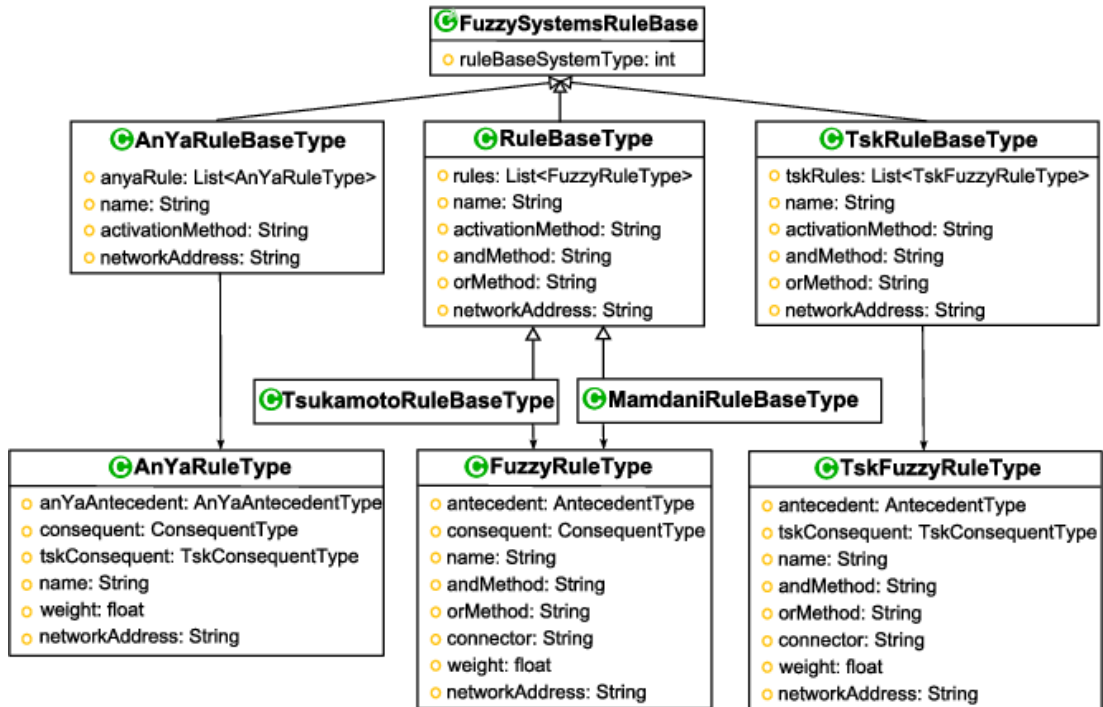


Figure 12: Main Class Diagram for the definitions of RBs.

7. Aggregated Fuzzy Variable class is used to express a new fuzzy variable in which the linguistic terms are specified by the aggregate of the linguistic terms used by existing variables. Each instance of the class object stores a list of Aggregated Fuzzy Term objects. These objects employ the AND (a t-norm) and OR (a t-conorm) operators to generate new terms for the variable based on the combination of two or more terms from other variables. The definition of variables and rules can be handled in an adaptable manner using this class.
8. The Fuzzy Systems Rule Base Class is an abstract representation of an RB of the Fuzzy Logic System. It mainly comprises the basic operations that can be employed for any rule. In contrast, the subclasses are responsible for defining the rules and determining how those rules should be applied (see Figure 12). They are introduced as follows.

Rule Base Type class represents an RB in which the consequent and

antecedent of the rules only have fuzzy variables. In addition to a list of fuzzy rules, each instance of this class also holds details on the activation methods, the AND and OR methods that are applied by default to all rules, and so on. This class is extended by the classes Mamdani Rule Base Type and Tsukamoto Rule Base Type to generate RBs with Mamdani and Tsukamoto rules, respectively. Both of them contain a list of Fuzzy Rule Type objects. Notice that the rules of the Mamdani Rule Base Type object involve Fuzzy Variable Type variables in the antecedent and consequent. The rules of the Tsukamoto Rule Base Type objects involve Fuzzy Variable Type variables in the antecedent and Tsukamoto Variable Type variables in the consequent in order to assure the use of monotone membership functions in the consequent of the rules.

TSK Rule Base Type class represents a TSK RB. The objects of this class contain the same information as the Rule Base Type objects and a list of Tsk Fuzzy Rule Type objects. Each Tsk Fuzzy Rule Type object involves Fuzzy Variable Type variables in the antecedent and Tsk Variable Type variables in the consequent of the rules.

The AnYa Rule Base Type class represents AnYa RBs. Each object of this class contains the activation method and a list of AnYa Rule Type objects. Each AnYa Rule Type object involves AnYa Data Cloud Type variables representing the data cloud used in the antecedent, and Fuzzy Variable Type or Tsk Variable Type variables in the consequent of the rules. These sort of rules can employ the same consequence consequent as Mamdani and TSK rules.

Each antecedent element (Antecedent Type) in the rules has one or more AND/OR-connected clauses. The specific connector and corresponding t-norm/t-conorm can be selected for each rule. In contrast, the consequent elements (Consequent Type and Tsk Consequent Type) have one element THEN, which represents the THEN-part of a rule, and an optional element ELSE, which represents the ELSE-part of a rule. Both parts may also have single or multiple clauses, enabling the modelling of FLSs with numerous outputs. In addition, we may assign a weight to each rule to describe its significance in the inference.

References

- [1] Alfredo J Perez, Miguel A Labrador, and Sean J Barbeau, “G-sense: a scalable architecture for global sensing and monitoring”, *IEEE Network*, vol. 24, no. 4, pp. 57–64, 2010. [4](#)
- [2] Yanjun Jia, “Dietetic and exercise therapy against diabetes mellitus”, in *2009 Second International Conference on Intelligent Networks and Intelligent Systems*. IEEE, 2009, pp. 693–696. [4](#)
- [3] Jie Yin, Qiang Yang, and Jeffrey Junfeng Pan, “Sensor-based abnormal human-activity detection”, *IEEE transactions on knowledge and data engineering*, vol. 20, no. 8, pp. 1082–1090, 2008. [4](#)
- [4] Franz Gosch, *Solutions to master the Demographic Change: Ambient Assisted Living for the Elderly*, Anchor Academic Publishing (aap-verlag), 2014. [4](#)
- [5] Se Jin Park, Murali Subramaniam, Seoung Eun Kim, Seunghee Hong, Joo Hyeong Lee, Chan Min Jo, and Youngseob Seo, “Development of the elderly healthcare monitoring system with iot”, in *Advances in human factors and ergonomics in healthcare*, pp. 309–315. Springer, 2017. [4](#)
- [6] Vincenzo Mighali, Luigi Patrono, Maria Laura Stefanizzi, Joel JPC Rodrigues, and Petar Solic, “A smart remote elderly monitoring system based on iot technologies”, in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*. IEEE, 2017, pp. 43–48. [4](#), [25](#)
- [7] Paola Patricia Ariza-Colpas, Enrico Vicario, Ana Isabel Oviedo-Carrascal, Shariq Butt Aziz, Marlon Alberto Piñeres-Melo, Alejandra Quintero-Linero, and Fulvio Patara, “Human activity recognition data analysis:

REFERENCES

- History, evolutions, and new trends”, *Sensors*, vol. 22, no. 9, pp. 3401, 2022. [8](#), [9](#)
- [8] Plamen Angelov and Ronald Yager, “Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density”, in *2011 IEEE Workshop on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 2011, pp. 62–69. [9](#)
- [9] James J Buckley and Esfandiar Eslami, *An introduction to fuzzy logic and fuzzy sets*, vol. 13, Springer Science & Business Media, 2002. [9](#), [38](#)
- [10] Hassnaa Moustafa, Eve M Schooler, Gang Shen, and Sanjana Kamath, “Remote monitoring and medical devices control in ehealth”, in *2016 IEEE 12th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2016, pp. 1–8. [10](#)
- [11] Lotfi A Zadeh, “Information and control”, *Fuzzy sets*, vol. 8, no. 3, pp. 338–353, 1965. [10](#)
- [12] Monish Kumar Choudhury and Neelanjana Baruah, “A fuzzy logic-based expert system for determination of health risk level of patient”, *International Journal of research in Engineering and Technology*, vol. 4, no. 5, pp. 261–267, 2015. [10](#)
- [13] Zheru Chi, Hong Yan, and Tuan Pham, *Fuzzy algorithms: with applications to image processing and pattern recognition*, vol. 10, World Scientific, 1996. [10](#), [77](#)
- [14] JA Goguen, “La zadeh. fuzzy sets. information and control, vol. 8 (1965), pp. 338–353.-la zadeh. similarity relations and fuzzy orderings. information sciences, vol. 3 (1971), pp. 177–200.”, *The Journal of Symbolic Logic*, vol. 38, no. 4, pp. 656–657, 1973. [10](#)
- [15] Giovanni Acampora, Vincenzo Loia, Michele Nappi, and Stefano Ricciardi, “Ambient intelligence framework for context aware adaptive applications”, in *Seventh International Workshop on Computer Architecture for Machine Perception (CAMP’05)*. IEEE, 2005, pp. 327–332. [11](#)

REFERENCES

- [16] Valentina E Balas, Jer Lang Hong, Jason Gu, and Tsung-Chih Lin, “Special issue on fuzzy theoretical model analysis for signal processing”, *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 4, pp. 4407–4411, 2019. [11](#)
- [17] Mamta Mittal, Valentina E Balas, Lalit Mohan Goyal, and Raghvendra Kumar, *Big data processing using spark in cloud*, Springer, 2019. [12](#)
- [18] Dariusz Mrozek, Anna Koczur, and Bożena Małysiak-Mrozek, “Fall detection in older adults with mobile iot devices and machine learning in the cloud and on the edge”, *Information Sciences*, vol. 537, pp. 132–147, 2020. [13](#)
- [19] Atis Elsts, Xenofon Fafoutis, Przemyslaw Woznowski, Emma Tonkin, George Oikonomou, Robert Piechocki, and Ian Craddock, “Enabling healthcare in smart homes: the sphere iot network infrastructure”, *IEEE Communications Magazine*, vol. 56, no. 12, pp. 164–170, 2018. [13](#)
- [20] Manuel Jesús Parra Royón, José Manuel Benítez Sánchez, et al., “Fuzzy systems-as-a-service in cloud computing”, 2019. [13](#), [27](#), [34](#)
- [21] Tabish Mufti, Shahab Saquib Sohail, Bulbul Gupta, and Parul Agarwal, “Sustainable approach for cloud-based framework using iot in healthcare”, in *Smart Technologies for Energy and Environmental Sustainability*, pp. 231–244. Springer, 2022. [13](#)
- [22] Abtoy Anouar, Touhafi Abdellah, Tahiri Abderahim, et al., “A novel reference model for ambient assisted living systems’ architectures”, *Applied Computing and Informatics*, 2020. [13](#)
- [23] Abtoy Anouar, Abdellah Touhafi, Abderahim Tahiri, et al., “Towards an soa architectural model for aal-paas design and implimentation challenges”, *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 7, 2017. [13](#)
- [24] Giovanni Acampora, Bruno Di Stefano, and Autilia Vitiello, “Ieee 1855™: The first iee standard sponsored by iee computational intelligence society [society briefs]”, *IEEE Computational Intelligence Magazine*, vol. 11, no. 4, pp. 4–6, 2016. [15](#), [30](#), [32](#), [35](#), [44](#), [46](#)

REFERENCES

- [25] José M Soto-Hidalgo, Jose M Alonso, Giovanni Acampora, and Jesús Alcalá-Fdez, “Jfml: a java library to design fuzzy logic systems according to the ieee std 1855-2016”, *IEEE Access*, vol. 6, pp. 54952–54964, 2018. [15](#), [32](#), [35](#), [134](#)
- [26] Marco S Nobile, Uzay Kaymak, Caro EM Fuchs, Simone Spolaor, Paolo Cazzaniga, and Daniela Besozzi, “Simpful python library”, 2020. [15](#)
- [27] Neha Nilesh Jadhav Sarnaik, “Human activity recognition using cnn”, *International journal of scientific and research publications*, vol. 10, pp. 9804, 2020. [22](#)
- [28] Abdallah Naser, Ahmad Lotfi, and Joni Zhong, “Towards human distance estimation using a thermal sensor array”, *Neural Computing and Applications*, pp. 1–11, 2021. [22](#)
- [29] Abdallah Naser, Ahmad Lotfi, and Junpei Zhong, “A novel privacy-preserving approach for physical distancing measurement using thermal sensor array”, in *The 14th PErvasive Technologies Related to Assistive Environments Conference*, 2021, pp. 81–85. [22](#)
- [30] Raffaele Gravina, Congcong Ma, Pace Pasquale, Gianluca Aloï, Wilma Russo, Wenfeng Li, and Giancarlo Fortino, “Cloud-based activity-aaservice cyberphysical framework for human activity monitoring in mobility”, *Future Generation Computer Systems*, vol. 75, 09 2016. [23](#)
- [31] Naveed Islam, Yasir Faheem, Ikram Ud Din, Muhammad Talha, Mohsen Guizani, and Mudassir Khalil, “A blockchain-based fog computing framework for activity recognition as an application to e-healthcare services”, *Future Generation Computer Systems*, vol. 100, pp. 569–578, 2019. [23](#)
- [32] Marcos Lupión, Javier Medina-Quero, Juan F Sanjuan, and Pilar M Ortigosa, “Dolars, a distributed on-line activity recognition system by means of heterogeneous sensors in real-life deployments—a case study in the smart lab of the university of almería”, *Sensors*, vol. 21, no. 2, pp. 405, 2021. [24](#)

REFERENCES

- [33] Mian Mujtaba Ali, Shyqyri Haxha, Munna M Alam, Chike Nwibor, and Mohamed Sakel, “Design of internet of things (iot) and android based low cost health monitoring embedded system wearable sensor for measuring spo₂, heart rate and body temperature simultaneously”, *Wireless Personal Communications*, vol. 111, no. 4, pp. 2449–2463, 2020. [25](#)
- [34] Mohammed Al-Khafajiy, Thar Baker, Carl Chalmers, Muhammad Asim, Hoshang Kolivand, Muhammad Fahim, and Atif Waraich, “Remote health monitoring of elderly through wearable sensors”, *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 24681–24706, 2019. [25](#)
- [35] Ananda Mohon Ghosh, Debashish Halder, and SK Alamgir Hossain, “Remote health monitoring system through iot”, in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*. IEEE, 2016, pp. 921–926. [25](#)
- [36] Serhan Dağtaş, Georgiy Pekhteryev, Z Şahinoğlu, Hasan Cam, and Narasimha Challa, “Real-time and secure wireless health monitoring”, *International Journal of Telemedicine and Applications*, vol. 2008, 2008. [25](#)
- [37] Xun Yi, Athman Bouguettaya, Dimitrios Georgakopoulos, Andy Song, and Jan Willemsen, “Privacy protection for wireless medical sensor data”, *IEEE transactions on dependable and secure computing*, vol. 13, no. 3, pp. 369–380, 2015. [25](#)
- [38] M Surya Deekshith Gupta, Vamsikrishna Patchava, and Virginia Menezes, “Healthcare based on iot using raspberry pi”, in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE, 2015, pp. 796–799. [25](#)
- [39] Junaid Mohammed, Chung-Horng Lung, Adrian Ocneanu, Abhinav Thakral, Colin Jones, and Andy Adler, “Internet of things: Remote patient monitoring using web services and cloud computing”, in *2014 IEEE international conference on internet of things (IThings), and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom)*. IEEE, 2014, pp. 256–263. [25](#)

REFERENCES

- [40] Ali I Siam, Nirmeen A El-Bahnasawy, Ghada M El Banby, Atef Abou Elazm, and Fathi E Abd El-Samie, “Efficient video-based breathing pattern and respiration rate monitoring for remote health monitoring”, *JOSA A*, vol. 37, no. 11, pp. C118–C124, 2020. [25](#)
- [41] Shourjya Sanyal and Koushik Kumar Nundy, “Algorithms for monitoring heart rate and respiratory rate from the video of a user’s face”, *IEEE Journal of translational engineering in health and medicine*, vol. 6, pp. 1–11, 2018. [25](#)
- [42] Mohammad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany, “Cuffless blood pressure estimation algorithms for continuous health-care monitoring”, *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 4, pp. 859–869, 2016. [25](#)
- [43] Josep Sola, Martin Proença, Damien Ferrario, Jacques-André Porchet, Abdessamad Falhi, Olivier Grossenbacher, Yves Allemann, Stefano F Rimoldi, and Claudio Sartori, “Noninvasive and nonocclusive blood pressure estimation via a chest sensor”, *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 12, pp. 3505–3513, 2013. [25](#)
- [44] Damien Brulin, Yannick Benezeth, and Estelle Courtial, “Posture recognition based on fuzzy logic for home monitoring of the elderly”, *IEEE transactions on information technology in biomedicine*, vol. 16, no. 5, pp. 974–982, 2012. [24](#)
- [45] Minoorashidpour, F Abdali-Mohammadi, and Abdolhossein Fathi, “Fall detection using adaptive neuro-fuzzy inference system”, *Int. J. Multimed. Ubiquitous Eng*, vol. 11, pp. 91–106, 2016. [26](#)
- [46] Bogdan Kwolek and Michal Kepski, “Fuzzy inference-based fall detection using kinect and body-worn accelerometer”, *Applied Soft Computing*, vol. 40, pp. 305–318, 2016. [26](#)
- [47] M Jayalakshmi, Lalit Garg, K Maharajan, K Jayakumar, Kathiravan Srinivasan, Ali Kashif Bashir, and K Ramesh, “Fuzzy logic-based health monitoring system for covid’19 patients”, *Cmc-Computers Materials & Continua*, pp. 2430–2446, 2021. [26](#)

REFERENCES

- [48] Barbara Pekala, Teresa Mroczek, Dorota Gil, and Michal Kepski, “Application of fuzzy and rough logic to posture recognition in fall detection system”, *Sensors*, vol. 22, no. 4, pp. 1602, 2022. [26](#), [62](#)
- [49] A Thilagavathy, S Meenakshi, V Vijayabhaskar, MD Babu, S Kumari, and MA Gunavathie, “An efficient health monitoring method using fuzzy inference system via cloud”, *Indian Journal of Science and Technology*, vol. 14, no. 25, pp. 2145–2151, 2021. [28](#)
- [50] Daohua Pan, Hongwei Liu, Dongming Qu, and Zhan Zhang, “Human falling detection algorithm based on multisensor data fusion with svm”, *Mobile Information Systems*, vol. 2020, 2020. [28](#)
- [51] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang, “Xai—explainable artificial intelligence”, *Science robotics*, vol. 4, no. 37, pp. eaay7120, 2019. [30](#)
- [52] Giovanni Acampora, *Transparent Fuzzy Agents in Ambient Intelligence Environments*, PhD thesis, Ph. D. Thesis, University of Salerno, 2007. [30](#)
- [53] Chang-Shing Lee, Mei-Hui Wang, Li-Chuang Chen, Yusuke Nojima, Tzong-Xiang Huang, Jinseok Woo, Naoyuki Kubota, Eri Sato-Shimokawara, and Toru Yamaguchi, “A gfml-based robot agent for human and machine cooperative learning on game of go”, in *2019 IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 793–799. [31](#)
- [54] Giovanni Acampora, “Fuzzy markup language: A xml based language for enabling full interoperability in fuzzy systems design”, in *On the power of fuzzy markup language*, pp. 17–31. Springer, 2013. [31](#)
- [55] Michael Tiegelkamp and Karl-Heinz John, *IEC 61131-3: Programming industrial automation systems*, vol. 166, Springer, 2010. [31](#)
- [56] D López, FJ Moreno, A Barriga, and S Sánchez-Solano, “Xfl: a language for the definition of fuzzy systems”, in *Proceedings of 6th International Fuzzy Systems Conference*. IEEE, 1997, vol. 3, pp. 1585–1591. [31](#)

- [57] Francisco José Moreno Velo, María Iluminada Baturone Castillo, Santiago Sánchez Solano, and Ángel Barriga Barros, “Xfuzzy 3.0: a development environment for fuzzy systems”, 2001. [31](#)
- [58] Francisco Jose Moreno-Velo, Angel Barriga, Santiago Sánchez-Solano, and Iluminada Baturone, “Xfsm: An xml-based modeling language for fuzzy systems”, in *2012 IEEE International Conference on Fuzzy Systems*. IEEE, 2012, pp. 1–8. [32](#)
- [59] Bruno N Di Stefano, “On the need of a standard language for designing fuzzy systems”, *On the Power of Fuzzy Markup Language*, pp. 3–15, 2013. [32](#)
- [60] Francisco Jesús Arcos, José Manuel Soto-Hidalgo, Autilia Vitiello, Giovanni Acampora, and Jesús Alcalá-Fdez, “Interoperability for embedded systems in jfml software: An arduino-based implementation”, in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–8. [32](#)
- [61] Christian Wagner, “Juzzy-a java based toolkit for type-2 fuzzy logic”, in *2013 IEEE Symposium on Advances in Type-2 Fuzzy Logic Systems (T2FUZZ)*. IEEE, 2013, pp. 45–52. [35](#)
- [62] Jesús Alcalá-Fdez, Jose M Alonso, Ciro Castiello, Corrado Mencar, and José M Soto-Hidalgo, “Py4jfml: A python wrapper for using the ieee std 1855-2016 through jfml”, in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6. [35](#)
- [63] Simone Spolaor, Caro Fuchs, Paolo Cazzaniga, Uzay Kaymak, Daniela Besozzi, and Marco S Nobile, “Simpful: a user-friendly python library for fuzzy logic”, *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 1687–1698, 2020. [35](#), [102](#)
- [64] Amir Pourabdollah, Christian Wagner, Giovanni Acampora, and Ahmad Lotfi, “Fuzzy logic as-a-service for ambient intelligence environments”, in *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2018, pp. 1–7. [34](#), [39](#)

REFERENCES

- [65] Giovanni Acampora and Autilia Vitiello, “Extending iee std 1855 for designing arduino™-based fuzzy systems”, in *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2017, pp. 1–6. [34](#), [47](#)
- [66] Giovanni Acampora, Kofi Appiah, Andrew Hunter, and Autilia Vitiello, “Interoperable services based on activity monitoring in ambient assisted living environments”, in *2014 IEEE Symposium on Intelligent Agents (IA)*. IEEE, 2014, pp. 81–88. [34](#)
- [67] Giovanni Acampora, Jesus Alcala-Fdez, Roberta Siciliano, José M Soto-Hidalgo, and Autilia Vitiello, “Visualjfml: A visual environment for designing fuzzy systems according to iee std 1855-2016”, in *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 2019, pp. 1–6. [34](#)
- [68] José Manuel Soto Hidalgo, Autilia Vitiello, Jose M Alonso, Giovanni Acampora, Jesús Alcalá Fernández, et al., “Design of fuzzy controllers for embedded systems with jfml”, 2019. [34](#), [47](#)
- [69] JS Roger Jang, *MATLAB: Fuzzy logic toolbox user’s guide: Version 1*, Math Works, 1997. [39](#)
- [70] Christian Wagner, Simon Miller, and Jonathan M Garibaldi, “A fuzzy toolbox for the r programming language”, in *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*. IEEE, 2011, pp. 1185–1192. [39](#)
- [71] Giovanni Acampora and Autilia Vitiello, “Interoperable neuro-fuzzy services for emotion-aware ambient intelligence”, *Neurocomputing*, vol. 122, pp. 3–12, 2013, Advances in cognitive and ubiquitous computing. [40](#)
- [72] G. Acampora and V. Loia, “Fuzzy control interoperability and scalability for adaptive domotic framework”, *IEEE Transactions on Industrial Informatics*, vol. 1, no. 2, pp. 97–111, 2005. [41](#)
- [73] Giovanni Acampora, Matteo Gaeta, Vincenzo Loia, and Athanasios V. Vasilakos, “Interoperable and adaptive fuzzy services for ambient

-
- intelligence applications”, *ACM Trans. Auton. Adapt. Syst.*, vol. 5, no. 2, May 2010. [41](#)
- [74] Yong Ho Kim, Sang Chul Ahn, and Wook Hyun Kwon, “Computational complexity of general fuzzy logic control and its simplification for a loop controller”, *Fuzzy Sets and Systems*, vol. 111, no. 2, pp. 215–224, 2000. [41](#)
- [75] M MENDEL JERRY, *Uncertain rule-based fuzzy systems: Introduction and New Directions*, Springer, 2019. [41](#)
- [76] Javier Cubo, Adrián Nieto, and Ernesto Pimentel, “A cloud-based internet of things platform for ambient assisted living”, *Sensors*, vol. 14, no. 8, pp. 14070–14105, 2014. [42](#)
- [77] Sonal Chandrakant Chavan and Arun Chavan, “Smart wearable system for fall detection in elderly people using internet of things platform”, in *2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 2017, pp. 1135–1140. [42](#)
- [78] Ashish Patel and Jigarkumar Shah, “Sensor-based activity recognition in the context of ambient assisted living systems: A review”, *Journal of Ambient Intelligence and Smart Environments*, vol. 11, no. 4, pp. 301–322, 2019. [43](#)
- [79] Thato E Foko, Nomusa Dlodlo, and Litsietsi Montsi, “An integrated smart system for ambient-assisted living”, in *Internet of Things, Smart Spaces, and Next Generation Networking*, pp. 128–138. Springer, 2013. [43](#)
- [80] Jaime Lloret, Alejandro Canovas, Sandra Sendra, and Lorena Parra, “A smart communication architecture for ambient assisted living”, *IEEE Communications Magazine*, vol. 53, no. 1, pp. 26–33, 2015. [46](#)
- [81] OO Ogunduyile, Keneilwe Zuva, OA Randle, and Transos Zuva, “Ubiquitous healthcare monitoring system using integrated triaxial accelerometer, spo2 and location sensors”, *arXiv preprint arXiv:1309.1542*, 2013. [62](#)
- [82] Jian He, Chen Hu, and Xiaoyi Wang, “A smart device enabled system for autonomous fall detection and alert”, *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, pp. 2308183, 2016. [62](#)

-
- [83] Quoc T Huynh, Uyen D Nguyen, Lucia B Irazabal, Nazanin Ghassemian, and Binh Q Tran, “Optimization of an accelerometer and gyroscope-based fall detection algorithm”, *Journal of Sensors*, vol. 2015, 2015. [62](#)
- [84] Jane Fleming and Carol Brayne, “Inability to get up after falling, subsequent time on floor, and summoning help: prospective cohort study in people over 90”, *Bmj*, vol. 337, 2008. [62](#)
- [85] Dorit Kunkel, Ruth M Pickering, and Ann M Ashburn, “Comparison of retrospective interviews and prospective diaries to facilitate fall reports among people with stroke”, *Age and ageing*, vol. 40, no. 2, pp. 277–280, 2011. [62](#)
- [86] Jian He, Shuang Bai, and Xiaoyi Wang, “An unobtrusive fall detection and alerting system based on kalman filter and bayes network classifier”, *Sensors*, vol. 17, no. 6, pp. 1393, 2017. [62](#)
- [87] Raul Igual, Carlos Medrano, and Inmaculada Plaza, “Challenges, issues and trends in fall detection systems”, *Biomedical engineering online*, vol. 12, no. 1, pp. 1–24, 2013. [62](#)
- [88] Bruno Aguiar, Tiago Rocha, Joana Silva, and Ines Sousa, “Accelerometer-based fall detection for smartphones”, in *2014 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*. IEEE, 2014, pp. 1–6. [62](#)
- [89] S Abbate, M Avvenuti, F Bonatesta, G Cola, P Corsini, and A Vecchio, “A smartphone-based fall detection system. perv mob comput j 2012 (8): 883–899”, 2012. [62](#)
- [90] Gaetano Anania, Alessandro Tognetti, Nicola Carbonaro, Mario Tesconi, Fabrizio Cutolo, Giuseppe Zupone, and Danilo De Rossi, “Development of a novel algorithm for human fall detection using wearable sensors”, in *SENSORS, 2008 IEEE*. IEEE, 2008, pp. 1336–1339. [62](#)
- [91] Majd Saleh and Régine Le Bouquin Jeannès, “Elderly fall detection using wearable sensors: A low cost highly accurate algorithm”, *IEEE Sensors Journal*, vol. 19, no. 8, pp. 3156–3164, 2019. [62](#)

REFERENCES

- [92] Hongtao Zhang, Muhandand Alrifaa, Keming Zhou, and Huosheng Hu, “A novel fuzzy logic algorithm for accurate fall detection of smart wristband”, *Transactions of the Institute of Measurement and Control*, vol. 42, no. 4, pp. 786–794, 2020. [67](#), [68](#)
- [93] Alan K Bourke and Gerald M Lyons, “A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor”, *Medical engineering & physics*, vol. 30, no. 1, pp. 84–90, 2008. [69](#)
- [94] Tanmoy Sarkar Pias, David Eisenberg, and Muhammad Aminul Islam, “Vehicle recognition via sensor data from smart devices”, in *2019 IEEE Eurasia Conference on IOT, Communication and Engineering (ECICE)*. IEEE, 2019, pp. 96–99. [74](#)
- [95] Chang-Shing Lee, Mei-Hui Wang, Li-Wei Ko, Bo-Yu Tsai, Yi-Lin Tsai, Sheng-Chi Yang, Lu-An Lin, Yi-Hsiu Lee, Hirofumi Ohashi, Naoyuki Kubota, et al., “Pfm-based semantic bci agent for game of go learning and prediction”, *arXiv preprint arXiv:1901.02999*, 2019. [77](#)
- [96] Chang-Shing Lee, Mei-Hui Wang, Chi-Shiang Wang, Olivier Teytaud, Jialin Liu, Su-Wei Lin, and Pi-Hsia Hung, “Pso-based fuzzy markup language for student learning performance evaluation and educational application”, *IEEE Transactions on Fuzzy Systems*, vol. 26, no. 5, pp. 2618–2633, 2018. [77](#)
- [97] L-X Wang and Jerry M Mendel, “Generating fuzzy rules by learning from examples”, *IEEE Transactions on systems, man, and cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992. [77](#)
- [98] Abdulatif Alabdulatif, Ibrahim Khalil, Abdur Rahim Mohammad Forkan, and Mohammed Atiquzzaman, “Real-time secure health surveillance for smarter health communities”, *IEEE Communications Magazine*, vol. 57, no. 1, pp. 122–129, 2018. [88](#)
- [99] Yu Fu and Jian Liu, “System design for wearable blood oxygen saturation and pulse measurement device”, *Procedia manufacturing*, vol. 3, pp. 1187–1194, 2015. [88](#), [89](#)

REFERENCES

- [100] Prisma Megantoro et al., “Detection of hypoxic symptoms system based on oxygen saturation and heart rate using arduino based fuzzy method”, in *2020 2nd International Conference on Industrial Electrical and Electronics (ICIEE)*. IEEE, 2020, pp. 107–111. [88](#)
- [101] Sami Alshorman, Fadi T Jaber, and Faycal Bensaali, “A wireless oxygen saturation and heart rate monitoring and alarming system based on the qatar early warning scoring system”, in *2015 International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2015, pp. 787–790. [88](#)
- [102] Muhammad F Rian, Michrandi N Surya, and Astuti N Ratna, “Health monitoring application using fuzzy logic based on android”, in *Journal of Physics: Conference Series*. IOP Publishing, 2019, vol. 1192, p. 012052. [88](#)
- [103] World Health Organization, *Oral health surveys: basic methods*, World Health Organization, 2013. [88](#)
- [104] Mike Patton, “Us health care costs rise faster than inflation”, *Retrieved from Forbes: <https://scholar.google.com/scholar>*, 2015. [88](#)
- [105] Frans M Coetzee and Ziad Elghazzawi, “Noise-resistant pulse oximetry using a synthetic reference signal”, *IEEE Transactions on Biomedical Engineering*, vol. 47, no. 8, pp. 1018–1026, 2000. [89](#)
- [106] Marek Krehel, Martin Wolf, Luciano F Boesel, René M Rossi, Gian-Luca Bona, and Lukas J Scherer, “Development of a luminous textile for reflective pulse oximetry measurements”, *Biomedical optics express*, vol. 5, no. 8, pp. 2537–2547, 2014. [89](#)
- [107] Hariharan Subramanian, Bennett L Ibey, Weijian Xu, Mark A Wilson, M Nance Ericson, and Gerard L Coté, “Real-time separation of perfusion and oxygenation signals for an implantable sensor using adaptive filtering”, *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 12, pp. 2016–2023, 2005. [89](#)
- [108] Felix Adochiei, Cristian Rotariu, Razvan Ciobotariu, and Hariton Costin, “A wireless low-power pulse oximetry system for patient telemonitoring”,

REFERENCES

in *2011 7th International Symposium on Advanced Topics in Electrical Engineering (ATEE)*. IEEE, 2011, pp. 1–4. [89](#)