Towards Designing AI-aided Lightweight Solutions for Key Challenges in Sensing, Communication and Computing Layers of IoT: Smart Health Use-cases

by

Sadman Sakib
B.Sc. Ahsanullah University of Science and Technology, 2017

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE FACULTY OF GRADUATE STUDIES
OF LAKEHEAD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE (SPECIALIZATION IN ARTIFICIAL INTELLIGENCE)

Supervisory Committee

Dr. Zubair Fadlullah
Supervisor
(*Research Chair, Thunder Bay Regional Health Research Institute*
Associate Professor, Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada.)

Dr. Quazi Abidur Rahman
Internal Examiner
(*Assistant Professor, Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada.*)

Dr. Sameh Sorour
External Examiner
(*Assistant Professor, School of Computing, Queen's University, Kingston, Ontario, Canada*)

ABSTRACT

The advent of the 5G and Beyond 5G (B5G) communication system, along with the proliferation of the Internet of Things (IoT) and Artificial Intelligence (AI), have started to evolve the vision of the smart world into a reality. Similarly, the Internet of Medical Things (IoMT) and AI have introduced numerous new dimensions towards attaining intelligent and connected mobile health (mHealth). The demands of continuous remote health monitoring with automated, lightweight, and secure systems have massively escalated. The AI-driven IoT/IoMT can play an essential role in sufficing this demand, but there are several challenges in attaining it. We can look into these emerging hurdles in IoT from three directions: the sensing layer, the communication layer, and the computing layer. Existing centralized remote cloud-based AI analytics is not adequate for solving these challenges, and we need to emphasize bringing the analytics into the ultra-edge IoT. Furthermore, from the communication perspective, the conventional techniques are not viable for the practical delivery of health data in dynamic network conditions in 5G and B5G network systems. Therefore, we need to go beyond the traditional realm and press the need to incorporate lightweight AI architecture to solve various challenges in the three mentioned IoT planes, enhancing the healthcare system in decision making and health data transmission.

In this thesis, we present different AI-enabled techniques to provide practical and lightweight solutions to some selected challenges in the three IoT planes. Therefore, by exploring one important use-case from a diverse pool of available use-cases in each of the IoT planes, we summarize the contribution of the thesis as the following:

- Focusing on the sensing plane, chapter 3 employs Reservoir Computing (RC) for noise-removal from the magnetocardiography (MCG) signal for continuous remote monitoring of cardiovascular activities.

- In chapter 4, to tackle the challenging task of dynamic channel selection in the communication plane, a deep learning-based predictive channel selection method is leveraged. The proposed AI-aided method will unravel the potential challenges associated with the dynamic channel conditions in the B5G networks while transmitting massive health data (big data) from countless IoT devices.

- Finally, to facilitate the computing plane of IoT, we investigate the use-case of arrhythmia classification by analyzing electrocardiography (ECG). Hence, in chapter 5, we explored how to design a lightweight AI model embedded into the ultra-edge IoT nodes for arrhythmia classification. Then in chapter 6 we press the concept of designing federated learning architecture based on asynchronously model updating online and decentralized learning technique which uses the Ultra-Edge Nodes (UEN) as the local users and utilizes the lightweight AI technique to detect irregular heartbeats.

We have employed publicly available data sources to gain insights and evaluated the proposed AI solutions by carefully identifying different performance indicators. Thus, we envision that, in the forthcoming future of B5G networks, we can achieve a smart and connected healthcare system in decision-making and efficient health data transmission by blending lightweight AI computing with ultra-edge IoT sensors.

ACKNOWLEDGEMENTS

In the name of the Almighty Allah, the most gracious and the most merciful.

All praises belong to Allah and his blessing for the completion of this thesis. I thank God for all the patience, opportunities, and strength that have been showered on me to finish the thesis.

Apart from my efforts, the success of this thesis depends largely on the encouragement and guidelines of many others. I would take this opportunity to express my gratefulness to the people who have been instrumental in completing this thesis.

I am very grateful to the following funding sources for financial supporting my research:

- Lakehead University Faculty of Graduate Studies;
- Lakehead University Faculty of Science and Environmental Studies;
- Dr. Zubair Fadlullah (Faculty Research Award)
- Thunder Bay Regional Health Research Institute (TBRHRI)
- Mitacs Accelerate

I want to express my deepest gratitude and gratefulness to my supervisor, Dr. Zubair Fadlullah, for advising me, mentoring me, encouraging me, and guiding me in my research and academic life. Dr. Falullah's patience, positive attitude, and unique research directions always gave me inspiration and the freedom to perform my tasks with ease. I am always grateful to him for his invaluable guidance in the last couple of years in both my academic and personal life.

I would like to thank our research collaborators and co-authors for their participation and contributions to the research works. My absolute gratitude and appreciation to Dr. Mostafa Fouda (Assistant Professor, Idaho State University) for helping me in different stages of my research with his valuable insights and opinions. Also, I am thankful to all of my lab members from the ACCESS Lab for the stimulating research discussions. Thanks to the examiners (Dr. Quazi Abidur Rahman and Dr. Sameh Sorour) for their valuable comments and constructive suggestion. My utmost admiration and thanks to Dr. Salimur Choudhury for all the academic as well as personal advice during the last few years.

Finally, I am sincerely grateful and forever in debt to all my family members: my parents, mother-in-law, wife, and brother, for their immense patience, constant support, and continuous prayer in my academic/research journey and life in general. I am also grateful to my teachers, colleagues, friends, and everyone who helped me in my life and motivated me to pursue my higher study.

PUBLICATIONS

Parts of this thesis have been submitted for peer-review, published or accepted for publication:

- **Noise-Removal from Spectrally-Similar Signals Using Reservoir Computing for MCG Monitoring** has been accepted in the IEEE International Conference on Communications (ICC) 2021. (part of Chapter 3)
- **An Efficient and Light-weight Predictive Channel Assignment Scheme for Multi-Band B5G Enabled Massive IoT: A Deep Learning Approach** is published in IEEE Internet of Things Journal, doi: 10.1109/JIOT.2020.3032516. (part of Chapter 4)
- **A Deep Learning Method for Predictive Channel Assignment in Beyond 5G Networks** is published in IEEE Network, vol. 35, no. 1, pp. 266-272, January/February 2021, doi: 10.1109/MNET.011.2000301. (part of Chapter 4)
- **A Proof-of-Concept of Ultra-Edge Smart IoT Sensor: A Continuous and Lightweight Arrhythmia Monitoring Approach** is published in IEEE Access, vol. 9, pp. 26093-26106, 2021, doi: 10.1109/ACCESS.2021.3056509. (part of Chapter 5)
- **Migrating Intelligence from Cloud to Ultra-Edge Smart IoT Sensor Based on Deep Learning: An Arrhythmia Monitoring Use-Case** is published in 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 2020, doi: 10.1109/IWCMC48107.2020.9148134. (part of Chapter 5)
- **A Rigorous Analysis of Biomedical Edge Computing: An Arrhythmia Classification Use-Case Leveraging Deep Learning** is published in 2020 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS), BALI, Indonesia, 2020, doi: 10.1109/IoTaIS50849.2021.9359721. (part of Chapter 5)

Apart from the manuscripts mentioned above, during my MSc, I also authored a few other papers outside the scope of the thesis. Following are the list of such published/accepted papers:

- **On COVID-19 Prediction Using Asynchronous Federated Learning-Based Agile Radiograph Screening Booths** has been accepted in the IEEE International Conference on Communications (ICC) 2021.
- **DL-CRC: Deep Learning-Based Chest Radiograph Classification for COVID-19 Detection: A Novel Approach** is published in IEEE Access, vol. 8, pp. 171575-171589, 2020, doi: 10.1109/ACCESS.2020.3025010.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The recent universal advancement in the Internet of Things (IoT), Internet of Medical Things (IoMT), fifth-generation (5G), and beyond 5G (B5G) wireless networks are envisioned to pave the way towards Artificial Intelligence (AI)-based smart, secure, and connected healthcare system in the upcoming future. The Healthcare system is increasingly utilizing information technologies for delivering ubiquitous services aiming at speeding up health diagnostics, and treatment [3]. The in-depth integration of the IoT/IoMT, 5G, AI, big data, cloud computing, and other advanced technologies is already solving many healthcare system challenges. Such systems are already allowing a substantial reduction of cost and enhancing patient care by providing intelligent services for health monitoring and medical automation in diverse contexts and environments (i.e., hospitals, home, office). The AI-empowered automatic diagnostic techniques can also make healthcare and remote health monitoring more effective and faster [4]. The emergence of a wide range of smart sensors, IoT devices, and remote toolkits has observed remarkable advances in remote patient monitoring and telehealth solutions. Using AI and virtual reality, medical professionals can step into hospitals and medical facilities worldwide to provide prompt care to patients in hazardous or remote regions. Wearable devices to monitor patient vital signs such as heart rate, brain signals, oxygen saturation, and so forth to can minimize healthcare practitioners' risk while ensuring remote patient care.

Medical 4.0 is now emerging as the fourth medical revolution where new modern technologies are integrated into the healthcare system, and proper decision-making processes are implemented for extensive customized healthcare services [5]. However, lately, the spread of novel coronavirus (COVID-19) has caused significant strain on medical centers' resources and escalated the demand for remote automated patient monitoring. The pandemic has radically and suddenly altered how medical practitioners provide care to subjects. Healthcare providers are now responding to the urgency through the rapid adoption of digital tools and technologies such as telemedicine and virtual care, which refer to delivering healthcare

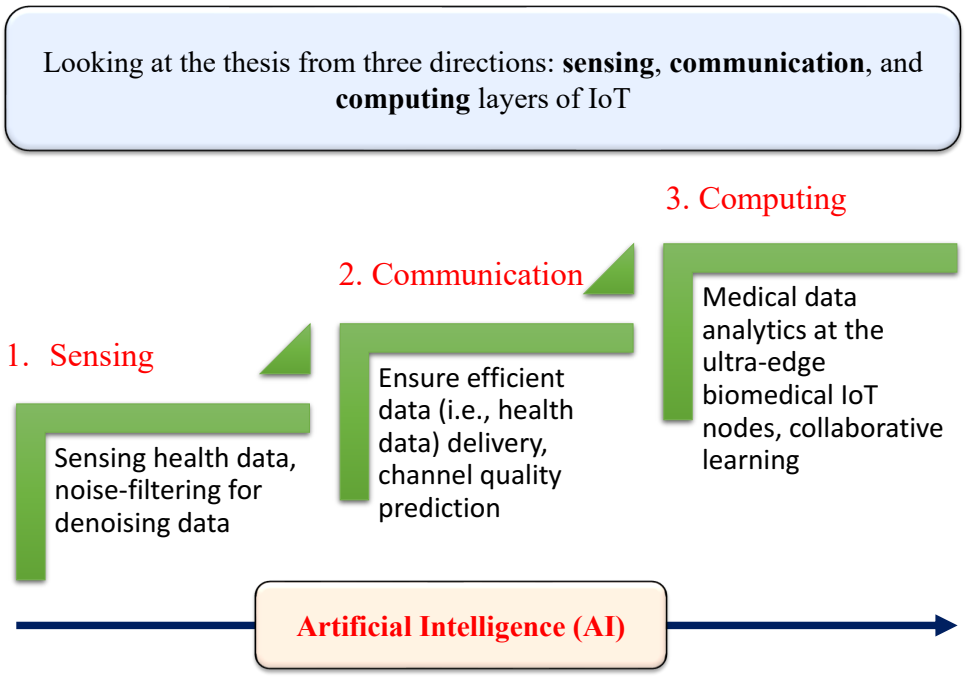


Figure 1.1: Focusing on the IoT from three directions: sensing layer, communication layer, and computing layer. The Artificial Intelligence (AI) acting as the enabling bridge among the three layers.

services remotely using information technologies to treat patients [6]. The pandemic is forcing healthcare providers to stress on services such as telehealth and remote monitoring. Prominent tech enablers such as Microsoft, Google, and Amazon Web Services (AWS) are all pushing deeper into obtaining better digital and automated platforms for remote health monitoring after the pandemic impact [7].

IoT/IoMT devices in the healthcare industry will generate massive data, including some highly sensitive data, and some health data require quick analysis and immediate decision-making. To facilitate the healthcare industry with decision-making intelligence, usually remote cloud server-based AI analytics is used in various use-cases. However, the cloud computing-based paradigm will not be suitable for the upcoming massive demand. Furthermore, it will consume a tremendous amount of network bandwidth for mass-scale deployment and will require a considerable amount of time. Along with these, there is massive privacy concern due to sending sensitive health data to the remote server. For minimizing these challenges to some extend, researchers have focused on combining edge/fog computing, Mobile Edge Computing (MEC), and cloud computing to mitigate these challenges [8].

Currently, 4G and other communications are used to assist healthcare services and applications throughout the world. These technologies play a vital role in facilitating the eHealth and smart health industries. However, with the rapid growth of remote healthcare

services and IoT and IoMT devices, the amount of data in different formats and sizes is expected to increase drastically over the next few years. Such massive and diverse data requires unique solutions considering end-to-end delay, network bandwidth consumption, privacy issues. The current communication technologies are not likely to be sufficient for fulfilling the requirements of dynamic and time-sensitive future smart and connected healthcare services [9]. Hence, the future-generation fully 5G and B5G networks can be utilized to tackle the challenges in the communication layer of IoT to assist the healthcare system's services. With the 5G/B5G-enabled communication advancements, healthcare providers can monitor patients remotely and transmit real-time data for preventative care and other individually-tailored healthcare requirements.

Therefore, as shown in Fig. 1.1, this thesis investigates the development and viability of lightweight AI models for solving challenges in three considered layers of IoT (i.e., sensing, communication, and computing layer), which can be combined with the envisioned and efficient 5G/B5G-aided networks to conceptualize next-generation smart and connected remote healthcare systems. The term "lightweight technique" in this thesis refers to the task of designing such an AI-aided system so that the decision-making process consumes less time and memory, especially in the inference/running phase of the considered use-case. Thereby, we investigate the development of such AI systems that consume lower memory and time compared to other traditional methods so that they can be regarded as lightweight, and we can utilize these techniques with resource-constrained sensors for localized and automatic decisions in the health monitoring use-case. To sum up, the seamless integration of lightweight AI, ultra-edge IoT, and future-generation beyond 5G networks can facilitate the healthcare providers as follows:

- Decreasing massive privacy concerns by restricting raw health-related data transmission over the internet.
- Reducing end-to-end transmission delay due to not sharing raw data for ultra-edge analytics and only sharing learned knowledge for collaborative learning paradigms.
- Minimizing network overhead via reducing massive bandwidth consumption.
- Facilitating faster and smoother packet (i.e., model parameters) delivery in case of collaborative and decentralized learning via distributed ultra-edge nodes.
- Providing a scalable system to meet the ever-growing demand and handling massive medical data. Hence, improving the flexibility, scalability, and adaptability of the system.
- Empowering real-time decision making for remote health monitoring with localized intelligence and ultra-low latency via lightweight AI and ultra-edge IoT node.

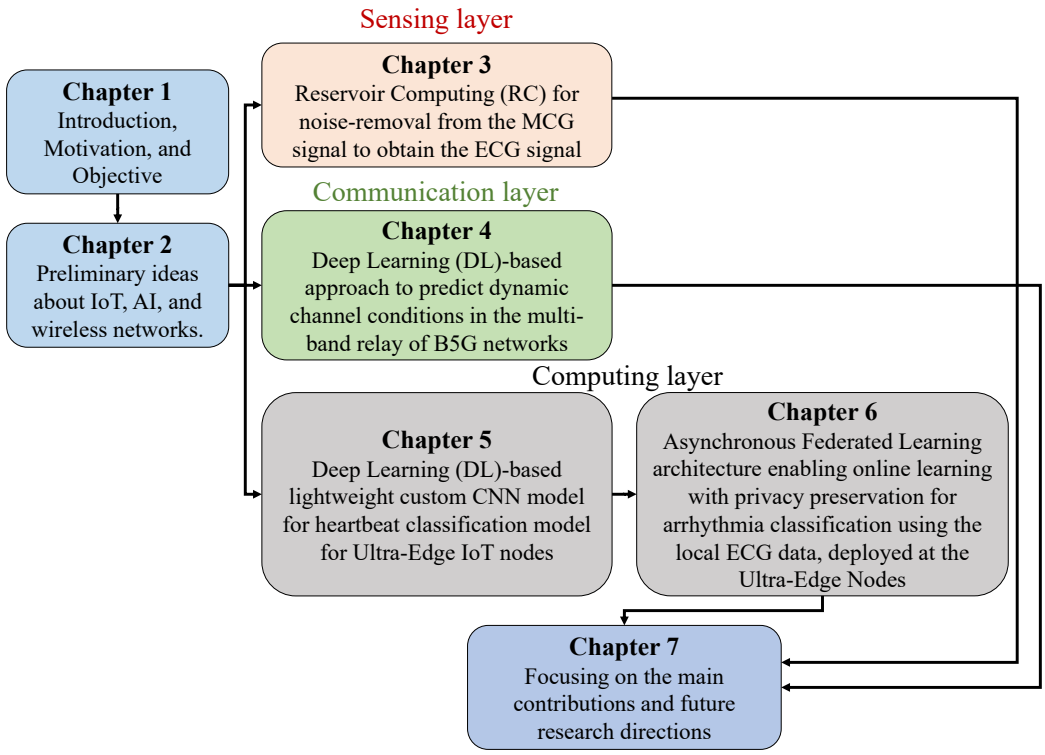


Figure 1.2: Organization of all the chapters of the thesis.

Fig. 1.2 outlines the organization and brief contributions in all the chapters of the thesis. Therefore, this thesis investigates the development and viability of lightweight AI models, which can be combined with the envisioned and efficient B5G networks systems to conceptualize next-generation smart and connected remote healthcare systems.

The remaining chapters of this thesis is organized as follows:

Chapter 2 provides a general overview of the fundamental idea behind the technologies and methodologies utilized in the later chapters of the thesis.

Chapter 3 considers the research problem of cardiac magnetic signal sensing use-case by analyzing biosignal. Smart AI-based IoT sensors are typically envisioned to have onboard intelligence and can interact collaboratively among themselves or with a remote server through the internet. To achieve the high level of automation required in today's intelligent IoT applications, sensors incorporated into nodes must be efficient, context-aware, reliable, and connected. To design smart edge computing-enabled IoT sensors with high sensitivity and low-energy, recently developed spintronic sensors have a massive potential. These sensors are capable of biosignal detection (i.e., Magnetocardiography (MCG), MagnetoEncephaloGraphy (MEG)) at room temperature and can be used to analyze health data on-chip. However, one of the significant challenges associated with these sensors is the $1/f$ noise, which is inherently present in such devices, interfering with the bio-signals of

interest. Standard linear filtering techniques are not suitable to denoise the signal, and we need to develop a noise-reduction process that is both efficient and can be integrated with the sensors. Hence, we employ Reservoir Computing (RC) for noise-removal from the MCG signal to obtain the ECG signal while conserving computing resources. Simulation results (low training time and memory requirements) demonstrate the RC model's potential when coupled with the sensors for continuous health monitoring. The efficiency of the proposed method is also observed to be superior to the conventional methods. The encouraging experimental outcomes can be the basis for the physical RC implementation to combinedly sense and analyze the biosignals at the ultra-edge nodes of the IoT environment.

Chapter 4 presents the considered challenge in the communication layer/plane of the IoT to assist faster data transmission of massive IoT data (i.e., medical/health data, model parameters from collaborative learning, and so on) generated from the ultra-edge IoT nodes. The AI-aided solution can facilitate several kinds of data delivery, such as medical or health-related data generated from biomedical and IoMT sensors, and also suitable for generic types of data generated by typical IoT devices. Here, we focus on a significant research challenge of spectrum scarcity and overloading for the next generation B5G networks. An AI-enabled technique is designed for the predictive smart channel selection method to unravel the potential hurdles associated with the dynamic channel conditions in the multi-band relay of B5G networks. A lightweight Deep Learning (DL)-based approach is proposed to select the appropriate channels. Our DL-based customized Convolutional Neural Network (CNN) model demonstrated efficiency in determining the best channel to transmit and receive data based on its quality. Two proactive channel assignment strategies referred to as controlled and smart prediction schemes are employed to compare the performances of shallow and deep variants of the CNN model. Our proposal is evaluated on multiple publicly available datasets from diverse network systems. The proposed technique outperformed existing machine/deep learning-based methods by proactively predicting the quality of the available channels and selecting the most suitable channels in multi-band relay systems. Thus, this paper can be regarded as a pioneering research work to encourage researchers and industry experts to consider adopting the proposed AI-based technique to enhance spectrum and energy efficiency while offloading massive IoT/IoMT traffic in next-generation B5G networks.

In chapter 5 and chapter 6, in the computing layer/plane of IoT, we shed light on the need for lightweight AI analytics at the ultra-edge nodes for localized intelligence to facilitate a more secure, faster, and localized intelligence to serve the rapidly growing surge of remote health monitoring. Arrhythmia (i.e., irregular heartbeat) classification is considered as the use-case for health monitoring. We press the need to go beyond the conventional cloud-based AI analytics and explore how to incorporate intelligence into the ultra-edge IoT sensors for obtaining more secure, faster, and localized on-chip intelligence to serve the rapidly growing

surge of remote health monitoring. In both chapters, to evaluate the proposals, we have employed publicly available four different real datasets from PhysioNet, complying with the ANSI/AAMI EC57:1998 standard. We considered four heartbeat types as class labels to detect arrhythmia.

Thereby, chapter 5 focuses on designing a lightweight AI model meeting the demand of low time and memory as the ultra-edge nodes are resource-constrained. Accordingly, we have developed a deep learning-based lightweight custom CNN model for heartbeat classification model defined as DL-LAC, which utilizes raw single-lead ECG without any manual pre-processing (i.e., noise-filtering from ECG). The proposed AI method is compared with traditional machine learning techniques and the DDE-based optimization technique. Experimental results show that the proposal can detect arrhythmia with high efficiency and low computational overhead (i.e., memory and time consumption), making it a viable solution for integrating with the ultra-edge IoT nodes.

Finally, chapter 6 extends the proposed lightweight AI methodology from the previous chapter to design a distributed collaborative online learning architecture for remote long-time health monitoring. Here, we introduce an asynchronously updating federated learning architecture (Async-FL) for mobile and deployable ultra-edge nodes to obtain the AI model's online learning capability while also maintaining privacy concerns. The simulation outcomes reflect the proposal's effectiveness in addressing the aforementioned research objective.

Lastly, we summarize and conclude the thesis and put forward some future research directions in Chapter 7.

Chapter 2

Background

This chapter depicts an overview of the preliminaries of the three considered layers/planes of IoT (i.e., sensing, communication, and computing). We discuss the existing theories and methodologies that are fundamental to perceive the remaining parts of the thesis.

2.1 Preliminaries: IoT Sensing Layer

Smart world is a trending term that envisions lower energy consumption, excellent public services, and better quality of life for human beings. The IoT and AI are compelling plat-

forms connecting various sensors around us to the Internet, providing ample opportunities to fulfill smart living and smart healthcare. The first considered plane of IoT is the sensing plane/layer where the sensors are adopted in such as way that these can be used to read the health data, remove unwanted noise, and intelligently analyze with AI logic.

Medical and healthcare applications still linger behind with only a few applications, such as glucometers, ECG management, cancer prediction, biomarker identification, blood pressure monitoring, and so on. There is a need to develop more point-of-care diagnosis and prognosis-based biosensors, helping experts diagnose diseases as early as possible by making intelligent judgments to enhance decision-making. Smart biosensor devices will allow people to live a smart and connected lifestyle from the healthcare perspective and in many other aspects of life. The advancement of artificial intelligence, machine learning, big data analytics, next-generation wireless network, and the IoT/IoMT have opened up the research gap of integrating these enabling technologies with biosensors for real-time on-premise and remote monitoring, diagnosis, prediction, and decision-making [10].

Traditional monitoring systems with limited sensors and wired communication can merely collect fragmented data in the application domains. Furthermore, for IoT-based health monitoring, conventional sensors without embedded intelligence are assumed to be used to sense health-related data/parameters and then use a cloud-based centralized server to conduct AI analytics. However, conventional IoT sensors lack the on-chip intelligence that is essential for ultra-edge analytics. To address this challenge, IoT devices utilizing a spintronic-technology-based can be considered as a suitable solution to obtain smart sensing with intelligence for a prolonged period of time [11, 12]. Hence, this sub-section depicts a conceptual understanding of the spintronic-based sensors and its potential for embedding intelligence to obtain the envisioned ultra-edge logic-in-sensor concept.

A smart logic-in-sensor is a device that has integrated electronics, and it can perform certain functions such as logic functions, two-way communications and possess localized intelligence that make them proficient in making decisions. Sensing with traditional IoT sensors or biomedical devices can be challenging for effective and easy remote/home monitoring. For example, ECG cannot be applied effectively to monitor the cardiac state of patients. Hence, new applications with new values in the IoT industry for cardiovascular monitoring need to be considered. In this regard, spintronic sensors using Magnetic Tunnel Junction (MTJ) devices offer a decisive advantage in terms of high sensitivity and portability and its ability to facilitate the ultra-edge logic-in-sensor architecture with embedded intelligence. Spintronic sensors can provide information on the magnetic field and magnetic-field-related parameters. The tunnel magneto-resistance (TMR) effect in MTJ devices developed by the physicists has tremendous potential for sensing human heart and brain signals [13].

The TMR sensor consists of an MTJ that operates at room temperature and has been

developed for spintronic devices such as magnetic random access memory. The sensor's magnetic random access memory characteristic enables the potential of embedding lightweight AI models for ultra-edge analytics. These sensors are envisioned to measure the data more accurately with noise filtering capability and analyze data with high accuracy, which makes them viable for ultra-edge analytics of medical data. Smart sensors are used for monitoring and to do the controlling mechanism in many industrial applications. One of the significant differences between a logic-in-sensor and a conventional sensor is that the smart sensor is tinier in size and faster than the traditional type, and it is more accurate too. As the envisioned logic-in-sensors are smaller in size than a typical sensor, they are resource-constrained. Thereby, when our goal is to integrate AI logic into these sensors and to achieve that, the developed AI paradigm should be lightweight. In this vein, in this thesis, we have investigated different AI modules which can be integrated with the ultra-edge smart logic-in-sensors for diverse health monitoring use-cases.

2.2 Preliminaries: IoT Communication Layer

In this sub-section, we focus on the fundamental architectures and basics ideas behind the communication plane of IoT. Firstly, the conventional concepts of the IoT communication plane are discussed. Then we shed light on the necessity of intelligent resource allocation for efficient data delivery of IoT communication plane.

2.2.1 Traditional Cloud and Edge Architectures

The IoT has immensely transformed the way businesses work in recent years, and the industry has seen a massive shift from on-premise software to cloud computing. After emerging during the mid-1990s, Cloud computing is a rapidly developing and ubiquitous computing architecture for AI analytics and smart health analytics. This paradigm is the on-demand availability of computer system resources, especially data storage (remote cloud storage) and computing power, without direct user direct active management. One of the significant advantages of cloud computing services is that an organization can avoid the upfront cost and complexity of maintaining its own IT infrastructure and, instead of that, use a remote computing platform to perform a particular task. In most cloud services, charges are pay-per-use, based on remote dedicated servers for computing tasks, sometimes also supports parallel computing. By storing and processing data using cloud technology, we have liberated ourselves from the relentless trouble of accessing data in a limited manner. Due to cloud computing's emergence, additional features can be accessed on our smart devices and IoT devices without pondering too much about investing in computing and memory capacities. A broad range of industries, including biomedical informatics enterprises, can take advantage of the new computing paradigm [14]. Existing research in the literature state

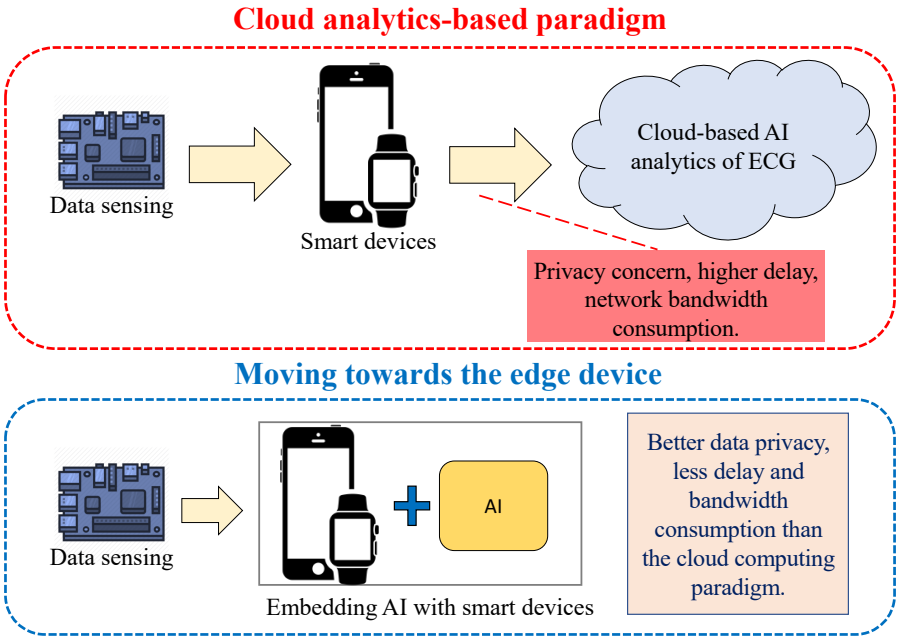


Figure 2.1: Conventional architectures for IoT-based AI analytics.

that cloud computing has the potential to overcome health data management, and analysis challenges [15]. Some striking features of cloud computing are resource outsourcing, a large number of machines, automated resource management, virtualization, Parallel computing, etc. [14]. However, as depicted in the first part of Fig. 2.1, cloud-based AI analytics has some challenges, and it is not suitable for remote health monitoring for a long time, especially when the number of subjects is rising. For the cloud-based framework, the health data need to deliver over to the remote server via the internet which causes the privacy concern, network bandwidth consumption, and massive delay.

To address the challenges of the cloud paradigm, the researchers and industry experts focus on edge and fog computing, which brings the cloud's capabilities close to the end-user or end-device. There are debates around edge computing and fog computing; however, both have almost similar objectives. Both edge and fog paradigms shift the AI analytics of data closer to the source of data generation. The main focus of doing so is to reduce the amount of data sent to the remote cloud server. These paradigms decrease delay or latency and enhance system response time, especially in remote time-sensitive tasks such as health monitoring. One of the main differences is that fog computing can include running intelligence on the end-device such as IoT devices. Moving computing power closer to the edge of the network will help degrade cost and enhance security.

To sum up, the main difference between the IoT devices communicating with a remote cloud server is that the bi-directional communication with a cloud server can take up to several minutes, while it may only take up to a few milliseconds when interacting with AI

analytics placed near the device. While cloud analytics still remains the primary choice for storing, analyzing, and processing data, different organizations are gradually progressing towards edge and fog computing-based intelligence to enhance data privacy and reduce costs (i.e., bandwidth, delay overhead). Keeping these fundamental architectures and their challenges in mind, we focus on the compelling necessity of bringing lightweight AI analytics from the cloud to the ultra-edge sensor itself to analyze the private health data where the data will be generated.

2.2.2 Wireless Technologies in IoT

IoT-aided fifth-generation (5G) and beyond 5G (B5G) system will be a game-changer in the future generation [16]. It will open a gateway for new wireless structures and smart services. Existing cellular network LTE (4G) will not be sufficient and efficient to meet the demands of multiple device connectivity, high data rate, more bandwidth, low-latency Quality of Service (QoS) [17]. The IoT encompasses a large number of seemingly connected devices around us, which includes all the smart devices, machines, flying devices such as Unmanned aerial vehicles (UAVs). IoT interconnects an abundance of devices to a network that readily shares information. Since IoT is a widely diverse and multifaceted realm, a one-size-fits-all communication solution cannot efficiently accommodate the communication aspect, especially in the case of the massive data load generated from remote health monitoring using IoT. While all IoT devices transmit and receive information via wireless technologies, they don't do it identically using the same band or channel. There are different options for connectivity, and some are better suited to specific applications than others. Factors like battery life, range of coverage, power requirements, and throughput must all be taken into account when deciding which option to pick for any particular situation. Furthermore, the IoT incorporates multiple long-range, short-range, and personal area wireless networks.

Along with other IoT sectors, the healthcare sector is generating a tremendous amount of data due to the leap in remote monitoring of numerous subjects, and it is expected to grow in the future as well. These massive IoT data needs to be delivered to the server in terms of a cloud-based centralized analytics approach, or the local training knowledge needs to be shared with the surrounding IoT nodes for distributed online training. With the emergence of 5G, among diverse available Radio Frequency (RF)-based options, some of the significant evolving wireless technologies in the IoT are ZigBee, Z-wave, LPWAN, Bluetooth Low Energy (BLE), Radio Frequency Identification (RFID), LoRa, and the different versions of Wi-Fi (i.e., 2.4 GHz and 5 GHz). All of these enabling wireless technologies for the IoT communication layer vary in range, power consumption, and data rate [18]. Hence, to maximize throughput and minimize delay, among several other challenges of IoT wireless communication, one of the striking challenges of an enhanced IoT experience is to make use of an AI-empowered lightweight model for intelligent selection of the channel among the

diverse range of channels and bands.

2.3 Preliminaries: IoT Computing layer

This sub-section depicts the fundamental ideas of diverse AI-enabled methods exploited in the remaining chapters of the thesis for tackling challenges in different smart health use-cases. Firstly, we discuss the traditional machine learning-based techniques, and then we shed light on the neural network and deep learning techniques adopted in the thesis.

2.3.1 Fundamentals of Machine Learning Models

Here, we discuss the fundamentals of some simple ML techniques adopted in the later chapter of the thesis for the purpose of health data analytics (i.e., classification) and intelligent delivery of health data by prediction network's channel quality. The ML techniques discussed in the following are under the supervised machine learning subset, requiring historical data in the learning phase.

2.3.1.1 K-Nearest Neighbors

K Nearest Neighbour (KNN) algorithm [19] is a lazy and simple machine learning algorithm. KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in different domains. In KNN classification, the output is a class member. An instance is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors. The neighbors are found out by calculating the distance from a test instance and all the training instances. Different measures can be used for calculating distance, such as Euclidean distance, Manhattan distance, Minkowski distance, Hamming distance, etc. The value of K is a generally positive integer and usually small. If $K = 1$, then the object is directly appointed to that individual nearest neighbor's class. However, if the value of $K = n$ where n is the number of instances in the training data, then the algorithm becomes an eager learning algorithm as it will explore all the examples. Traditionally, hyperparameter tuning is conducted for selecting the best value of K. The drawback of the KNN method is that the prediction time is relatively costly as it finds the distance between one point and all other data points. Some of the conventional distance measures (i.e., Minkowski distance, Euclidean distance) can be denoted as follows in Eq. 2.1 and 2.2:

$$d_M(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \tag{2.1}$$

$$d_E(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.2}$$

2.3.1.2 Random Forest

Random Frest (RF) is a supervised ensemble learning method for both classification and regression. Multiple decision trees are formed at the time of training, and a bootstrap sample technique is used for outputting the class that is the mode of the classes of the individual trees for classification objectives. A large number of nearly uncorrelated trees operate as a combination, and a set of decision trees form a randomly selected subset of the training set. All the leaf nodes of the trees are designated with a class. Then it aggregates the majorities from all the trees to pick the final label of the test instance. In the RF algorithm, measuring the quality of a split can be calculated utilizing gini impurity or entropy. The gini impurity and entropy can be expressed as the Eq. 2.3 and 2.4:

$$Gini = \sum_{i=0}^{c} p_j^2 \tag{2.3}$$

$$Entropy = \sum_{i=0}^{c} p_j \log_2 p_j \tag{2.4}$$

Here, c is expressing the number of classes in the considered problem, and p_j is the probability of the class j. Computationally, entropy is more complicated since it uses logarithms, and consequently, the calculation of the Gini impurity will be more suitable for resource-constrained edge nodes.

The performance of the RF algorithm is related to the correlation among trees and the strength of the individual trees. If there is a more significant correlation among the trees, it will decrease the error, whereas each tree's strength will increase its performance. One other important property of the RF is that they are instrumental when determining feature importance as essential features tend to be at the top of each tree. Therefore, the importance score of features can be calculated after applying the RF classifier. However, one of the drawbacks of the RF algorithm is that, in the inference phase, it is slower than many ML algorithms in creating predictions, which can be a challenge while deploying the model with edge nodes for health data analytics.

2.3.1.3 Linear Regression

Linear Regression (LR) is also a supervised ML algorithm that is mainly utilized to predict or forecast the value of an attribute by modeling the relationship between a scalar response

and one or multiple variables. A continuous range of values is considered for prediction rather than classifying them into different categories. LR model assumes a linear relationship between the input variables and the output variable. Mainly there are two main types of LR technique: Simple regression and Multivariable regression. Denoted in Eq. 2.5, the simple LR algorithm employs a conventional slope-intercept form, where m (slope) and b (bias) are the variables that the algorithm will learn and generate accurate predictions with minimum error. Here, X represents the input data, and y is the variable to predict.

$$y = mX + b \tag{2.5}$$

When there are multiple input variables, it is referred to as multiple linear regression. The Eq. 2.6 denotes the expression of multiple linear regression. Here w_i represents the coefficients or weights that the model will try to learn. The variables a, b, and c represent the attributes/feature or distinct pieces of information about each observation.

$$f(a, b, c) = w_0 a + w_1 b + w_2 c \tag{2.6}$$

The simple LR model can be interpreted graphically as a best-fit line between the data sample. In contrast, the multiple LR can be depicted as a plane (in 2 dimensions) or a hyperplane (in higher dimensions). Several procedures can be utilized to train the linear regression equation from data, and the most common is Ordinary Least Squares. The Ordinary Least Squares procedure attempts to minimize the sum of the squared residuals, which means that for a regression line through the data, we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors collectively.

2.3.1.4 Auto Regression

A statistical model is autoregressive (AR) if it predicts future values based on past values. An AR model might seek to predict the relay network's channel quality utilizing past samples in order to deliver health-related data. The order of an AR is the number of prior values used to predict the present. In terms of multiple linear regression, the variable of interest is predicted using linear combinations of values, whereas, in an AR model, we forecast the variable of interest using a linear combination of historical data of the attributes.

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of the variable's past values. The term autoregression indicates that it is a regression of the variable against itself. Thus, an autoregressive model of order p can be written as Eq. 2.7.

$$y_t = c + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \epsilon_t \tag{2.7}$$

Here ϵ_t is white noise. This model acts similar to a multiple regression but with lagged values of y_t as predictors. We refer to this as an AR(p) model, an autoregressive model of order p. An AR(1) autoregressive process is adopted when the current value is based on the immediately preceding value. In an AR(2) method, the current value is predicted based on the previous two values, which means the value at time t is predicted from the values at times $t-1$ and $t-2$.

2.3.2 Fundamentals of Deep Learning Models

This sub-section illustrates some fundamental ideas behind Neural Networks (NN) and Deep Learning (DL). NN is a sub-category of machine learning, and the deep learning sub-field mainly originated from the NN as well. Neural networks and deep learning methods can be categorized into three categories such as supervised, semi-supervised, and unsupervised learning. Additionally, Reinforcement Learning (RL) is also considered another type of DL/NN approach. The NN-based deep learning approach is also called universal learning because we can apply it to almost any application domain [20]. The DL can be considered as a viable approach when we have a lot of data in hand to analyze. Therefore, these are appropriate techniques to solve diverse problems in the computing plane of IoT that generates a tremendous amount of data continuously. Among various available variations of the DL algorithms, in the following chapters, we have mainly utilized diverse custom Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Echo State Network (ESN)-based Reservoir Computing (RC) technique in order to propose solutions to the selected problems in the mentioned three IoT planes.

2.3.2.1 Artificial Neural Networks

Artificial Neural Networks (ANN) are mathematical models typically designed and utilized to solve diverse problems such as pattern recognition, autonomous control, and smart health through learning and a vast amount of historical data. Many organizations employ neural networks to solve problems in multiple fields, and the economic sector traditionally falls under the operations research domain [21]. Similar to the biological neural network, the basic building block of an ANN is called an artificial neuron, also referred to as a node. Hence, ANN can be considered an information handler model comparable to the human brain's biological nervous system function.

Fig. 2.2 exhibits a high-level architecture of an ANN model. Typically, an ANN model consists of input, hidden, and output layers. The input layers can have up to n neurons or nodes, which is equal to the number of features or attributes. In the hidden layers, there can be multiple hidden layers in between the input and output layers. The output layer contains n number of nodes in terms of the typical classification task, where n is

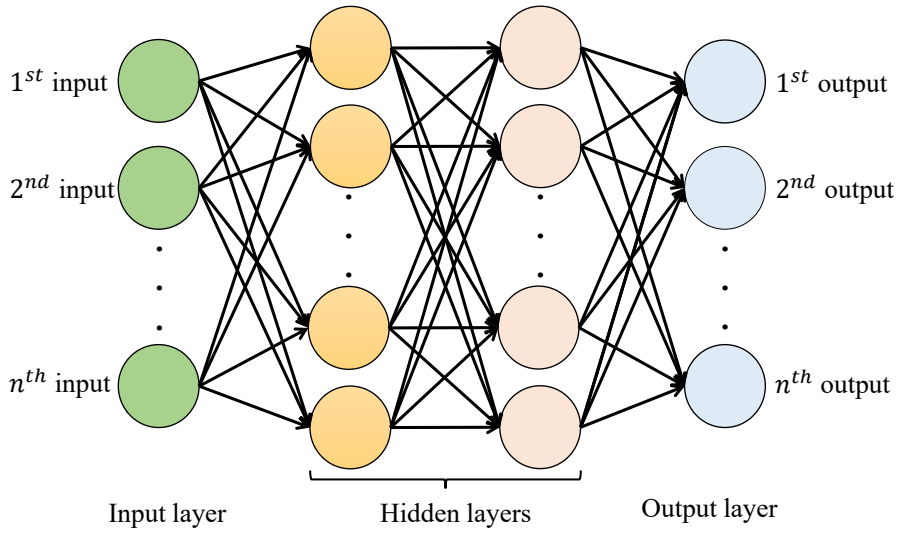


Figure 2.2: Architecture of a typical ANN model.

the number of classes. In terms of regression/prediction tasks, usually, the output layer contains only one node/neuron. The hidden layers are independent of one another; that is, a specific hidden layer can have an arbitrary number of neurons or nodes. Generally, the number of designed hidden nodes is higher than those of input nodes/neurons. In order to determine the output of a node, firstly, the weighted sum of all the inputs, weighted by the weights of the connections from the information of the node, is calculated. A bias term is added to this sum and then passed through an activation function to produce the output. As most real-world problems are non-linear in the pattern, the activation functions are applied to introduce non-linearity in the model. Various optimizers (i.e., gradient descent-based optimizers) are adopted to learn the weights or model parameters. The weights are adjusted by the backpropagation technique until the end of the learning phase. We can use Eq. 2.8 to summarize the computation of the ANN model:

$$Output_{ANN} = \mathrm{f}\left(\sum_{i=0}^{n} w_i x_i + b\right) \tag{2.8}$$

Here. w_i is the weight of the input x_i, b is the bias term, n is the number of inputs for the node, and $f(.)$ denotes the activation function. Typically, the weights for all the nodes are initialized randomly. In the forward pass, the element-wise non-linear activation function is applied to the matrix dot products. Some popular and frequently-used activation functions can be expressed as follows:

$$\mathrm{sigmoid}(x) = \frac{1}{1+e^{-x}} \tag{2.9}$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.10}$$

$$\text{ReLU}(x) = \max(0, x) \tag{2.11}$$

$$\text{ELU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\left(e^x - 1\right) & \text{if } x < 0 \end{cases} \tag{2.12}$$

Here, Eq. 2.9, 2.10, 2.11, and 2.12, refers to Sigmoid, Tanh, Rectified Linear Unit (ReLU), and Exponential Linear Unit (ELU), respectively, activation function equations applied on input x. Here, in Eq. 2.9, e is the Euler's number, and in Eq. 2.12, α is the scale for the negative factor.

In summary, ANN is a very flexible yet compelling DL model. ANN can be customized to approximate any complex function for getting an insight into any real-world problem. With the emergence of massive health-related data, the ANN can be considered a robust AI-based solution to many use-cases of both classification and prediction tasks of health data to achieve a smart health system.

2.3.2.2 Convolutional Neural Network

Recently, one of the most popular categories of deep NNs is the Convolutional Neural Network (CNN). CNN is a typical NN that is extended across space via sharing weights. Popular variations of the CNN are 1-D and 2-D CNN. The 1-D CNNs are generally customized to analyze tabular and time-series data, whereas the 2-D CNNs are widely adopted for pattern detection from image data. As 1-D CNN requires significantly less computational complexity than 2-D CNN, it is more suitable for lightweight real-time applications where resources are limited, such as the resource-hungry IoT sensors. A standard CNN has multiple layers, including the convolutional layers, sub-sampling layers, regularization layers, and fully-connected/dense layers. The dense layers and convolutional layers have parameters for training/learning; however, regularization layers and pooling do not have parameters. Various studies have concluded that CNN illustrates excellent performance in a diverse range of AI tasks [22].

The typical architecture of CNNs consists of two fundamental parts: the feature extractor and the classification part. In the feature extraction layers, the previous layer's outcome is used as the input of a particular layer, and the result of that layer is then passed to the next layer. Usually, the lower level or shallow layers are responsible for extracting higher-level features from the input data. With the propagation to the higher-level layers or the deep layers, the dimensions of the features are usually decreased depending on the size of the kernel size of the convolution. The max-pooling layers are also crucial for downsam-

pling the dimension of the features, hence, reducing the computation. As the sub-sampling layer, the max and average pooling are typically adopted to take the maximum and average values, respectively. After several layers of convolutional and sub-sampling layers, usually fully-connected/dense layers are used to conduct high-level reasoning. In the dense layers, like typical NNs, each neuron is connected to every other neuron to generate global semantic information. However, an atomic dimensioned (i.e., 1x1 Conv) convolution layer can replace the fully-connected layer. The last layer of CNNs is an output layer. For classification tasks, the softmax activation function is commonly employed, and for regression/prediction tasks, the linear and sigmoid activation function is mostly applied [23].

In order to design a robust CNN model, we need to determine optimal values for different hyperparameters. Typically, in different layers of the CNN architecture, we need to define a number of hyperparameters before the learning phase begins. Firstly, we have the number of layers used in the CNN model, which determines how deep or shallow a particular model is. In the convolution layers, typically, the kernel size and the number of filters are the hyperparameters. The hyperparameter of the sub-sampling layer is usually the pool size for reducing the size of features. The regularization layer (i.e., dropout layer) is often used in the CNN architecture to avoid overfitting. The dropout rate is the general hyperparameter in the dropout layer. Other than these hyperparameters, we also have a few other vital hyperparameter needed in the CNN model, such as the activation function, optimizer, learning rate, batch size, and the number of epochs [24]. By finding the appropriate values for the diverse set of hyperparameters of the CNN model, we can control the trade-off between the performance (i.e., accuracy) and computation burden (i.e., memory and time requirements) of the resource-constrained IoT devices in order to analyze health data. Hence, considering this trade-off, in the following chapter of the thesis, we have emphasized the hyperparameter optimization for constructing suitable custom CNN models to analyze medical data and efficient health data transmission.

2.3.2.3 Reservoir Computing

The Reservoir Computing (RC) [25] is closely related to Echo State Network (ESN) [26], and Liquid State Machine (LSM) [27] has emerged as a unique variant of a Recurrent Neural Networks (RNN). Unlike traditional DL/NN methods, only the output layer is trained rather than the entire network's weights in terms of the RC approach. The RC-based paradigm can be considered a great alternative to gradient descent techniques for training a recurrent neural network, as this requires only a simplistic and effective least-squares estimation rather than the more expensive fully non-linear optimization needed with fully training an RNN. In terms of the RC method, the learning/training is conducted only at the readout stage, as the reservoir dynamics are kept fixed. Surprisingly, in spite of being a straightforward method, the forecasting or predictive capacity of the RC technique can still

be quite robust even for complex, chaotic problems. The RC-based approach has gained popularity for its easy training process and its ability to deal with temporal data.

The main idea of RC lies in leveraging a fixed non-linear system of higher dimensions to obtain a rich non-linear representation of the inputs. After this mapping, a simplistic readout layer is employed to harvest the reservoir's state and train it to the desired output. In principle, given a complex enough system, this architecture should be competent in any computation. The overall RC structure has two main components or modules: Reservoir and Readout.

- Reservoirs: In the first part of the RC system, we have the reservoirs with fixed weights acting as black-box models. The reservoir in the reservoir computing paradigm has two properties: it must be made up of individual, non-linear units, and it should be capable of storing data. The reservoir module can have several units or nodes, and for implementation, it can be considered as a hyperparameter that needs to be tuned in order to design a robust model.

- Readout: The second part is known as readout, which is connected by a set of weights to the reservoir units. The readout performs a linear transformation on the reservoir's output, and the weights are trained by analyzing the spatiotemporal patterns of the reservoir after excitation by known inputs and utilizing a training method such as linear regression or a ridge regression [28].

One of the vital aspects to consider during the construction of an RC model is the activation function to characterize the reservoir's nodes' behavior. Typically, various types of activation functions, starting from a simple linear model to more elaborated non-linear ones, such as sigmoid function, have been employed in solving diverse problems using the RC method. The fundamental characteristic of RC is that the input weights and the weights of the recurrent model within the reservoir are not trained; instead, only the readout weights are learned in the learning phase. Besides the striking efficiency of training algorithms, some specific challenges are associated with the ESN-based RC method for solving a particular task. The specification of selecting the most suitable parameters for the RC model, such as input, reservoir connection, spectral radius, and the nodes' connectivity status, need to be cautiously selected for designing a robust model [29].

Furthermore, each reservoir node characteristic timescale and decay rate of information decide the length of data in the time domain. However, in some cases, it can cause similar obstacles to deep neural networks where the number of layers generates identical restrictions. Thereby, we need to be careful while designing the RC model to overcome these challenges. Along with the virtual reservoirs, the RC architecture is distinctively viable for hardware implementations using intelligent sensors such as spintronic technology-based

sensors. Thereby, solving resource-constrained complications, such as developing a smart health monitoring system by exploiting intelligent IoT sensors, can be accomplished by RC systems, as it meets the demands for low training cost and real-time processing in these applications.

2.3.3 Federated Learning (FL) architecture

Conventional ML/DL models comprise a central cloud server that hosts the trained model to make medical decisions. A drawback of this architecture is that the private data collected by local devices and sensors are sent back to the remote cloud server for processing and finally returned to the devices. Hence, this round-trip manner of AI analytics limits the ability to learn in real-time. To overcome this issue of traditional computing architecture, a distributed collaborative learning approach can be a viable alternate. Federated learning (FL) is an approach that downloads the current AI model and computes an updated model at the device using the edge node's local and private data (i.e., smart devices or sensors). Afterward, these local AI models are then sent to the cloud server, where they are aggregated iteratively, and an improved global AI model is sent back to the edge nodes. The term federated learning task is a loose federation of participating devices/sensors (typically referred to as clients) regulated by a central server. Each client holds the local/private data that is never transferred to the server. Alternatively, each client computes an upgrade to the current global model kept-up by the cloud server [30].

The distributed FL architecture presents us with a number of unique features that are crucial for many IoT-based sectors and remote health monitoring. As the FL architecture occurs on the device/sensor level, it enables real-time AI decision-making. Thereby, the time-delay usually occurs for the traditional approach because sending raw private data to the cloud is minimized via sharing only the model parameters or the knowledge gained from the data. Furthermore, since the local AI models are on the edge node, the prediction process works even when there is no internet connectivity available, making it highly suitable for remote health monitoring for a long time. However, along with various striking pros of the FL architecture, some challenges are associated with it. Communication is a critical issue because, in order to train an AI model using user's local data, it is essential to employ an intelligent communication method that can reduce the total number of required communication rounds and obtain faster convergence. Also, the FL architecture needs to be able to handle dropped devices in the network because, in a dynamically changing environment, all the devices/sensors may not participate in the learning process for the same amount of time.

Chapter 3

Noise-Removal from Spectrally-Similar Signals Using Reservoir Computing for MCG Monitoring

Continuous low-rate monitoring is an important IoT application, which requires high-fidelity in observing signals with low frequency. However, most sensors exhibit noise that is inversely-proportional to spectral frequency ($1/f$ noise). Because both the relevant signal and noise share the same spectral properties, standard linear filtering techniques cannot be used. We are looking into a special application for remote healthcare of the magnetic field sensing of cardiac activity, magnetocardiography (MCG). For such an application, we need to develop a noise separation method, that is also resource-efficient. Previously, we demonstrated AI-based removal of $1/f$ noise in MCG by a convolutional neural network coupled with gated recurrent units. However, it needs a large amount of data for training, requiring significant training time and computational power. In this work, we employ reservoir computing (RC) for noise-removal, while being conservative in computing resources.

3.1 Introduction

Recently, with the massive adoption of the IoT (Internet of Things) sensors and wearable devices, there has been a significant push toward collecting and analyzing health data of patients, elderly citizens, athletes, and ordinary users with an aim to enhance the everyday quality of life of humans. Cardiac health is a crucial concern in both developed and developing countries, and numerous smartphone-based applications are now available to passively monitor the heartbeat and even electrocardiography (ECG). However, the ECG data sensing using these commodity devices are not accurate compared to clinical-grade ECG machines, which are intrusive in general due to the need to place electrodes or leads on the human body. During the ongoing pandemic of COVID-19, the continuous remote monitoring of patients with cardiovascular conditions is needed to predict any complications, especially that care will be limited in an overwhelmed medical system.

Therefore, we need a cardiac sensing technology that is portable, non-intrusive, and compatible with IoT technologies. In this vein, an earlier work [13] demonstrated the acquisition of magnetocardiography (MCG) signals using spintronic magnetic tunnel junction (MTJ) sensors that operate at room temperature. Therefore, our spintronics-based monitoring of MCG is a high-impact solution for accurately monitoring the cardiac health of the masses. However, there are two interlinked challenges, and in Fig. 3.1 we propose to combine MTJ sensors [11] with AI-based signal and data analysis at the sensing node. The first challenge is that sending unprocessed data consumes a lot of communication bandwidth and power and constitutes a privacy risk. The second is the presence of noise, which requires cleaning before transmission. While MCG does not require contacting leads, it typically requires a magnetic-shielding to avoid environmental magnetic noise. Furthermore, the magnetic sensors themselves produce noise that is inversely proportional to spectral frequency ($1/f$ noise). The $1/f$ noise can be seen as correlated fluctuations at short time scales, which can obscure the similar dynamics of the cardiac activity [31]. Therefore, we need lightweight local AI solutions that can remove noise and monitor cardiac irregularities, such as arrhythmia, ischemia, and so forth. In this work, we are focusing on the more challenging task of noise-removal.

In our previous work [11], AI noise filtering based on a convolutional neural network (CNN) model with gated recurrent units (GRUs) reduced $1/f$ noise power by ten times compared to the moving average filtering. However, such a model required extensive training,

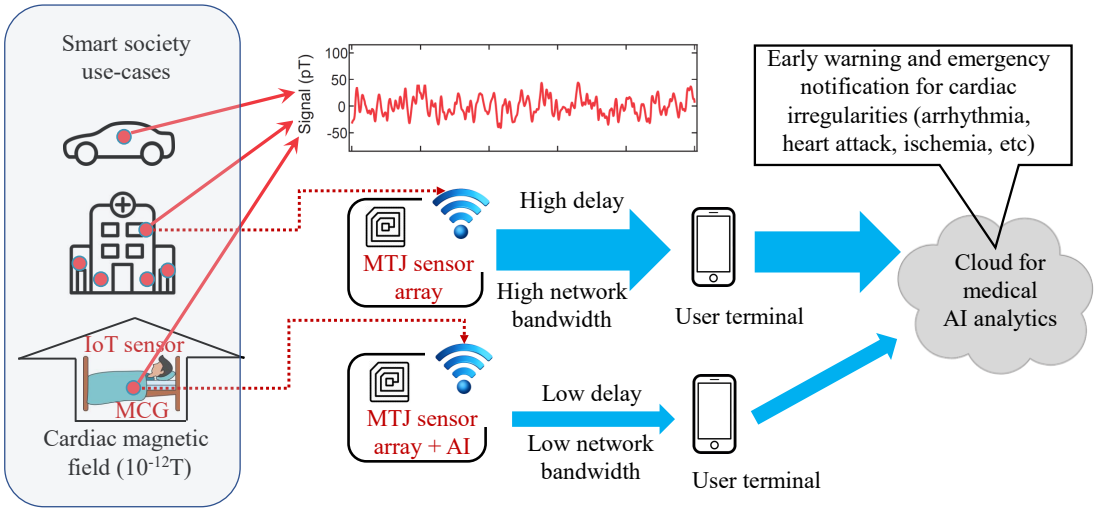


Figure 3.1: Continuous MCG monitoring with conventional and proposed paradigms without and with AI model for smart and localized noise processing and medical analytics using spintronic devices.

and in unpredictable environments, it may require retraining in the cloud. This repeated process is both time-consuming and costly. The excellent performance and training challenge come from the recurrent part of the model. The recurrent neural networks (RNN) can process the temporal context of sensor information and deal with multiple information from different sources, but RNN training is expensive and complex [32]. Recently, a subset of RNNs has come to prominence, called "*Reservoir Computing*" (RC) [33]. The *reservoir* here refers to a large network of interconnected state variables that are non-linear to their excitation with fixed connection weights as depicted in Fig. 3.2, akin to the dynamics of waves in a water tank, magnetization dynamics, non-linear optics, *etc.*. RC has been investigated intensively in many spin, optical, memristor systems [28]. The rich dynamics of the RC map temporal data sequences into different trajectories in a high-dimensional space (hyperspace). Then, the training task is only limited to a readout layer to produce useful classifications or inferences from the reservoir's transient states. In this work, we show how to utilize RC to remove $1/f$ noise from the MCG signal. Using computer-based simulations, we demonstrate the RC method's effectiveness in terms of accuracy, memory requirement, and execution time.

The remainder of the chapter is organized as follows. Section 3.2 presents preliminaries of spintronic sensors for embedding edge intelligence and describes the fundamental problem considered in this work. Next, in section 3.3, we provide an RC-based $1/f$ noise minimization of the envisioned smart IoT sensor. The performance of our proposed reservoir computing methodology is evaluated and compared with existing techniques in section 3.4. Finally, the chapter is concluded in section 3.5.

3.2 Preliminaries of Spintronic Sensors For Embedding Edge Intelligence and Problem Description

The MTJ sensors are made from two ferromagnetic metals (FMs) separated by an insulating tunneling barrier (e.g., magnesium oxide). The application of an external magnetic field (H) changes the magnetization angles of the FMs. Owing to the tunneling magnetoresistance effect (TMR), the resistance of MTJs depends linearly on H. Thus, MTJ sensors are simple to measure and can be combined with the integrated circuit fabrication process. MTJ sensors and structures are the main drivers of spintronics research and information storage applications, see Ref. [34] for a review. In the next generation networks, embedded edge intelligence is regarded as a crucial enabler for reducing bandwidth use, energy consumption, and end-to-end communication delay for network nodes [35]. Because embedding intelligence onto typically resource-constrained IoT nodes to facilitate edge computing is challenging [35], the implementation of the theoretically appealing logic-in-sensor concept still remains illusive to implementation at a mass-scale.

The main challenge for sensors is the noise at the low-frequency side of the spectrum. The MTJ sensor's noise is dominated by a $1/f$ character, similar to many other systems [36]. According to [37], this issue worsens in the high sensitivity area. The power spectral density (PSD) of low-frequency noise can be represented as [38]:

$$S_v \propto \frac{\chi}{f^{\beta}}, \tag{3.1}$$

where χ is related to the sensor sensitivity, f is the spectral frequency, and β is the exponent of noise spectrum.

The cardiac dynamics are slowly-varying and stochastic. Therefore, the signal-of-interest and noise share the same $1/f$ character, which introduces a problem of separating two chaotic sources. The linear time-invariant filters, such as the moving average filtering, are traditionally used for noise-removal but cannot separate cardiac activity noise with considerable efficiency. The deep learning (DL) filtering in our earlier work [11] showed a 10-times decrease in noise power over the moving average technique. However, the retraining overhead could be a potential bottleneck for the practical deployment of deep learning models to the resource-constrained IoT devices for monitoring time-series signals such as MCG. Therefore, the challenge in this research is to model an alternative solution, which is practical as well as lightweight, to significantly reduce the training time to mitigate the $1/f$ noise and provide the corresponding ECG from the noisy MCG with high accuracy. We propose an RC technique based on Echo State Network (ESN) for predicting the ECG signal from the sensed MCG signal.

3.3 Envisioned RC-based Technique for Noise-Removal

As a proposed noise filtering technique, we have adopted the ESN-based RC, which is considered a subset of RNNs with randomly fixed connectivity weights [33]. The RC-based noise filtering and ECG estimating method consists of a reservoir part represented as sparsely connected units, and a readout part depicted as a regression paradigm. This ESN-based RC method is suited for temporal or sequential data processing at a low cost, making it a viable technique for noise filtering from the MCG signal to predict the corresponding ECG. The reservoir parts are fixed in the learning phase, and only the readout part is trained [39]. Hence, this fast learning method can result in lower requirements during the training/learning phase [40]. This characteristic of the RC-based noise filtering makes it feasible for hardware implementation utilizing physical systems such as the spintronic MTJ sensors.

The ESN-based RC method maps input MCG signals into higher dimensional space to achieve a deep non-linear representation of the input. A linear combination between the high dimensional space and the readout units is learned for efficient noise filtering inputs in a lightweight manner. The following eqs. 3.2, 3.3 depict the states of the reservoir nodes and the output nodes:

$$x_{t+1} = x_t(1-\alpha) + \Omega(W_i u_t + W_r x_t)\alpha \tag{3.2}$$

$$y_t = W_o \times x_t \tag{3.3}$$

Here, W_i represents the connection weights between the input and the reservoir units, W_r represents the weights of the recurrent connections within the reservoir, which are not trained, and the W_o indicates the readout weights which are trained during the learning/training phase. The discrete time-step values are taken to be, $(t = 1, 2, 3, \cdots)$. At the time t, the state of each reservoir is represented by x_t, the state of the output vector by y_t, and the input vector by u_t. The element-wise activation function is denoted as Ω, and α indicates the leaking rate. Here, the leaking rate (α) regulates the update frequency of the states.

Algorithm 1 depicts the workflow of selecting the best RC architecture in the training/learning phase and then utilizing the model to predict unknown data in the test/inference phase. The algorithm's input section demonstrates the details of each of the inputs provided to the algorithm. Instead of utilizing the whole MCG cycle as input to the RC model, smaller segments were used [11], each with a segment size of λ MCG samples. Therefore, the pre-processed dataset, denoted by X_{data}, contains a collection of λ MCG samples as input features, and a single, corresponding ECG sample is the output label.

The algorithm begins with initializing the expected parameters in steps 1 and 2. In

Algorithm 1: Proposed RC-based training algorithm for $1/f$ noise filtering at MTJ-based sensor for automated MCG-ECG mapping.

Input : X_{data} (pre-processed dataset containing multiple instances of the schema {λ MCG samples as input features : 1 corresponding ECG sample as output}), U (the set of number of reservoir units), Ω (activation function), α (leaking rate)

Output: M_t (the parameters of the selected model)

1 $M_t \leftarrow \emptyset$
2 $\varepsilon_{min} \leftarrow \infty$
3 X_{train}, $X_{test} \leftarrow$ prepare the training and test data, respectively, from X_{data} based on the split ratios (s_{train} and s_{test}, respectively)
4 **foreach** *index* i = *1 to* $|U|$ **do**
5 $\quad M_i \leftarrow$ load the RC model employing $U[i]$, Ω, α
6 $\quad$ train the model (M_i) for X_{train} employing Eqs. (3.2) and (3.3)
7 $\quad \varepsilon_i \leftarrow$ compute performance of model (M_i) utilizing X_{test}
8 $\quad$ **if** $(\varepsilon_i < \varepsilon_{min})$ **then**
9 $\quad\quad \varepsilon_{min} \leftarrow \varepsilon_i$
10 $\quad\quad M_t \leftarrow M_i$
11 $\quad$ **end**
12 **end**
13 save the model parameters of the selected model (M_t)
14 return M_t

step 3, the pre-defined splitting ratios (s_{train} and s_{test}) are employed to distribute the X_{data} for training and test phases. From steps 4 to 12, the algorithm identifies the best performing RC architecture by adopting a varying number of reservoir units (U). The i_{th} RC model is loaded and trained in step 5 and 6 using X_{train}. Afterward, the model's performance is evaluated in step 7, employing X_{test}. As an initial performance indicator, we have considered the prediction error in Root Mean Square Error (RMSE) [41]. The i_{th} model's performance is checked with that of the previously evaluated models in steps 8 to 11 and updated accordingly. In step 13, the selected model (M_t) is saved. Finally, the selected model (M_t) is returned in step 14. With this trained model, the $1/f$ noise filtering for MCG-ECG mapping can be conducted at the MTJ sensor in an online manner.

3.4 Performance Evaluation

In this section, we evaluate the performance of our reservoir computing proposal and compare it with the traditional moving average (MA) filtering technique and the deep learning (DL) method described in [11]. We have adopted three different performance indicators to evaluate the proposed system with the MA and DL techniques, which give us a strong understanding of the effectiveness of the proposed approach. Firstly, we demonstrate the

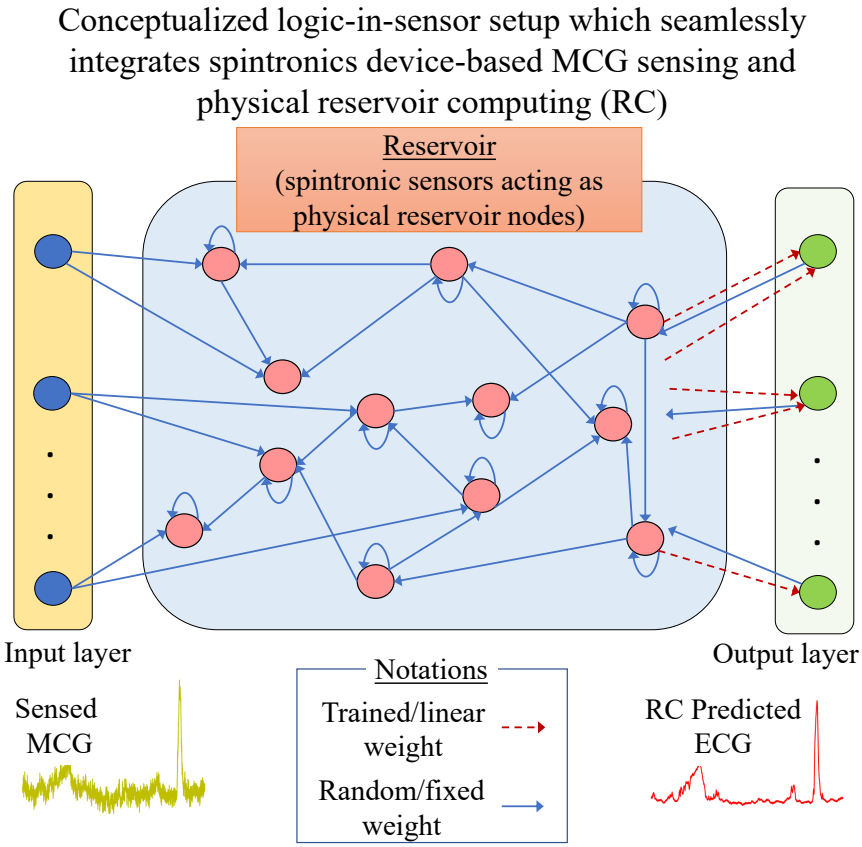


Figure 3.2: Reservoir computing (RC) model for MCG noise-filtering to obtain the ECG for continuous cardiac activities monitoring.

noise-filtered signal result visually to ensure that the proposed method can resemble the original ECG signal. Then, we determine the Root Mean Squared Error (RMSE) to calculate the error in signal prediction by each technique. Lastly, we illustrate the filtering efficiency in the power spectral density of the remaining noise after prediction.

3.4.1 Data Preparation

For performance comparison, we used the same data preparation methods as our previous work [11]. We synthesized MCG cycles from ECG cycles available in the open PTB Diagnostic Database [42–44], using the data preparation setup from our earlier work in [11]. We used the ECG traces from lead II of the healthy individuals. They were divided into single cardiac cycles, starting from the R peak to the next QRS complex, with the following sequence ($RSTPQRS$). The traces are upsampled to 3008 sample points without padded zeros, corresponding to a sampling frequency (f_s) of 2000 Hz. Then, the preconditioned ECG cycle is added to numerically-generated $1/f$ noise. We generated 100 MCG cycles with

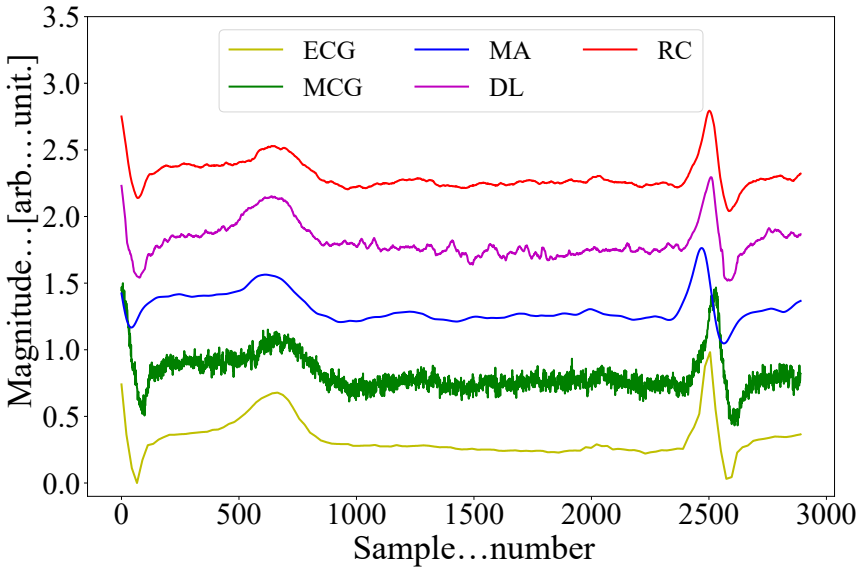


Figure 3.3: Performance evaluation demonstrating the original ECG cycle, synthetic noisy MCG cycle used as input, comparison between conventional moving average method, DL-based method, and proposed RC-based (RC-10) approach to process and remove the input signal's noise. The curves are vertically shifted for clarity.

different noise sequences for each ECG cycle. We generated the $1/f$ noise from a white noise floor of PSD $= 10^{-18}V^2$/Hz, based on the characters from real measurements [13]. The knee frequency between $1/f$ and white noise is set at $f_k = 250$ Hz $= 0.125f_s$. After the data collection and pre-processing, the MCG and original ECG cycles are used to train the RC model depicted in Fig. 3.2.

3.4.2 Simulation Parameters

The simulations for experimental results were conducted using Python 3 libraries (e.g., NumPy, Pandas, Matplotlib, and Scikit-learn) for data processing and visualization purposes. The RC and DL-based models are primarily implemented employing TensorFlow with Keras library in python. For all the experimental simulations, we have equally split X_{data}, i.e., both s_{train} and s_{test} are set to 0.5. In terms of the proposed RC method, we have examined different architectures considering $U \in \{10, 30, 50, 70\}$. For each RC architecture, the hyperbolic tangent (tanh) was used as the activation function (Ω). The value of the leaking rate (α) was fixed at 0.1. The weight values for W_i and W_r were initialized randomly. The number of input MCG samples per segment, λ, in both the RC and DL models was set to 50.

The RC-based proposal was compared with a DL-based (CNN and GRU) noise-filtering technique, the structure which was adopted from our previous work [11]. The epoch was

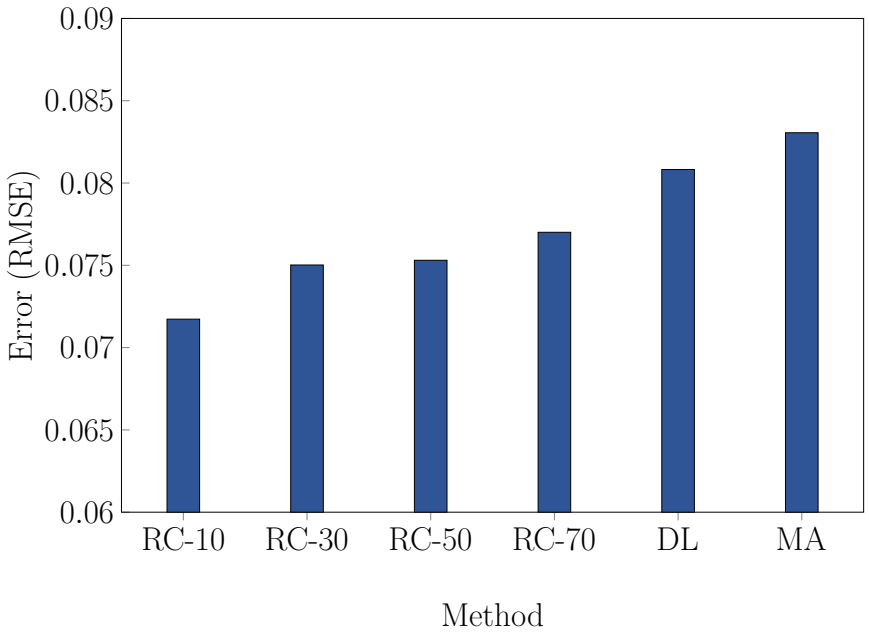


Figure 3.4: Inference performance comparison of RC with moving average and deep learning methods.The different RC architectures consist of 10, 30, 50, and 70 units, respectively.

set to 30 for the DL training phase. In terms of the moving average filtering technique, we have employed a striding length of 50 samples to filter the MCG to be consistent with the value of λ.

3.4.3 Results and Discussion

The simulations are conducted multiple times, and the average is used as the result. First, Fig. 3.3 demonstrates the filtering by the traditional moving average method, the deep learning method [11], and our proposed RC approach to jointly sense and minimize the $1/f$ noise in the input MCG signal. For ease of reference, we refer to the moving average filtering and deep learning method as MA and DL, respectively. Notice that the predicted ECG from the reservoir computing model is quite close to the original ECG/MCG cycle and successfully identifies the essential features such as the R-peak of the input ECG/MCG signal.

Next, Fig. 3.4 demonstrates the error in terms of the root mean squared error (RMSE) for the proposed reservoir computing-based model where the number of reservoir units is varied between 10, 30, 50, and 70. The errors incurred for these different configurations of the reservoir computing-based proposed method are compared with our earlier deep learning-based approach and the traditional moving average technique for noise processing. As shown in the result, when the number of reservoir units is set to 10, the error value is just above 0.07%. For increasing the number of reservoir units, the echo state network

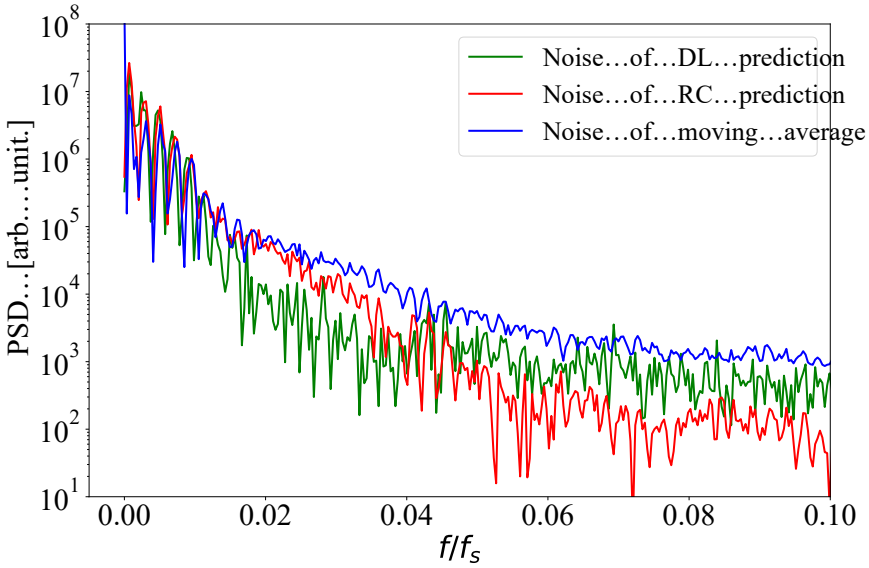
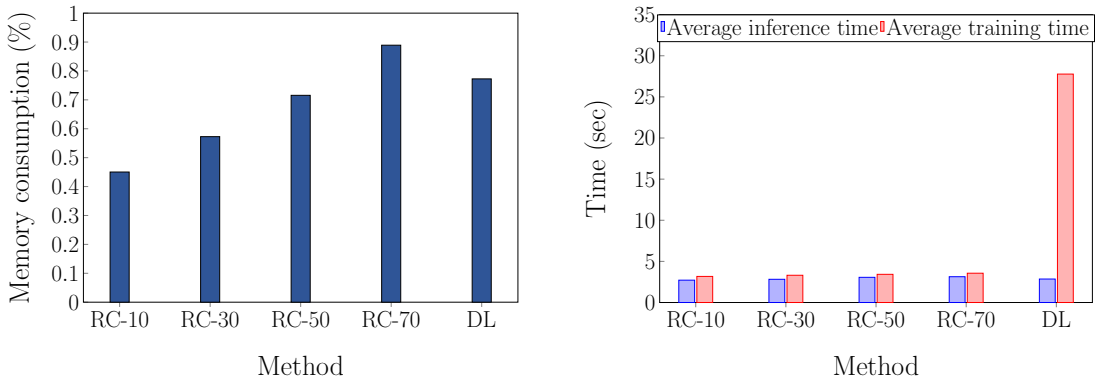


Figure 3.5: Dependence of noise power on spectral frequency for the RC-based prediction method, DL-based prediction, and the moving average filtering. Spectral frequency is normalized.



(a) Memory consumption rate in the training phase for different settings of RC and DL methods.

(b) Required time (per cycle) in the training and inference phases for different settings of RC and DL methods.

Figure 3.6: Memory and time requirement for the RC architectures and DL method. The different RC architectures consist of 10, 30, 50, and 70 units, respectively.

experiences more chaotic behavior in the state variables that slightly increases the error. Interestingly, the error remains much below 0.08% for the highest number of reservoir units considered (i.e., 70). On the other hand, the deep learning-based method results in the incurred error to reach 0.08%, whereas the moving average approach leads to the highest error.

Fig. 3.5 shows the filtering efficiency as seen in the power spectral density of the remain-

ing noise after prediction, i.e., PSD (predicted-original). Notice that the spectral frequency is normalized by the sampling frequency, i.e., f/f_s. The RC and the deep learning predictions show a noteworthy reduction in noise power compared to the moving average filtering technique, especially at the crucial low-frequency region $f/f_s = 0.01 - 0.03$. However, in that region, the DL method outperforms both MA and RC-based techniques. Interestingly, the proposed RC-based method exhibits better performance for $f/f_s > 0.04$ as the underlying ESN attenuates the high-frequency components that are not related to the QRS complex. Overall, we can observe that the collective noise reduction in different portions of the signal was quite significant in terms of the RC method compared to the other ones.

Next, whether the RC proposal is, indeed, lightweight or not, is shown in Fig. 3.6 by taking into account the execution time and memory overhead of the adopted AI-aided (i.e., DL method) methods for the resource-constrained IoT device. Here, we only compare the AI methods because the MA technique cannot be considered an AI-based method, and hence there are no training and inference phases in terms of the MA. Therefore, we primarily compare the time and memory required by the RC and DL methods. Fig. 3.6a demonstrates the memory consumption rates during the training phase of the RC proposal for the various settings of the proposal where the number of reservoir units is varied from 10 to 70. The memory consumption for the lowest and highest numbers of reservoir units was found to be 0.45% and 0.89%, respectively. In contrast, the memory required for the deep learning prediction-based method was 0.77%, which is significantly higher than the proposal with 10, 30, and 50 reservoir units.

On the other hand, Fig. 3.6b exhibits the average training and inference time requirements for the different configurations of our RC proposal and the deep learning prediction-based approach. Notice that while the average inference time for all the configurations is reasonably low (with increasing error rates, however, as earlier reported in Fig. 3.4). The average training time for all the configurations of our RC proposal is relatively constant and much faster than the deep learning method. Because RC training is limited to the output weights, the RC method has the advantage over the deep learning counterpart that requires retraining in the cloud, which incurs further network overheads and transmission delays. For remote health monitoring use-case employing distributed online learning, the training/learning time is more vital than the inference time. Hence, for an online learning scenario where the model training occurs continuously, the RC method will be more suitable than the DL method, even the there is not much difference between the inference time of both approaches. Thus, we may conclude that compared to other AI-aided techniques, our RC-based proposal is a lightweight one and suitable for deploying at the IoT nodes.

3.5 Summary

Recently developed spintronic devices have a vast potential for constructing smart and edge computing-capable IoT sensors with high sensitivity and low energy, particularly for magnetic biosignal detection (*e.g.*, MCG) at room temperature. However, their deployment is challenged by the $1/f$ noise, which is inherently present in such devices, interfering with the bio-signals of interest. This chapter addressed this problem in the cardiac magnetic signal sensing use-case and proposed a reservoir computing model based on echo state networks. Through simulations, we demonstrated that the RC model is lightweight in terms of much lower training time and memory requirements. Therefore, it is promising for continuous health monitoring. The accuracy of the RC method is also found to be better than the conventional moving average filtering and comparable with a recent DL approach. The simulation-based results are encouraging and can be regarded as a proof-of-concept basis for the physical reservoir computing implementation, using the sensors as physical RC model units to jointly sense and analyze the sensed bio-signals at the "ultra-edge" of the IoT ecosystem.

Chapter 4

Deep Learning-based Predictive Channel Assignment In Multi-Band Multi-Channel Relay Networks for Offloading Medical Data of Under-served Users

Multi-hop Device-to-Device (D2D) enabled relay networks are envisaged to be utilized by the Internet of Things (IoT) and massive Machine Type Communication (mMTC) traffic in multi-band multi-channel relay networks for offloading medical data of under-served users in the remote/rural areas. The emerging challenge of spectrum scarcity and overloading of cellular base stations can be addressed using such relay nodes in terms of spectrum and energy efficiency while transmitting medical or health-related data. In order to improve spectral efficiency, in this chapter, we intend to employ several frequency bands in the relay node of the Beyond Fifth Generation (B5G) networks rather than the traditional concept of specifying one channel on a specific band at a time. It will allow data transmission at multiple bands individually or simultaneously, accelerating the communication for a large number of users (i.e., subjects for remote health monitoring). A deep learning-based predictive channel selection method is leveraged to unravel the potential challenges associated with the dynamic channel conditions in the multi-band relay networks. For predicting the most appropriate channel based on its quality, Signal-to-Interference-plus-Noise-Ratio (SINR) is adopted as the metric, which is predicted by the proposed Convolutional Neural Network (CNN) model. The best modulation and coding rates of the predicted band are attained in order to transmit the packets received from the source or previous relay node to

the successive relay node/destination. Two proactive channel assignment strategies referred to as controlled and smart prediction schemes are employed to exhibit the performance of the shallow and deep-CNN models. The proposed model is evaluated on multiple publicly available datasets from diverse network systems and compared with several machine/deep learning methods. Our proposal leads to encouraging results for proactively predicting the conditions of the channels and choosing the most suitable ones in multi-band relay systems which will assist the communication efficiency while transmitting a vast amount of generic data and health-related data, especially in remote areas.

4.1 Introduction

In Beyond Fifth Generation (B5G) networks, wireless relay-based communication technologies (e.g., Device-to-Device (D2D) communications) are developing as a promising technique in the era of Internet of Things (IoT) and massive Machine Type Communication

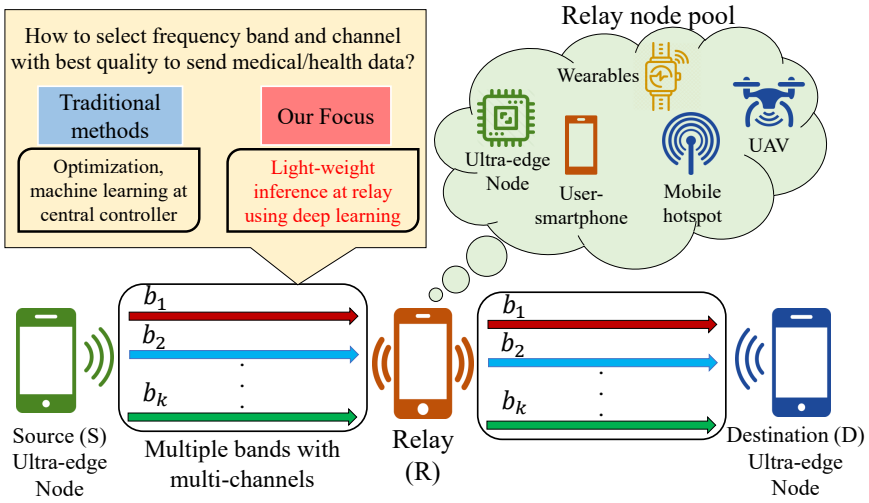


Figure 4.1: Our research focus compared to the traditional focus for selecting the best channel of multi-band relay networks.

(mMTC) [45–50]. The legacy cellular networks were initially employed to fulfill human-driven services and hence were unable to keep up with the surging IoT/IoMT traffic, especially in the rural/remote areas; it is very challenging to incorporate real-time decision-making during the critical remote health monitoring use-cases. Various mobile devices, user-smartphones, wearable devices, Unmanned Aerial Vehicles (UAVs), ultra-edge sensors, and so forth, can be exploited as D2D nodes or relays [51] in order to augment cellular base stations as represented in Fig. 4.1, which does not require any additional transmission power [52–54]. Therefore, a greater coverage area can be attained to deploy IoT/IoMT devices over remote communities, and a greater number of use-cases can be served (e.g., remote health-monitoring, forest, oil-rigs, energy supply lines, and so forth). In such systems, it is vital to preserve channel quality and minimize the delay and packet drop rate for a more efficient spectrum and throughput while transmitting massive data or medical/health-related data. In the traditional data transmission technique, namely, Decode, and Forward (DF) [55,56], data are encoded and forwarded from a source to relay nodes where decoding and demodulation are performed and re-encoded to pass to the succeeding relay/destination node [57]. Nevertheless, this reception process followed by decoding, encoding, and forwarding at the relay node triggers a considerable delay and high packet drop rate. Another technique called Truncated Decode and Forward (TDF) was proposed by one of the co-authors [1], which utilized multiple bands for concurrent data reception and transmission over a wireless network. The propositioned concept was such that, while receiving data through one of the channels of the node, data will be simultaneously transmitted to the following node through another channel of the node. This chapter justifies the improvement of throughput by implementing a preemptive approach of predicting and then selecting the best channel and band for data transmission in advance in multi-hop relay systems offload-

ing massive IoT traffic. This technique can be adopted for IoT devices for heterogeneous data delivery or IoMT devices for medical data transmission as well.

The prediction of network traffic flows and Channel State Information (CSI) using Artificial Intelligence (AI) have been heavily analyzed throughout the years [58–60]. However, none of these systems have exploited the notion of predicting optimal band/channel in a multi-band relay system, as shown in Fig. 4.1 where each relay node employs various bands (e.g., 5GHz, 2.4GHz, and 920MHz). These stated relay nodes are considered to be resource-constrained and, therefore, are unable to train a complex deep learning model locally. In order to solve this problem of the multi-band, multi-channel prediction task with marginal error, we intend to implement various pre-trained AI models [61–63] such as Linear Regression (LR) [64], Auto Regression (AR) [65], Artificial Neural Network (ANN) [59], and Convolutional Neural Network (CNN) [66] with shallow and deep layers. In the adopted shallow architecture and deep architecture for both the ANN and CNN models, we employed one hidden layer for the shallow construct and multiple hidden layers (four layers) for the deep construct. These pre-trained DL models will be installed locally in the relay nodes to make precise channel quality predictions. From the experimental results, it is found that, among these models, shallow-CNN provided the most promising outcome in predicting the best channel for transmitting data in the resource-constrained relay nodes. After an optimal channel is predicted in the relay node through our distinct deep learning model, the modulation, coding rate, and sending rate are acquired from the Modulation and Coding Scheme (MCS) table, which are used to calculate the link rates. The necessity for these data transfers is determined by first transferring the data frame's header to the relay node followed by transferring the rest of the data based on a respective decision. The data is then further forwarded to the destination node from the relay node through another band. Since data is being transferred simultaneously in this method, the delay is minimized significantly, and thus this method performs better than the conventional approach, where data is transmitted only after the reception of the entire data frame. The model is validated using extensive computer-based simulations and real datasets in order to ensure accuracy and efficiency. The proposed methodology has the potential to accelerate the IoT/IoMT-based medical/health-related data sending process as well as generic data transmission.

The rest of the chapter is constructed as follows. Sec. 4.2 surveys the relevant research work. The proposed multi-band network architecture is described in Sec. 4.3. The following Sec. 4.4 describes the existing problems related to channel selection in multi-band systems. Our proposed input representation and deep learning model are demonstrated in Sec. 4.5. Then, in the Sec. 4.6, the algorithmic analysis of the proposed method is conducted. An illustrative example of how the proposed model produces the output from the input through different stages is manifested in Sec. 4.7. The performance of our proposal is assessed in

Sec. 4.8 and contrasted with those of LR, and AR. Finally, Sec 4.9 concludes the chapter.

4.2 Related Work

In this section, an extensive literature review has been performed by assessing from the standpoint of two domains – implementations of machine/deep learning models to predict diverse parameters related to network condition and constructions of various algorithms to improve spectral efficiency of the overall scheduling process.

4.2.1 Wireless Network Condition Prediction Using AI

An extensive survey conducted by Mao et al. [61] established the significance and possibilities of deep learning in numerous wireless network scenarios. AI techniques have been utilized to predict network traffic efficiently. Predictors from multiple classes, including classic time series, ANN model-based, and wavelet transform-based predictors, have been proved to be viable for predicting network traffic [67]. A deep learning architecture based on the deep belief network is proposed for predicting network traffic in the wireless mesh network [68]. In the wireless network domain, AI-based models have also been exceptionally prevalent for predicting link qualities and links susceptible to failure [69–71]. The work in [72], evaluated the link quality using channel rank measurement along with machine learning algorithms based on the network's Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI). Herath et al. [73] predicted a successive series of signal strength data based on previous time-slots using Recurrent Neural Networks (RNNs) such as Long Short Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. The proposed models in [73] were able to surpass baseline techniques like linear regression and autoregression; however, their work assumed such complex computations to be conducted at the base stations with adequate computational resources. Therefore, the deep learning prediction can be further enhanced for light-weight real-time predictive channel assignment by transferring to resource-constrained relay nodes.

4.2.2 Multi-Band Scheduling Over Relay Networks

In a prior study conducted by one of the coauthors' [1], the spectral efficiency in a multi-band relay transmission scenario was heightened while keeping the end-to-end communication delay minimal. A TDF method was integrated at the relay node instead of the traditional DF method to perform demodulation, de-interleaving, de-puncturing, and Viterbi decoding. To evaluate the channel quality, Signal-to-Noise-Ratio (SNR) was considered for each channel, and an MCS table was used in addition to calculating the transmission duration and the channel with lowest transmission time was selected for data transfer. In an

extension to the work [74], the Signal-to-Interference-plus-Noise-Ratio (SINR) was assumed while selecting the channel in addition to a finite buffer size at the relay node. Moreover, the impact of physical proximity and channel conditions upon the interface of a multi-band system in order to improve the efficiency was stated in [75]. The research works stated above reveals a significant gap in the realms of channel selection and channel assignment in wireless systems. Therefore, it can be established that the prevailing studies have not explicitly explored the problem of multi-band channel prediction in relay networks for proactive assignment.

4.3 Proposed System Model

In this section, we discuss the multi-hop, multi-band network architecture, and transmission model that has been regarded in this chapter.

4.3.1 Network Topology

The relay network topology is represented as a cohesive cellular-D2D system containing a set of N transmitter-receiver (Tx-Rx) pairs. The Tx-Rx pair can contain different configurations as follows: (i) a source node S, such as an IoT device, that transfers its collected data to a mobile User Equipment (UE) (e.g., an energy and performance constrained user-smartphone); (ii) a D2D relay node R that transmits data to the succeeding node; and (iii) A D2D Rx node also called destination node D (such as cellular gateway or base station), that receives data from a D2D relay node. The Tx-Rx pair can communicate over links having L frequency bands with a finite number of channels C. The regarded network topology's routing matrix can be stated as $R = [r_{nlc}] \in \{0,1\}^{\{N \times L \times C\}}$. If the data is routed across band l and channel c between nodes, the element r_{nlc} equals 1; otherwise, it is 0. In order to improve throughput of this whole architecture to perform improved scheduling, the channel quality q_C should be detected precisely. To perform a predictive channel assignment system, the ground truth, i.e., the channel quality in terms of SINR, can be established by the following formulation:

$$SINR = 10log_{10}\frac{P_S}{(P_I + P_N)}, \tag{4.1}$$

where P_S, P_I, and P_N denote the desired signal power, interference signal power, and noise power, respectively. P_S is measured as follows.

$$P_S = P_T(\frac{v}{4\pi dl})^2, \tag{4.2}$$

where P_T, v, d, and l denote the transmission power, the velocity of light and the distance between the Tx-Rx pair and the frequency band of the currently assigned channel,

respectively.

In addition to that, the interference signal power P_I is computed as follows.

$$P_I = \int_{\omega} P_T(\frac{v}{4\pi dl})^2 \; n_I * \rho(x,y)dS, \tag{4.3}$$

where dS indicates an infinitesimal area (i.e., the communication range) in the circles centered both at the Tx and Rx nodes given by ω; $\rho(x,y)$ denotes the probability density function of the interfering nodes in the communication area of the Tx-Rx pair; and n_I represents the total number of nodes distributed around the Tx-Rx pair. For each channel c on each band l, a separate n_I value needs to be considered.

4.3.2 Packet Transmission Model

A TDF model was considered to carry out packet transmission, that employs multiple channels for simultaneous data transfer from source to destination node using (N-2) relay nodes. In order to comprehend the necessity of relay node transmission, the header of the data block is forwarded to the subsequent relay node initially. If a successful header transmission occurs, the rest of the data is also forwarded to the relay node. The link rate L_r of the links between the source and relay (S-R), and relay to destination (R-D), can be calculated as follows:

$$L_r = smr_c, \tag{4.4}$$

where s, m, and r_c denote the symbol rate, symbol density, and coding rate, respectively. s can be calculated as follows.

$$s = \frac{W}{1+B}, \tag{4.5}$$

where W and B represent the roll-off rate and bandwidth, respectively. Next, m is calculated as:

$$m = \log_2 M, \tag{4.6}$$

where M denotes the number of waveforms on which the binary digits are mapped. Additionally, the transmission time of header T_h, and data from source to relay T_{S-R} and relay to destination T_{R-D} for each channel is calculated based on the data size S_d. In other words,

$$T_h = \frac{S_d}{L_r} \tag{4.7}$$

4.4 Problem Statement

This section discusses the traditional optimization-based approaches employed in the literature and the challenges related to optimal channel selection in dynamic network conditions

in the multi-band relay system, in a resource-constrained manner. A communication method comparable to that of [76] recognizes the channel selection problem in multi-band systems as Mixed Integer Non-Linear Programming (MINLP) optimization problem. This communication architecture computes such a problem by relaxing the NP-hard problem into Master Problems (MPs) and forming a column generation-based procedure to determine the MPs in the multi-radio base stations, considering they have adequate computational resources. However, a centralized server requires substantial processing power and memory or a powerful Software Defined Network (SDN) controller in order to solve such an optimization problem. Therefore, to address this issue, the computations should be performed locally, in a light-weight manner at the resource-constrained relay node. Hence, our system model in (Section 4.3) took into account such resource-constrained relay nodes that are unable to solve the optimization problem locally in dynamic network conditions such as varying network traffic, data size, the various distances among the relay nodes, etc. The problem can be defined as utilizing a string of past data for T_w time steps, which can be represented as, $X_T = \{x\}_{T-(T_w-1)}^{T}$, based on measures like SINR, SNR, CSI or RSSI, in order to predict the next few time steps P_w, i.e., $Y_T = \{y\}_{T+1}^{T+P_w}$. Such inferencing task for channel quality prediction is expected to be performed in the resource-constrained relay nodes averting the heavy computations. Thus, the channel assignment time can be considerably reduced along with the improvement of throughput for the whole data transmission architecture as potential packet loss at the relay node can be averted. In the following section, we fathom a sustainable deep learning-based solution to the problem mentioned above.

4.5 Proposed Deep Learning-based Algorithm

This section presents a deep learning-based approach as a solution to the problem stated in Section 4.4. In this approach, the model is first trained in a centralized network with available traffic datasets containing channel conditions and other network signal measures. For various bands, the trained model in intended to correctly predict the channel quality (q_C) in advance. At each relay node, the pre-trained channel inference model is then transported.

The proposed deep learning-based CNN model is depicted in Fig. 4.2. Here, the number of hidden layers used to construct the model is represented by k. Each hidden layer consists of a 1-D (one-dimensional) convolution layer, a regularization layer (i.e., dropout), and a 1-D Max pooling layer. The 1-D convolution layers have an initial filter size of S_F in the first layer and Sigmoid as activation function, Ω [61]. We have used a dropout layer as the regularization method, with a rate of α to avoid overfitting during the training phase [77]. Afterward, the 1-D max pooling layer is added, which reduces the feature size and decreases the computational cost requirement of the model [35]. The channel prediction model utilizes

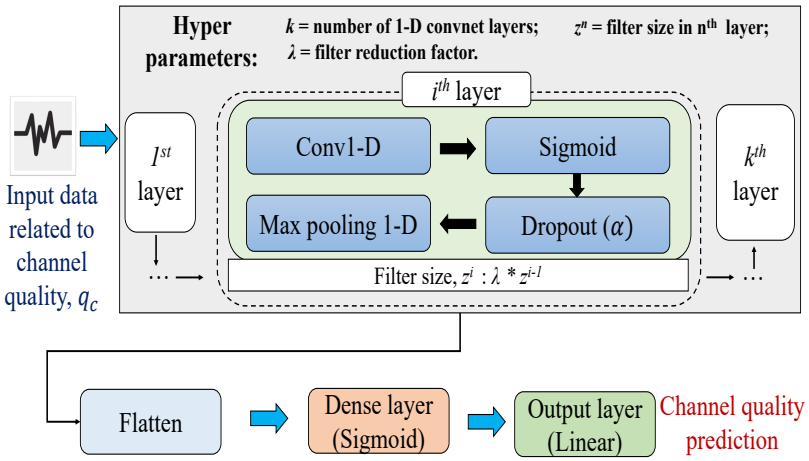


Figure 4.2: Proposed CNN-based training and inference model.

the past signal data in a sliding window of T_w size in order to predict the channel quality of the next P_w steps. The window is slid further P_w steps in order to predict the channel quality of the next time frame. Two strategies are exploited by the model in order to forecast the channel quality, which are: (i) *controlled prediction*, and $ii)$ *smart prediction*. The *controlled prediction* strategy operates by choosing the actual signal strength features as the input of the deep-learning model rather than the newly predicted value. A speedier convergence can be achieved with this strategy; however, due to the lack of accessibility of the previous data during prediction in the test dataset, the model experiences difficulties with generalization. Alternatively, the *smart prediction* strategy offers a prediction of future time steps based on the newly predicted values. Thus, the prediction error can be minimized through this online learning. To measure the performance of the model in the test window, we have employed Root Mean Square Error (RMSE) [41] as an error measurement method to find the difference between the predicted value (V_p) and the actual value (V_a) as calculated below employing Eq. 4.8.

Algorithm 2: Predict Channel Quality (Features).

Input : Various features (f) of signal strength with time
Output: Quality of a particular channel

1 Choose the best values for all the hyperparameters (i.e., k, S_F and Ω), based on experimental results from the considered corresponding values set
2 Scale and pass f through input layer and forward to hidden layers
3 Forward the sum of hidden layers through Ω followed by sub-sampling operation
4 Extract new features in the convolutional layers and forward to fully connected layer
5 Predict q_C in output layer

$$RMSE = \sqrt{\frac{1}{P_w}\sum_{i=1}^{P_w}(V_a - V_p)^2}. \tag{4.8}$$

The deep learning-based CNN model to predict the quality of the channel is described in Algorithm 2. The signal strength parameters such as band-specific channel quality measurements of the channels in the training window T_w (e.g., SINR measured by Eq. 4.1 or others such as RSSI, CSI) along with other features such as energy information, packet loss, throughput, delay are passed to the deep learning-based model. The best value for hyperparameters, such as the number of hidden layers (k), activation function (Ω), and size of the filter (S_F) are chosen after hyperparameter tuning. The features are passed through input layers to the hidden layers where a sequence of convolutions is performed. The final sum of the convolutions is forwarded through activation function and a sub-sampling operation is performed after that. New features are extracted from the layers and the features are forwarded to fully connected layer where the output, i.e., the channel quality (q_C) is predicted for prediction window P_w. The channel quality is then used to perform scheduling using the optimal channel as shown in Algorithm 3. A vector of channel quality $\mathbf{Q_C}$ for all the channels in each band is predicted using the adopted CNN model. The vector elements are then sorted according to their respective channel quality values, and thus, the best channel is selected. Subsequently, the selected channel is used to find the coding rate r_c from MCS table (Table 4.1). The link rate and transmission time of the channels are then computed as per Eq. 4.4 and Eq. 4.7, respectively. The channel where the transmission time of source to relay is larger than that of relay to destination is thus selected in advance. If the condition is not satisfied, the algorithm will look for another channel that fulfills the specified condition.

Algorithm 3: Predictive channel assignment of each relay node.

Output: Optimal channel to be assigned to outgoing link of the relay node.

```
for Each band do
    for All available channels do
        Q_C+ = Invoke algo. 2 to obtain q_C
    end
    Sort values in Q_C of each band from best to worst
    The coding rate of best channel r_c is collected from MCS table (Table 4.1)
    Compute Link rate L_r using Eq. (4.4)
    Calculate T_{S-R}, T_h and T_{R-D} using Eq. (4.7)
end
while T_{S-R} > T_{R-D} + T_h do
    Select min(T_h + T_{R-D} - T_{S-R}) such that (T_h + T_{R-D} - T_{S-R}) > 0
end
```

Table 4.1: Considered Modulation and Coding Scheme (MCS) [1].

SINR	Modulation Scheme	m	r_c
SINR $\leq$ 2	No Tx	0	0
2 <SINR $\leq$ 5	BPSK	1	1/2
5 <SINR $\leq$ 9	QPSK	2	1/2
9 <SINR $\leq$ 11	QPSK	2	3/4
11 <SINR $\leq$ 15	16QAM	4	1/2
15 <SINR $\leq$ 18	16QAM	4	2/3
18 <SINR $\leq$ 20	16QAM	4	3/4
20 <SINR $\leq$ 25	64QAM	6	2/3
25 <SINR	64QAM	6	3/4

4.6 Algorithmic Analysis

In this section, we investigate the algorithm's computational complexity and the time cost to run the proposed deep learning-based channel quality estimator for multiple bands. The analysis essentially centers around the algorithm complexity in the training phase and running phase via calculating the frequency of each operation (e.g., addition, subtraction, multiplication, division, and square root, etc.). The time cost of every procedure is denoted by ADD, SUB, MUL, DIV, and SQRT to precisely express the complexity. We analyze the training and prediction steps' complexity of the proposed model in terms of the number of different operations required by various steps of the algorithm. The outcomes from the complexity analysis are further explored in the performance evaluation section (Sec. VIII) for conducting a numerical analysis of the proposed model to manifest results in terms of processing time, throughput, and memory consumption to demonstrate the proposed model's applicability in the IoT environment.

4.6.1 Pre-processing Phase

Before starting the training phase, we scaled the dataset using the standardization technique to normalize the feature values in a particular range and it also helps in speeding up the calculations in the algorithm as mentioned in the steps 1-2 of Algorithm 2. Each i_{th} feature from input feature f is passed to the training phase after applying the standardization formula denoted by Eq. 4.9,

$$x_i^{'} = \frac{(x_i - \bar{x_i})}{\sigma} \tag{4.9}$$

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x_i})^2}{N_{x_i}}} \tag{4.10}$$

Here, x_i is the feature vector and $\bar{x_i}$, σ (expressed in Eq. 4.10) are the mean and the standard deviation of the x_i, respectively. For each i_{th} feature x_i, if we consider the length

of the feature vector as N_{x_i}, then to compute the mean $\bar{x_i}$ requires $(N_{x_i} - 1)$ ADD and one DIV operations. Then, to calculate σ from Eq. 4.10, it requires $(N_{x_i} - 1)$ ADD, N_{x_i} SUB, one DIV, one MUL, and one SQRT operations. Finally for computing the scaled values employing Eq. 4.9, one DIV and $(N_{x_i} - 1)$ SUB operations are necessary. Since the scaling is performed for the number of features in the input vector denoted as f_n, the overall computation complexity of the pre-processing process for all the features can be expressed in terms of $(2 * f_n * (N_{x_i} - 1))$ ADD, $(f_n * (2N_{x_i} - 1))$ SUB, $(3f_n)$ DIV, (f_n) MUL, and (f_n) SQRT operations.

4.6.2 Training Phase

The main purpose of the training phase is to harness previous data regarding the channels' condition and predict future channel quality for P_w steps. In this phase, the proposed CNN model's computation complexity is analyzed during the learning or training period, referring to the steps 3-5 of Algorithm 2. In the training phase, the model gets trained for T_w time window, and later in the prediction phase, the trained model is used to estimate the channels' conditions. For each i_{th} convolution layer having the number of neurons i_n, filter size S_F and activation function Ω, the computational complexity of i_{th} convolution layer can be expressed as Eq. 4.11,

$$\begin{aligned} O_{conv} = (i_n * (S_F * (f_n - (S_F - 1))) - (f_n * (S_F - 1)) + \\ i_n))ADD, (i_n * (S_F * (f_n - (S_F - 1))))MUL, (i_n)DIV. \end{aligned} \tag{4.11}$$

After the convolutional layers, the model will have fully connected layers. If we assume the number of node in j_{th} fully connected or dense layer to be j_n, the computation complexity will be, $O_{dl} = (j_n * y_i * (f_i - 1))$ ADD, $(j_n * y_i * f_i)$ MUL. Step 5 of the Algorithm 2 will return the predictions of the training phase. Since, according to the steps 1-4 of Algorithm 3 considering the number of bands = B_n, number of channels in each band = C_n and the number of layers in the model to be k can be written as Eq. 4.12,

$$O_{training} = (B_n * C_n * k * (O_{conv} + O_{dl})) \tag{4.12}$$

Hence, for each type of operation, the complexity will be increasing by k (number of layers). Furthermore, for B_n bands and C_n channels, the required number of operations will be multipled by C_n and B_n.

4.6.3 Running Phase

After getting the predicted channel quality from all channels of all the bands, step 5 of Algorithm 3 will sort the predicted channel quality (q_c) for each band. Considering the sorting algorithm to be quick-sort of n number of channel quality of a band, the required operations for the sorting will be $3n$ ADD, $(n-1)$ SUB. In steps 6-7, for all the bands (B_n) the best channel's coding rate will be selected, and the link rate L_r will be calculated. Therefore, these steps will require B_n ADD, B_n MUL and B_n DIV. The following step 8 of the algorithm will determine the values of T_{S-R}, T_{R-D}, and T_h, which will result in $3 * B_n$ DIV. In the last few steps of the algorithm, specifically from step 10-12, $min(T_h + T_{R-D} - T_{S-R})$ will be determined. Considering the loop in these steps will run for m number of times, for B_n bands, it will require (B_n*(2*m ADD, m SUB)) operations to be performed. Therefore, in the running phase of the model, the computation complexity required by the proposed model can be written as Eq. 4.13,

$$\begin{aligned} O_{running} = (3n * B_n + B_n + B_n * 2 * m)ADD + ((B_n * \\ (n-1)) + B_n * m)SUB + (B_n)MUL, (4 * B_n)DIV \end{aligned} \tag{4.13}$$

4.7 An Illustrative Example of the Proposed Model

In this section, we demonstrate a simple example for illustration of how our proposed algorithm operates, and feature size evolves in diverse layers of the model. Fig. 4.3 represents an illustrative example for this purpose. In this example, we have considered a shallow-CNN model consisting of a single layer. We assume 14 features, for instance, to pass through the layers of the proposed model. The number of filters is set to 150, and the kernel size is 3. Therefore, the overall filter size (S_F) is 150x3.

Firstly, the input will be passed to the convolution layer, and the filters will try to find out insights and patterns in the data by performing the sum of element-wise multiplication of the input and filter. We assume striding value of 1 for the filter, and hence after each filter is applied on the dataset, the output size is going to be reduced to 12 from the initial size of 14. Therefore, for 150 filters, the output size is going to be 150x12. Now let us determine how the filters convolve over the input vector and update the values. Assuming the input, f = [-8.41, 5.90, 7.35, 3.13 , -2.41 , -1.60, 5.92, 4.91, 22.7, 0, 15.92, 0, -8.39, -4.97] and the filter 1, τ_1=[0.76, 0.08, 0.60]. The results after the filter are applied on the input vector can be denoted as f_{τ_i}, and the first step of f_{τ_i} is determined as $(0.76 * -8.41 + 0.08 * 5.90 + 0.60 * 7.35) = -1.48$. Subsequently, the calculated f_{τ_i} will be passed to the sigmoid activation function (Ω) and hence, $\Omega(f_{\tau_i})$=0.190. This procedure will be applied at each step using a striding value of 1 on the whole input sequence, which

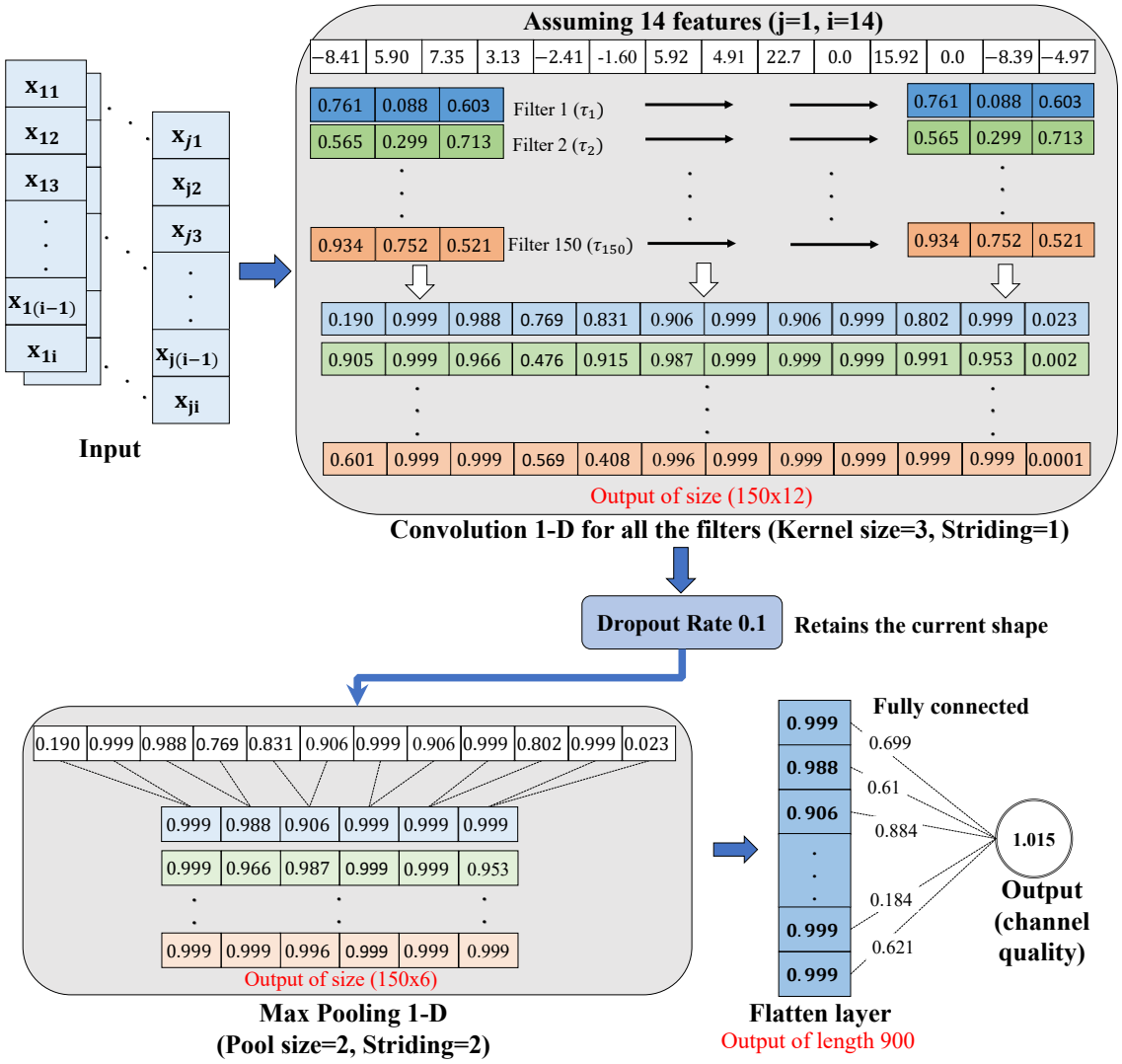


Figure 4.3: An illustration of how the data size evolves in the proposed CNN model with single layer.

will result in an output of size 12 for a single filter. Consequently, for 150 filters, the final output shape in this step will be 150x12.

The dropout phase is a regularization technique of the proposed method will not change the shape of the output as it will only ignore a few nodes in order to avoid overfitting. The following step is the max pooling with a pool size of 2 and striding of 2. Therefore, the maximum value will be selected from the pairwise comparison at each step. The size of the output of this step is 6 for each input from a particular filter. Hence, for all 150 filters, the output size will be 150x6. After the max-pooling layer, at the penultimate stage, the flatten layer will be applied to the feature set, which will reconstruct a multi-dimensional

matrix of features into a vector that can be fed into a fully connected dense layer of the neural network classifier. Therefore, the 150x6 sized output of the max-pooling layer will be converted into a 900 length vector. Lastly, for determining the output scalar, the output vector of the flatten layer will go through a fully connected layer. Summation of element-wise multiplication of the output vector of the flatten layer and the weight vector will be passed through the activation function (Ω) to get the final output value (1.015), which indicates the channel quality of the corresponding band. This procedure is conducted in terms of each band to obtain the predicted value for channel quality.

In this example case, we have considered a shallow-CNN of our proposed model, where we assumed the number of convolution layers to be one to perceive how the input vector transforms for different layers. The deeper architecture of the proposed model will employ the output of the preceding layer as input to the subsequent layer and will contain similar calculations in each of the layers.

4.8 Performance Evaluation

In this section, we demonstrate the experimental results to evaluate the performance of the proposed deep learning-based proactive channel assignment to the multi-band relay nodes. A brief description of the datasets used to feed to our proposed deep learning model is first presented, followed by experimental results and discussion.

4.8.1 Data Preparation

We have employed three datasets, denoted by DS1, DS2, and DS3, respectively, for performing the channel quality prediction. Brief details of the datasets are given below.

1. DS1: This dataset consists of RSSI data obtained with a mobile robot in two environments: indoor and outdoor [78]. RSSI data of five wireless receivers in indoor conditions are collected using the a youBot mobile robot. For the outdoor environment, data for signal strength and location are collected from a mobile robot in a semi-outdoor environment.

2. DS2: This dataset contains signal strength measurement of a Zigbee-based wireless network [79]. It contains around 8000 data samples between a Tx-Rx pair is placed over a distance of 10 to 35 meters, and it consists of information of energy, throughput, delay, and loss of data transfer from source to destination nodes.

3. DS3: This particular dataset describes an extensive set of traces that represent the radio channel conditions between the base station and the end-user device [80]. The

records were employed to design and simulate a mobile networking environment practically. The LTE signal strength values, Reference Signal Received Power (RSRP), SNR, and RSSI. All of these features contribute a comprehensive insight into the channel condition. Each of the annotated traces is of different environments (e.g,. bus, train, pedestrian, static and train) and at different speeds.

4.8.2 Simulation Results and Discussion

We applied Linear Regression (LR), Auto Regression (AR), and a Artificial Neural Network (ANN) [71] to compare the performance of our proposed CNN-based controlled and smart channel prediction methods. The training and prediction window for both deep and shallow architectures of the ANN and CNN models are selected based on tuning of these parameters. The selected tuned values are: training window, $T_w = 200$, prediction window, $P_w = 50\%$ of the training window size (i.e. 100), and the number of hidden layers in the DL models, $k \in \{1, 4\}$. Note that these parameters are selected based on the results of manual hyperparameter tuning without resorting to grid search.

Table 4.2: Comparison of average RMSE values across all time steps for LR, AR, ANN, and CNN-based methods for DS1-indoor environment.

Method	**Controlled Prediction**	**Smart Prediction**
LR	6.933	1.441
AR	25.715	1.386
Shallow-ANN	0.726	0.392
Deep-ANN	0.725	0.414
Shallow-CNN	0.595	**0.371**
Deep-CNN	0.643	0.388

Table 4.3: Comparison of average RMSE values across all time steps for LR, AR, ANN, and CNN-based methods for DS1-outdoor environment.

Method	**Controlled Prediction**	**Smart Prediction**
LR	2.753	34.656
AR	3.198	5.860
Shallow-ANN	1.210	1.308
Deep-ANN	1.144	1.297
Shallow-CNN	1.057	**0.964**
Deep-CNN	0.971	0.994

Tables 4.2 and 4.3 list the performance of ANN, and CNN compared to the LR and AR technique, which is obtained from DS1 (indoor and outdoor conditions). The results are represented in terms of the average RMSE across all the channels. We evaluated the performance of the deep learning models in both controlled and smart prediction scenarios by employing a shallow architecture (a single hidden layer), and a deep architecture (four hidden layers). In all the cases, the deep learning-based prediction performance exhibits

Table 4.4: Comparison of proposed shallow and deep-CNN models with baseline techniques for different prediction window (P_w) sizes with respect to training window (T_w) using DS2. Here, D indicates distances in meters, S-CNN and D-CNN represents shallow and deep CNN, respectively.

D (m)	**RMSE of different methods for Size of (P_w) with respect to (T_w)**											
	25%				**50%**				**75%**			
	LR	AR	S-CNN	D-CNN	LR	AR	S-CNN	D-CNN	LR	AR	S-CNN	D-CNN
10	2.35	58.97	1.22	0.99	1.76	61.30	0.78	0.71	1.91	2.83	0.86	1.68
15	2.13	79.63	0.82	0.77	2.35	63.79	0.90	0.64	2.02	52.23	0.61	1.29
20	2.63	20.95	1.85	1.00	1.98	56.21	0.72	0.66	1.74	2.28	0.75	1.49
25	2.63	56.37	0.45	0.80	2.56	51.61	0.46	0.55	2.16	59.91	0.72	0.75
30	2.95	73.08	0.61	1.05	1.99	75.81	0.57	0.55	2.33	30.24	0.69	0.92
35	3.16	65.45	1.93	0.99	2.21	56.24	0.86	0.81	2.73	7.93	0.76	1.69
Mean RMSE	2.64	59.08	1.15	0.93	2.14	60.82	**0.71**	**0.65**	2.14	25.90	0.73	1.30

better performance in contrast with that of AR and LR. Between controlled and smart prediction schemes, the model demonstrated better estimation performance in terms of the smart prediction than its controlled prediction counterpart. This implies that the deep learning-based techniques were able to explore the search space more robustly in contrast with the baseline techniques. Therefore, they were able to generalize the diverse channel conditions, especially in the smart prediction strategy. Hence, we adopted the smart prediction scheme for the next experimental setups to further optimize the hyperparameters and evaluate the results. In the case of the outdoor environment, the proposed CNN-based method can outperform the baseline approaches by a significant margin. This massive performance gap between CNN and baseline techniques demonstrates that in noisy outdoor environments, the traditional algorithms are unable to predict channel conditions accurately compared to deep-learning-based techniques. Thus, it may be concluded that, although the baseline techniques perform well in a single-hop, single-radio wireless network for predicting channel conditions [73], are not appropriate for real-time channel prediction in multi-band relay networks. Furthermore, the performance comparison between ANN, and CNN reveals that CNN is more consistent and stable than the other neural networks. Hence, we have not considered ANN for further experiments, and elected one and four-layer architectures of CNN, referred to as shallow and deep-CNN, respectively, for further analysis.

4.8.2.1 Hyperparameter Tuning

To choose the best hyperparameters of the proposed CNN-based models, we have conducted a systematic investigation. Firstly, as a part of determining the best prediction window-size, we tuned the prediction window-size, P_w, by adopting varying ratios with respect to

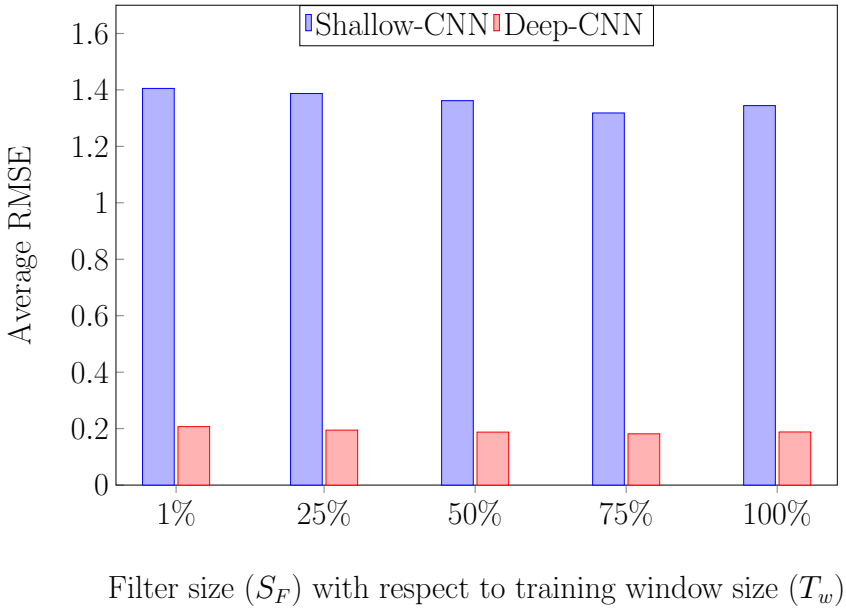


Figure 4.4: Comparison of CNN with respect to filter size (for DS1-indoor environment).

the training window T_w. Table. 4.4 illustrates the RMSE values for various distances from dataset DS2 for P_w values varied among 25%, 50%, and 75% of T_w, respectively. In all these cases, the baseline techniques are significantly outperformed by the proposed CNN-based approach. As the distance between the sender and the receiver nodes increased from 10 to 35 meters, the estimation error (i.e., RMSE) of LR showed a dramatically progressing trend. Therefore, this implies that the baseline techniques are not able to interpret channel quality in diverse situations with a growing distance as much as the proposed deep-learning-based technique. Hence, the CNN-based approach can perform the prediction task with much less error compared to that incurred in AR and LR. Among these three proportions, both the shallow and deep-CNN models manifest the best average RMSE values of 0.71 and 0.65, respectively, when 50% of the training window T_w is taken into consideration for the prediction window size (P_w). Initially, the training window size (T_w) is set to 200 time steps. Hence, after the hyperparameter tuning, the selected prediction window size (P_w) is 100 time steps. Among the baseline techniques, particularly, the AR suffers drastically for all the considered distances. Therefore, we considered P_w to be half (50%) of the training window T_w to accurately predict the most suitable bands and channels in order to transmit IoT data at a greater throughput in the multi-hop relay system. The trade-off in choosing the training and prediction window is that the training window should not be too large to reduce the time-delay to train the model. The prediction window size should not be more than 50% of the training window size to decrease prediction error, and it should not be too small in size as it will slow down the prediction process.

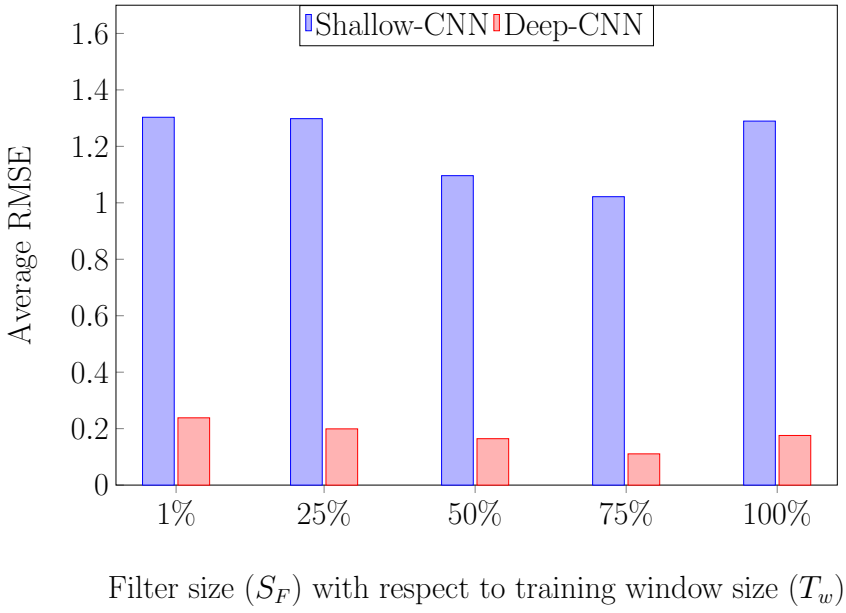


Figure 4.5: Comparison of CNN with respect to filter size (for DS1-outdoor environment).

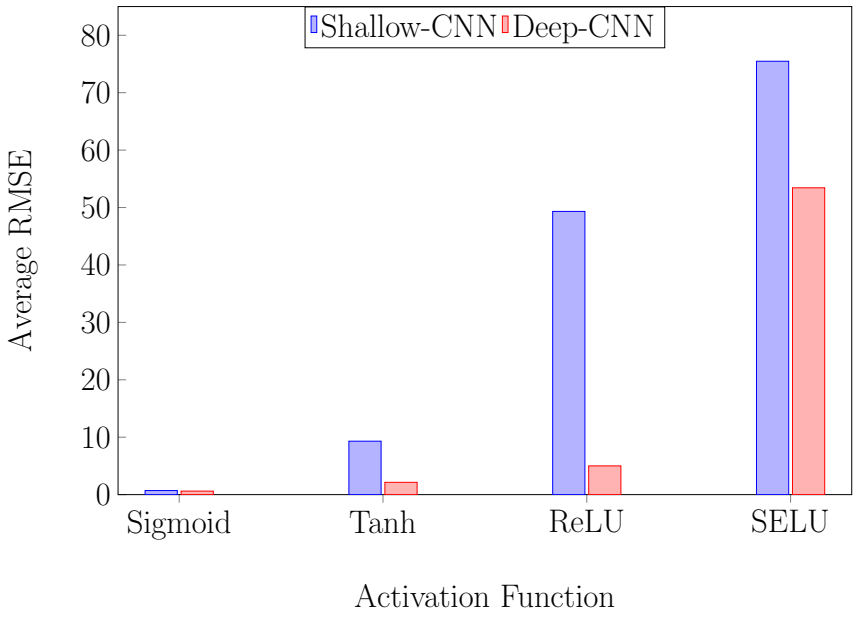


Figure 4.6: Comparison of different activation functions for the proposed CNN model using DS2.

Next, as a part of the hyperparameter tuning, we tuned the filter size (S_F) of the 1-D convolutional layers with respect to the training window (T_w) employing DS1. Figs. 4.4 and 4.5 demonstrate the outcomes in terms of average RMSE for different size of filter. In both cases, the least error is recorded when the initial filter size (S_F) is equal to 75% of the training window (T_w). Therefore, we elected this parameter to be the starting filter

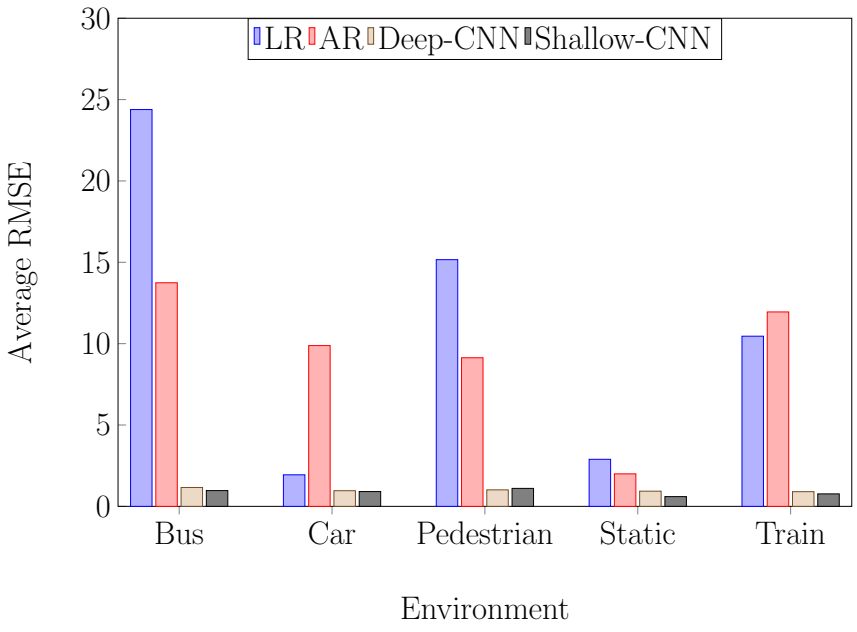
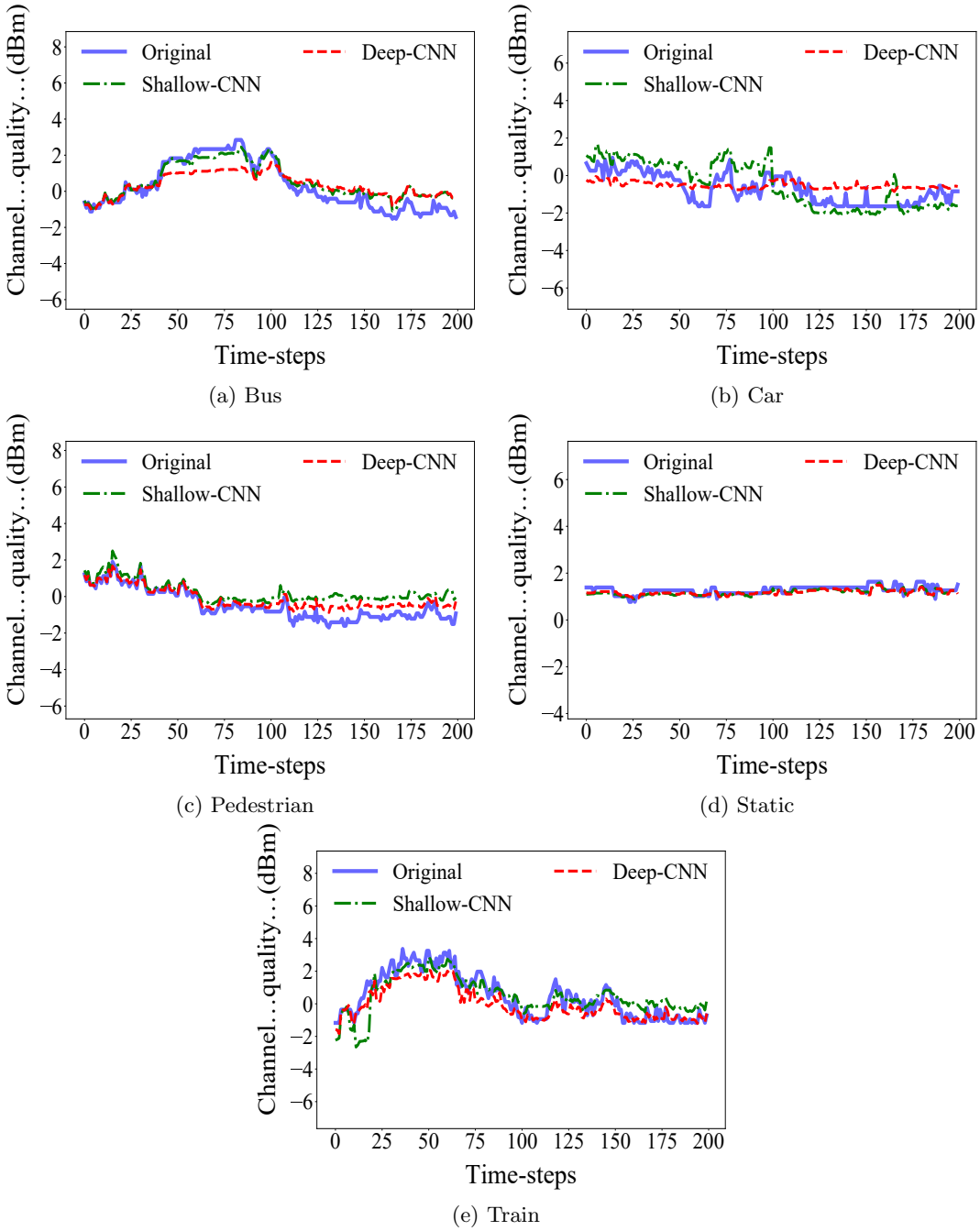


Figure 4.7: Comparison between CNN and baseline techniques for different environments of DS3.

size of our proposed model. The filter size reduction factor, α, is set to 0.5 as we have decreased filter size by 50% each time for the deeper layers compared to the previous layer. Furthermore, to pick the most suitable activation function (Ω) for the proposed deep learning-based CNN model, we have performed manual hyperparameter tuning from a set of proper activation functions (i.e., Sigmoid, Tanh, ReLU, and SELU) [81]. Fig. 4.6 portrays the results for the different activation functions in terms of average RMSE noted using DS2. The best performance is evident when the Sigmoid activation function is used in all the layers of the proposed shallow and deep architecture of CNN. Although the sigmoid activation function suffers from vanishing gradient problem in some cases, however, in this experiment, we observe that the utilization of sigmoid function provides robust performance. Apart from the prediction efficiency, the sigmoid activation function is generally less computationally expensive compared to many other activation functions, which is a striking advantage of adopting this function in the proposed AI model. Therefore, this verifies the selection of Sigmoid activation function for the channel quality prediction task in the multi-band relay communication system.

After the hyperparameter tuning experimental phase, we evaluated the model with another distinct dataset (DS3). Fig. 4.7 depicts the comparison of error (average RMSE) between deep-CNN, shallow-CNN, AR, and LR in varying environments of DS3. The figure demonstrates that shallow-CNN outperformed deep-CNN by a slight margin and the baseline ML techniques by a considerable margin. Therefore, this result verifies the pro-

posed CNN-based model's acceptability for being selected as a channel quality estimator for real-time channel prediction in multi-band relay networks for efficient data delivery.



(a) Bus

(b) Car

(c) Pedestrian

(d) Static

(e) Train

Figure 4.8: CNN-based prediction methods compared to the original channel quality for different environments of DS3.

We also plotted the predicted channel conditions over time and compared with the original values for visual comparison. Fig. 4.8 manifests a fragment of a few example cases

of the actual and predicted q_C values for varying conditions from DS3 employing smart predictive strategy. The experimental outcomes exhibit that the proposed CNN model's performance is very much indistinguishable to the original signal for most time steps. In the case of static and pedestrian, the overall prediction performance of the CNN models is more robust than other situations. In terms of the other environmental conditions, also the models' prediction is representative of the actual channel quality and is able to find the trend of channel quality.

4.8.2.2 Numerical Analysis

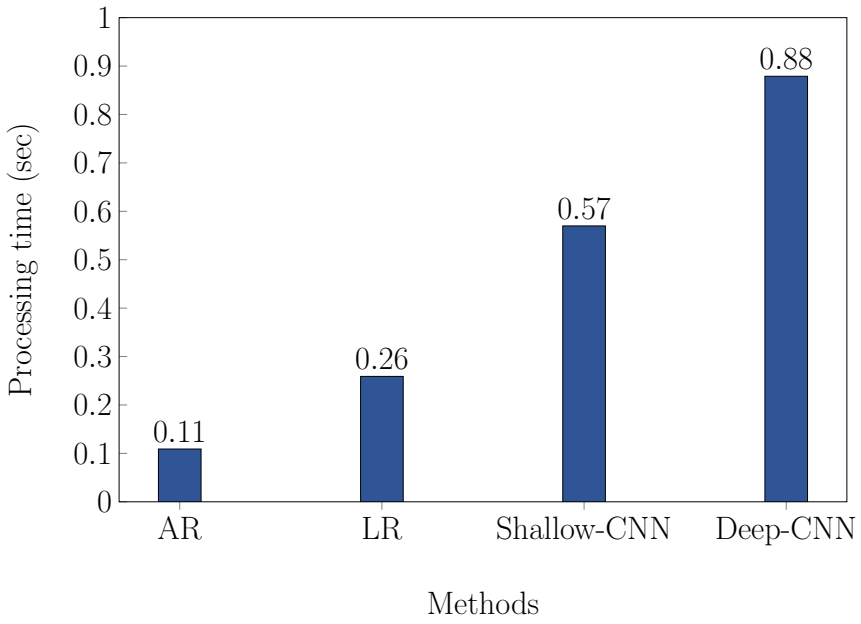


Figure 4.9: Processing time of different methods.

As the proposed CNN-based method's prediction performance manifested encouraging results, we conducted a numerical analysis of the model in terms of memory consumption, processing time delay, and throughput. Fig. 4.9 displays the processing delay for the model, while Fig. 4.10 illustrates the model's memory consumption at each time step with respect to the node's overall capability. In this case, the trade-off is: although the proposed DL-based approach's predictive performance is significantly more prominent than that of the baseline techniques, it consumes more memory and requires a higher time-dealy. However, the additional processing delay can be considered negligible by considering the prediction efficiency of the proposed technique. Admittedly, the proposed model will predict all the channels' conditions over many time-steps ahead of time, considerably minimizing the overall communication delay. Consequently, the acquired experimental outcomes illustrate that the proposed DL-based CNN model is suitable for efficient channel prediction and proactive

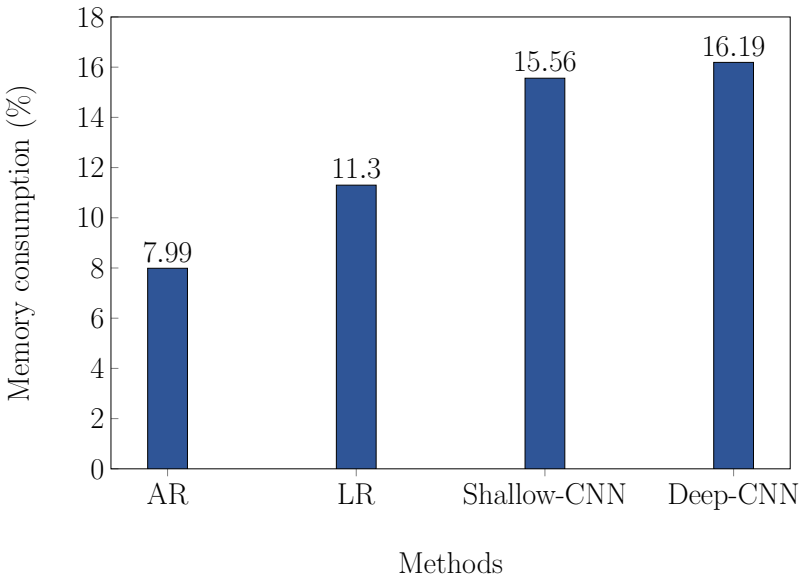


Figure 4.10: Memory consumption of different methods.

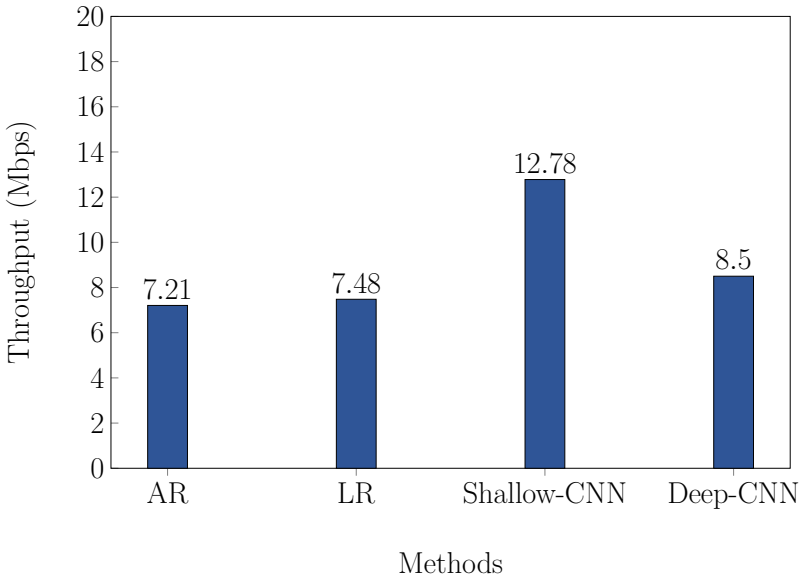


Figure 4.11: Throughput of different methods considering the additional processing delay due to possible poor channel selection.

channel distribution in a multi-band relay network system. Next, we estimate the throughput analysis to ensure how much the processing delay and performance trade-off influences the throughput during IoT data delivery.

Fig. 4.11 depicts the throughput comparison for transmitting data over the heterogeneous band relay-based network. We have considered an average data generation rate of 1MB/s from the nodes. A latency of 1 second is considered because of poor channel selec-

tion from the baseline techniques (i.e., LR, AR) as the predictive performances for these are worse than the proposed CNN model. We estimated the CNN-based model's latency considering the difference of error for poor channel selection between CNN and the baseline techniques (in percentage). Afterward, we determined the throughput by considering the processing delay and the possible delay in choosing a lousy quality channel. Note that, according to the average results observed in Fig. 4.7, the shallow CNN performed the best in diverse environments, indicating that the shallow-CNN would incur the slightest delay due to poor channel selection. On the other hand, even though AR and LR's processing time is less than that of the proposed DL-based CNN model, the throughput is approximated to be considerably lower. Hence, it unveils that traditional baseline techniques cannot be as effective as the proposed CNN in dynamic network conditions in multi-band relay networks, and possible packet loss may occur for specific relay nodes where the resources are inadequate.

The experimental results clearly demonstrate that the proposed shallow-CNN model emerges as the most viable light-weight, predictive channel inference model to deploy at the resource-constrained relay node for enhanced spectral efficiency and throughput for offloading IoT traffic. The proposed DL-based model's generalization capability in terms of prediction performance was evaluated on three different datasets from diverse setup. Furthermore, the prediction performance, as well as memory consumption, processing time, and throughput, were compared with popular AI-based prediction models. The encouraging outcomes illustrate that the proposed model performed efficiently in all three datasets. Hence, the model can be generalized and utilized for other varieties of wireless network setups along with the IoT environment.

4.9 Summary

This chapter focuses on the prevalent resource-constrained multi-band relay nodes for offloading massive IoT traffic in next-generation networks and the measures taken to overcome the most appropriate channel selection barriers for efficient data transmission. The transmission process involves sending the data header first, followed by the rest of the data on one band, and simultaneously forwarding it to the next node through another band. In order to select a channel, a lightweight deep-learning technique is proposed that will accurately determine the best channel to transmit and receive data based on its quality. To construct the Convolutional Neural Network (CNN) model, original datasets were used to forecast the channel quality for both smart and controlled prediction strategies. Afterward, the channel quality measures are utilized by a scheduling algorithm in order to determine the coding and modulation rates to forward the data to the next node. The model was assessed by comparing it with other predictive algorithms such as LR, AR, and ANN. The

performance of these algorithms at the relay node was comparatively inferior to CNN based channel prediction algorithm with respect to throughput, memory consumption, and processing delay. The results of deep-CNN and shallow-CNN were analogous; however, the processing delay and power consumption of shallow-CNN were relatively lower, thus improving throughput. Hence, the propositioned shallow-CNN model was nominated as the most feasible architecture to predict the channel state of a resource-constrained relay node.

Chapter 5

A Proof-of-Concept of Ultra-Edge Smart IoT Sensor: A Continuous and Lightweight Arrhythmia Monitoring Approach

Due to the proliferation of the Internet of Things (IoT), the IoT devices are becoming utilized at the edge network at a much higher rate. Conventionally, the IoT devices lack the computation resources required for carrying out ultra-edge analytics. In this chapter, we go beyond the typical edge analytics paradigm, which is mostly limited to user-smartphones, and investigate how to embed intelligence into the ultra-edge IoT sensors. To conceptualize the smart IoT sensors with enhanced intelligence, we select the arrhythmia detection task employing Electrocardiogram (ECG) trace as one of the mobile health (mHealth) cases. The existing approaches are not feasible for ultra-edge IoT sensors due to the extensive noise-filtering and manual feature extraction phase. Hence, in this chapter, to facilitate the analytics, we propose a Deep Learning-based Lightweight Arrhythmia Classification (DL-LAC) method, which employs only single-lead ECG trace and does not require noise-filtering and manual feature extraction steps. As the proposed technique, we design a one-dimensional Convolutional Neural Network (CNN) architecture. Complying with the ANSI/AAMI EC57:1998 standard, four heartbeat types are taken into consideration as class labels. The efficiency and the generalization ability of the proposed model are evaluated, employing four different datasets from PhysioNet. The experimental results demonstrate that the proposed DL method outperforms traditional methods such as the Delay Differential Equation (DDE)-based optimization, K-Nearest Neighbor (KNN), and Random Forest (RF). The proposed DL-LAC illustrates encouraging performance in terms of time and memory requirement when the trained model is transferred to virtualized microcontrollers

connected to IoT sensors.

5.1 Introduction

The escalation of Artificial Intelligence (AI), Internet of Things (IoT) sensors, and numerous wearable devices have radically enhanced mobile health (mHealth). However, due to the hurdle of incorporating intelligence into these resource-constrained IoT devices, the IoT sensors continue to be routine monitors. The conventional technique is to employ the IoT sensors and wearables to sense user's day to day health data such as Electrocardiogram (ECG), Electroencephalogram (EEG), temperature, respiration patterns, diabetes level, sleep patterns, weight change, and so forth. These health data accumulated by the regular IoT devices are dispatched to a remote cloud for medical analytics, as portrayed in Fig. 5.1. Although this IoT and cloud-based medical analytics serve the purpose of health monitoring, it still raises a few major concerns that cloud-based architecture cannot avoid easily. This paradigm of ECG data analytics results in bandwidth consumption, delay due to transmitting the enormous amount of health data, and privacy concerns associated with the user's health data.

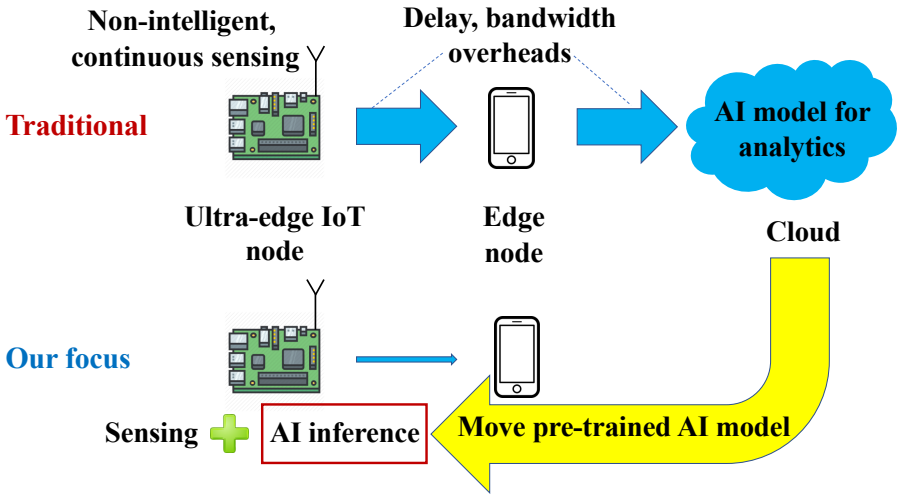


Figure 5.1: How migrate the pre-trained AI model towards the resource-constrained sensor.

Our goal in this chapter is to analyze how to exploit the logic-in-sensor concept, recently introduced by the coauthors' earlier research work [11]. The logic-in-sensor architecture, which is based on Magnetic Tunnel Junction (MTJ)-based spintronic technology, can revolutionize the mHealth industry by enhancing the Quality of Service (QoS) such as communication delay, network bandwidth consumption and privacy of user's health data. Considering the user-smartphone as an edge device that is capable of some analytics, the proposed ultra-edge architecture shown in Fig.5.1 aims to bring the intelligence or the analytics from the cloud to the edge device using the logic-in-sensor concept. Following the hardware enhancement and AI-based intrinsic noise processing, as demonstrated in [11], in this chapter, we intend to obtain a lightweight solution to relocate the cloud-based medical analytics to the ultra-edge smart IoT nodes, and hence, overcoming the issues as mentioned earlier.

We have chosen an essential use-case of cardiac arrhythmia, one of the major causes of Cardiovascular Diseases (CVDs) [82]. Cardiovascular diseases are the leading cause of death worldwide, which results in approximately 31% of all global deaths; however, the risk can be eliminated if detected and diagnosed with timely treatment [83]. Arrhythmias cause the heart not to pump blood in the body adequately, and the patients usually experience symptoms of faster or slower heart pulsations. Conventional clinically graded 12-lead ECG or consumer-grade wearables can be employed to monitor the heart activity of a person. The electrical activity of the heart is known as the ECG waveform, which is a crucial diagnostic tool used to monitor the conditions of the heart and can be used to identify arrhythmias [84]. Automatic detection of irregular heartbeats from ECG signals is a significant task for the smart diagnosis of CVDs, and it is becoming a prominent area where AI can be employed

extensively to automate the process.

Recent advances in AI and the availability of more health data, the utilization of the deep neural network has proven to be indispensable for automating the smart healthcare system [84]. ECG data analytics using Machine Learning (ML) or AI techniques and analyzing time series ECG with nonlinear Delay Differential Equations (DDEs) are explored broadly by traditional cloud-based medical analytics. However, the adaptation of localized embed intelligence at the ultra-edge devices is still not extensively studied in the literature. For diminishing the communication delay and network bandwidth with the cloud and preserve user-data privacy by considering the localized analysis of the health data, a more effective and lightweight analytics technique on-sensor is critical. Therefore, in this chapter, we considered several ML techniques to pave the way to move the arrhythmia analytics from the centralized cloud paradigm to ultra-edge smart IoT. Among different AI approaches, we propose a Deep Learning-based Lightweight Arrhythmia Classification (DL-LAC) algorithm employing the one-dimensional Convolutional Neural Network (CNN) that emerges as the most viable solution for ultra-edge ECG analytics.

The proposed CNN-based model is trained at a central node and then can be transferred to the logic-in-sensor simulation for inference. The proposed model can be used to classify heartbeats employing raw single-lead, and it does not require any noise-filtering of the ECG signal, which makes the system lightweight and easy to integrate with the ultra-edge node. In this vein, the proposed deep learning-based CNN employs the recommendation of Association for the Advancement of Medical Instrumentation (AAMI) for the arrhythmia classification task. We have considered four classes of heartbeats, namely N, S, V, and F, in this chapter, which represents normal, supraventricular ectopic, ventricular ectopic, and fusion beats, respectively [85]. To evaluate the model's generalization ability, we experimented using four clinically graded ECG datasets and considered different experimental settings to test the model's performance using accuracy, precision, and f-score as performance metrics. Lastly, due to the high fabrication cost of a single logic-in-sensor (approaching \$15k for the entire circuit and a further \$10k for further customization), we illustrate the viability of the proposed method's feasibility as a lightweight solution in an emulated ECG sensor with a Raspberry Pi and a few other IoT devices.

The remainder of the chapter is constructed as follows. Sec. 5.2 surveys the relevant research work. The problem of traditional cloud-based analytics and the necessity of lightweight analytics at the smart logic-in-sensor is discussed in Sec. 5.3. The data preparation is outlined in Sec. 5.4. Our proposed input representation and deep learning model are manifested in Sec. 5.5. The performance of our proposal is assessed in Sec. 5.6 and contrasted with those of K-Nearest-Neighbour (KNN), Support Vector Machine (SVM) and Random Forest (RF). Finally, Sec 5.7 concludes the chapter.

5.2 Related Work

Due to the availability of IoT devices that can deliver health data, researchers are have been working on ECG classification [86]. As an indispensable strategy for diagnosing heart diseases, ECG monitoring is comprehensively studied and analyzed. It is vital to detect cardiovascular diseases timely, and for that purpose, continuous observation of ECG for a prolonged period is essential. However, the conventional method of long-time ECG monitoring is invasive and expensive, and it hinders the daily activity of the patients. To overcome this issue and introduce some level of automation in the ECG monitoring system, cloud-based ECG analytics can be employed where the ECG signal is usually transmitted using wireless transmission techniques such as Bluetooth, Zigbee, or Wi-Fi [87–89]. Therefore, most of these traditional automated ECG monitoring systems analyze the data at the cloud and then send feedback back to the user or care-providers. One of the proposed cloud-based analytics where the ECG data are collected using a wearable monitoring node and are transmitted straight to the IoT cloud using Wi-Fi [90]. An IoT-based patient monitoring system is proposed where data is then processed using a Raspberry Pi, and useful information is delivered to the IoT cloud for cloud-based analytics [91].

In this proposed system [92], AdaBoost and Gradient Boosting algorithm were applied to classify ECG using single-lead ECG. An automatic and fast ECG arrhythmia classifier based on a brain-inspired ML approach known as Echo State Networks (ESN) was implemented in for faster ECG analytics [93]. In another work, an accurate arrhythmia classification method for ECG was proposed based on extreme weighted gradient boosting (XGBoost) using a broad range of feature set [94]. In [95], to tackle the patients' privacy concerns, Baza *et al.* have proposed a mimic learning-based machine learning approach for automatic, secure, and efficient analysis of Cardiovascular activities. A clustering-based feature extraction algorithm followed by employing a number of well-known ML classifiers for accurate recognition and classification of arrhythmias is proposed in [96]. Researchers have also employed mathematical methods to decompose ECG, such as a nonlinear DDE was utilized to classify ECG by differentiating features for various heart diseases [97].

Apart from traditional ML techniques, researchers have also employed neural networks and deep learning-based approaches for the classification of ECG heartbeats. In one of the research works, the convolutional neural network of 34-layer was adopted to classify with high accuracy that transcends the cardiologist performance [98]. Principal Component Analysis (PCA) based feature extraction followed by a Multi-Layer Perceptron (MLP) was utilized in another research [99]. Deep-learning-based, Long Short-Term Memory (LSTM) algorithm was proposed in [100], having considerable low computational costs. Recurrent Neural Networks (RNN) was used for binary classification (normal and abnormal) of heartbeat in this research [101]. A Deep Genetic Ensemble of Classifiers (DGEC) was proposed

by combining deep learning algorithms with an ensemble learning and genetic optimization of parameters for the classification of various types of arrhythmias [102]. In our recent work [35], these issues were raised and an attempt was made to embed AI at the IoT sensor level to perform ECG prediction at the ultra-edge network. However, the work concluded the need for a systematic investigation and computational analysis to conceptualize a fusion of logic and sensing to render a continuous and lightweight arrhythmia monitoring system.

5.3 Problem Formulation

As manifested in the previous section, the healthcare sector still needs accelerating improvement in establishing smart healthcare with embedded intelligent sensors. As our research focus in this chapter is lightweight arrhythmia monitoring, we will discuss the drawbacks of the existing ECG/arrhythmia monitoring system and the hurdles associated with transferring the existing analytics to ultra-edge IoT. Traditionally, researchers have employed diverse heartbeat classification techniques that generally require a number of pre-processing steps such as noise filtering, manual feature extraction, and so forth. The steps needed by the conventional heartbeat classification employing ML methods are exhibited in Fig. 5.2. Diverse methods such as DWT, DDEs [103], and ML techniques are commonly utilized in the conventional feature extraction and classification tasks. Though these ECG analytics techniques overcome many drawbacks of the manual ECG monitoring, it still lacks the potential to be integrated with logic-in-sensors due to the extensive computational steps. These conventional ECG monitoring approaches mostly rely on multi-lead ECG signal and requires multiple preparatory steps (i.e., noise filtering), which is a significant issue for combining these models with the ultra-edge IoT logic-in-sensors [35, 104].

Apart from ML techniques, traditionally DDE-based optimization techniques have also been proposed for the ECG monitoring task. However, the non-linear DDE for the time-series ECG analysis technique cannot adequately infer the system models in varying heart conditions. In this approach, exhaustive search or heuristics must be developed to select the most competent model for any given classification task, which is a considerable challenge for lightweight ECG analytics. Conventionally a non-linear DDE can be expressed as follows:

$$\begin{aligned} f(a_i, x_{\tau_j}) = {} & a_1 x_{\tau_1} + a_2 x_{\tau_2} + a_3 x_{\tau_3} + ... + a_{i-1} x_{\tau_n} \\ & + a_i x_{\tau_1} x_{\tau_1} + a_{i+1} x_{\tau_1} x_{\tau_2} + a_{i+2} x_{\tau_1} x_{\tau_3} + ... + a_{j-1} x_{\tau_n}^2 + a_j x_{\tau_1}^3 + a_{j+1} x_{\tau_1} 2 x_{\tau_2} + ... \\ & \vdots \\ & ... + a_1 x_{\tau_n}^m, \end{aligned} \tag{5.1}$$

Here, x_{τ_j} can be expressed as: $x_{\tau_j} = x(t - \tau_j)$, in Eq. 5.1, n, t, m, and τ_j represents the number of delays, time, the degree of non-linearity, and time delays, respectively. The

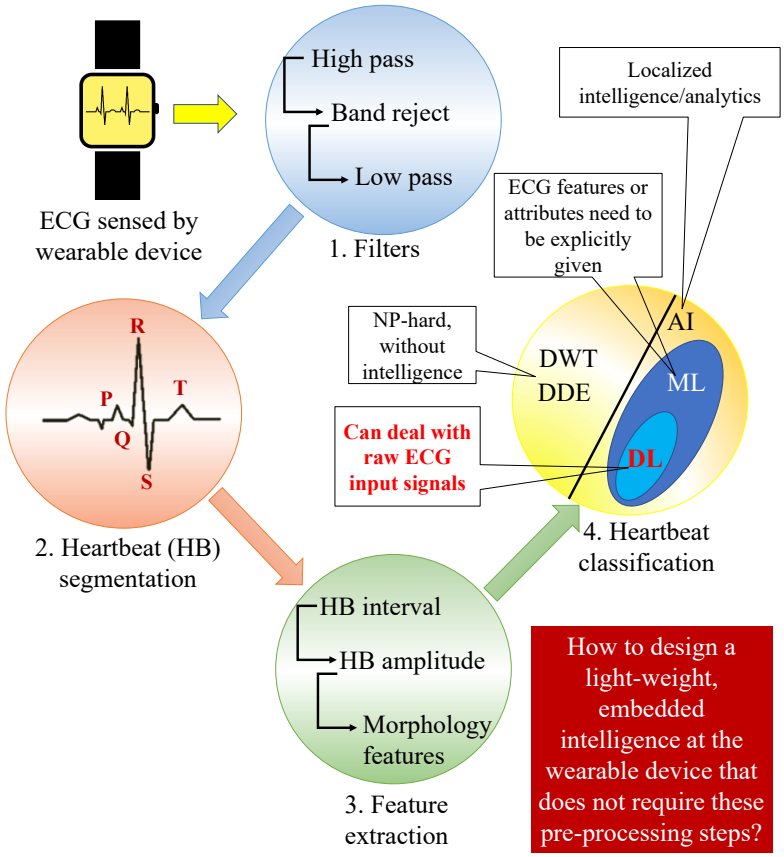


Figure 5.2: Steps of conventional ECG heartbeat classification.

selection of optimal time-delays and monomials is imperative for building an effective DDE-based classification system. For example, to select the optimal model for classification using the DDE-based model, the authors applied the genetic algorithm in [105]. Therefore, these approaches are not appropriate for integrating with the logic-in-sensors for ultra-edge IoT analytics in polynomial time.

Apart from expensive computational requirements, some of the other issues with the traditional ECG monitoring system are that it requires internet connectivity to communicate with the cloud servers for ECG analytics. Hence, it consumes considerable network bandwidth if the number of users is high. Furthermore, due to continuous data transmission, cloud-based analytics can also raise significant privacy concerns for the user's private data. Therefore, this approach can be a hindrance to secure ECG analytics for arrhythmia detection. To address this challenge, we focus on developing an automated, efficient, and lightweight system with localized intelligence that can be deployed and integrated with the logic-in-sensors for ultra-edge IoT analytics. To develop a lightweight ECG/arrhythmia monitoring system, we envision an AI-aided technique for classifying heartbeats employ-

ing a raw single-lead ECG signal and compared the proposed model with traditional ML techniques adopting the architecture depicted in Fig 5.2.

5.4 Data Preparation

We have conducted ECG signal analysis to detect arrhythmia by utilizing the MIT-BIH Supraventricular Arrhythmia Database (DS1) [106], MIT-BIH Arrhythmia Database (DS2) [107], St Petersburg INCART 12-lead Arrhythmia Database (DS3), and Sudden Cardiac Death Holter Database (DS4) [108] from PhysioNet [43]. The datasets contain recordings of many traditional and life-threatening arrhythmias along with cases of normal heartbeat rhythm. Various researchers have employed these datasets for diverse ECG based research [109] [110].

The datasets comprise a text header file, a binary file, and a binary annotation file with .txt, .dat, and .atr extensions, respectively.

- Header file (.hea): This file contains a brief text file that explains the signals' contents, such as the name of the record's file, number of examples, type and format of the ECG signal, and so forth.
- Binary file (.dat): The binary files include digitized representations of the ECG signals of each record.
- Annotation files (.atr): The annotation files contain heartbeat labels that define the type of ECG signals at a particular time in the ECG record.

We generated four separate heartbeat categories following the Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard from the annotation files in each of the datasets. The summary of mappings between the heartbeat annotations for each class is demonstrated in Table 5.1. We have employed the DS1 (MIT-BIH Supraventricular Arrhythmia Database) for the hyperparameter tuning and the training phase. In the running/inference stage, we test the model using the other three datasets (i.e., DS2, DS3, and DS4). We exploited multiple datasets to evaluate the generalization ability of the proposed model. Although each of the datasets contains multiple ECG lead's data, we have employed the lead II in our experiment as our model only requires single-lead-ECG tracing. The distribution of four heartbeat labels is manifested in the Table 5.2.

Table 5.1: Mapping DS1, DS2, DS3, and DS4 datasets to the AAMI heartbeat classes [2].

Heartbeat Class	**Heartbeat Annotation**
N (Normal)	N (Normal)
	L (Left bundle branch block beat)
	R (Right bundle branch block beat)
	e (Atrial escape beat)
	j (Nodal (junctional) escape beat)
S (Supraventricular ectopic beat)	A (Atrial premature beat)
	a (Aberrated atrial premature beat)
	J (Nodal (junctional) premature beat)
	S (Supraventricular premature beat)
	V (Premature ventricular contraction)
V (Ventricular ectopic beat)	E (Ventricular escape beat)
F (Fusion beat)	F (Fusion of ventricular & normal beats)

Table 5.2: Frequency of heartbeats of each class in DS1, DS2, DS3, and DS4.

Heartbeat Class	**DS1**	**DS2**	**DS3**	**DS4**
N	1,62,323	90,621	1,53,672	7,45,671
S	12,197	2,781	1,960	1,893
V	9,941	7,236	20,012	23,616
F	23	803	219	309

5.5 Proposed Methodology

5.5.1 Proposed CNN Model Structure

In this section, we illustrate the proposed lightweight heartbeat classification technique for arrhythmia detection that can be deployed and integrated with AI-aided logic-in-sensor. A lightweight model for classification is an essential part of integrating the AI-aided model at the ultra-edge IoT sensors for faster analysis. Hence, we primarily focused on designing the deep learning-based model that only requires a single lead raw ECG signal so that the model can be sufficiently lightweight. Sensors with embedded intelligence can be utilized for long-term, accurate monitoring of a person's cardiac activity, which is demonstrated in one of the coauthors' previous works [11]. Keeping the concept of logic-in-sensor in focus, we developed a deep-learning-based lightweight model that can be integrated with these AI-aided sensors for analysis of ECG at the ultra-edge device. The acquired results of the ECG analytics can then be sent from the IoT nodes to the care-providers.

We propose an automated deep learning-based one dimensional (1-D) CNN that does

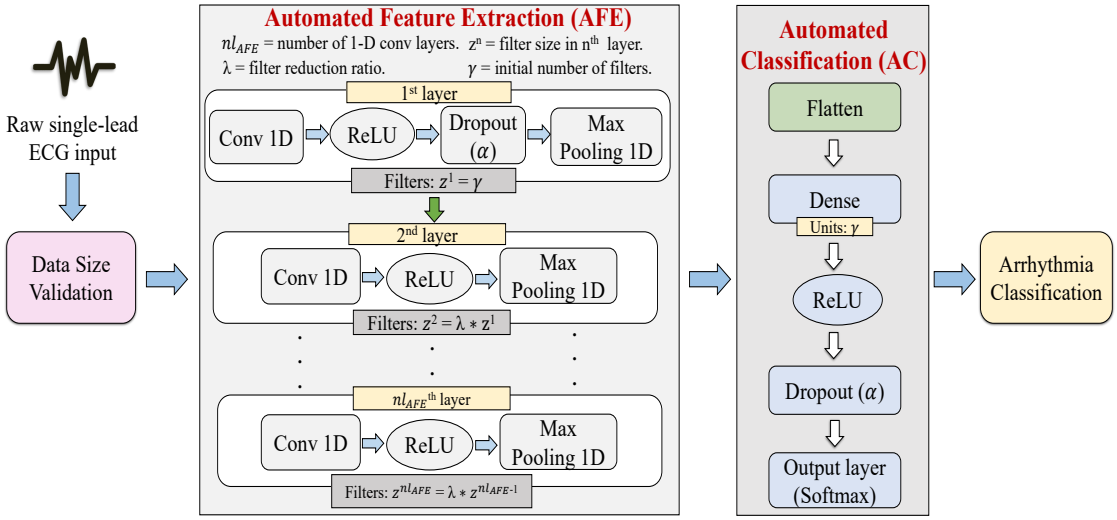


Figure 5.3: Proposed training architecture leveraging CNN structure for the considered use-case. Once the model is trained at the cloud, it is transferred to the smart IoT sensor's AI module.

not necessitate any noise-filtering and manual feature extraction. The CNN model detects unique patterns automatically from the raw single-lead ECG signal. The ECG signals are sampled at a frequency of f_s before passing to CNN as input. The lightweight ECG analysis for arrhythmia detection task takes an ECG signal as input $X = [x_1, x_2, x_3, \ldots x_n]$, and outputs a sequence of labels $Y = [y_1, y_2, y_3, \ldots y_n]$. Here each y_i represents one of four different heartbeat classes and in terms of arrhythmia classification $y_i \in \{F, N, V, S\}$. Table 5.1 exhibit of the summary of each of the classes. We consider a minimum length of ECG signal noted as δ to be passed as input to the model. Every output label corresponds to a portion of the input ECG signal, and collectively the output labels cover the full sequence of the ECG signal record of a subject.

As the deep learning-based solution, a 1-D CNN is designed and used because of its exceptional performance in automatically detecting patterns in the ECG signal. The proposed CNN model can be defined briefly as the combination of the convolution layers, max-pooling layer, and fully-connected layers. Fig. 5.3 represents the architecture of the proposed CNN model. Here the model receives raw ECG signal as input and generates heartbeat classes as output. CNN consists of two segments; the first segment comprises nl_{AFE} number of 1-D convolution layers performing Automated Feature Extraction (AFE) from the raw single-lead ECG signal and an Automated Classification (AC) module that process the extracted features using nl_{AC} number of fully connected layer followed by the output layer for classification. The 1-D convolution operation can be expressed as in Eq. 5.2.

$$x_k^l = \sum_{i \in nl_{AFE}} (x_i^{l-1} * w_i^l + b_k^l) \tag{5.2}$$

Here, x_k^l and b_k^l can be defined as the input and bias for the k^{th} node of l^{th} layer, respectively. The kernel is defined as w_i^l and the input of the i^{th} node of the $(l-1)^{th}$ layer is denoted as x_i^{l-1}. To select the optimal activation function for the proposed model, we performed hyperparameter tuning. The Rectified Linear Unit (ReLU) [111] is selected as activation function, Ω, defined previously in Eq. 2.11.

In the first convolution layer, we also apply dropout with a rate of α as the regularization technique, which will serve the network in avoiding overfitting. Hence, the model can gain enhanced generalization ability by randomly disregarding some selected neurons in the hidden layers. After the regularization layer, we employ the subsampling technique to compress the size of the ECG data and reduce computation time. We have employed the max-pooling layer to obtain the maximum value in a particular region. Eq. 5.3 determines the output of the j_{th} unit of the subsampling layer l. where x_j^l represents the output of the j_{th} unit of layer l and $x_{j_{output}}^{l-1}$ represents the j_{th} output group of layer $l-1$. The kernel size of the max-pooling layer is set to a constant km_{init} for each of the layers.

$$x_j^l = subsample(x_{j_{output}}^{l-1}) \tag{5.3}$$

The n^{th} layer of the AFE module produces a feature matrix from the ECG data. The extracted features by the initial module are relinquished to the subsequent stage for further analysis. In the next stage, the AC module consists of a single flatten layer, followed by a fully connected layer and an output layer. The flatten layer is responsible for transforming the features into a vector that can be forwarded into a fully connected [112]. ReLU and softmax activation functions are selected to be used in the fully-connected layer and output layer, respectively.

5.5.2 Deep Learning-Based Lightweight Arrhythmia Classification (DL-LAC) Algorithm

In this subsection, we present the steps of the training and inference phases of our proposed DL-LAC algorithm.

The training phase of the proposed DL-LAC algorithm includes Algorithms 4, 5, and 6. The training stage of DL-LAC commences from Algorithm 4 with the inputs D, k, ξ, B, Ω, and δ. The details of each of the inputs are provided in the algorithm's input section. The training phase of the algorithm returns the trained model (M_t), which is further harnessed in the inference phase. The algorithm initiates with initializing the required parameters in the steps 1 to 3. Then, in step 4, the ECG signal and the corresponding heartbeat class

labels are loaded from the dataset, which is later utilized in training. After that, in step 5, the ECG data is validated by checking with a pre-defined size threshold in the DSV algorithm described in Algorithm 5. Afterward, in the steps 6-11, the training ECG data and the heartbeat labels are employed to train the model (M_t) using k-fold stratified cross-validation. At the penultimate step, the trained model (M_t) is stored for further testing and validation. Finally, in step 13, the algorithm concludes by returning the trained model.

Algorithm 4: Training phase of DL-LAC

Input: D (ECG data collection for training), k (number of fold in cross-validation), ξ (number of epoch), B (mini-batch size), Ω (activation function), δ (threshold for data size)

Output: M_t (Trained model)

1 $M_t \leftarrow \emptyset$
2 $X_\delta \leftarrow []$
3 $y_\delta \leftarrow []$
4 X, $y \leftarrow$ read ECG signal and annotated heartbeats from D
5 X_δ $y_\delta \leftarrow$ call DSV(X, y) from algo. 5
6 **for** *(fold no. j=1 to k)* **do**
7 $X_{train}, y_{train}, X_{val}, y_{val} \leftarrow$ set data and labels of j^{th} fold from X_δ, y_δ
8 $F_{train} \leftarrow$ call AFE(X_{train}, Ω) from algo. 6
9 update the model parameters of M_t by passing F_{train} through the AC module as depicted in Fig 5.3
10 compute validation performance using X_{val}, y_{val}
11 **end**
12 save the model parameters of M_t
13 return M_t

For the data size validation purpose, our proposed DSV algorithm is demonstrated in Algorithm 5. This algorithm's main objective is to validate the length or size of the ECG signal by checking with a pre-defined threshold of δ. The algorithm takes X, y, and δ as input and produces an updated version of X and y, denoted as X_δ and y_δ, respectively. We utilize this algorithm in both the training and inference phase before the start of their workflow.

In the Algorithm 6, the required steps for the AFE module of the proposed model is manifested. We utilize this algorithm from step 8 of the Algorithm 4, to extract the unique features from the ECG signal. The extracted feature matrix from this algorithm is then employed in the later module for classification. The inputs to the algorithm are X and Ω, whereas the extracted unique features (F_{train}) are returned as the output of the algorithm. Step 1 and 2 initialize the required parameters. In steps 3 to 10, the automated feature extraction module's main workflow is illustrated for nl_{AFE} number of convolution layers. In step 4, the input is passed through the 1-D convolution, the results of which will then be

Algorithm 5: Data Size Validation (DSV)

Input: X (data), y (heartbeat labels), δ (threshold for data size)
Output: X_δ (updated data after size validation), y_δ (updated heartbeat labels after size validation)

$X_\delta \leftarrow []$
$y_\delta \leftarrow []$
for *(i=1 to length(X))* **do**
    if *(length(X_i) < δ)* **then**
        continue
    else
        $X_\delta \leftarrow$ add $X_i[1:\delta]$
        $y_\delta \leftarrow$ add y_i
    end
end
return X_δ, y_δ

Algorithm 6: Automated Feature Extraction (AFE)

Input: X_t (training data), Ω (activation function)
Output: F_x (extracted features)

initialize γ (initial filter size), λ (filter size reduction factor), nl_{AFE} (number of conv. layers), α (dropout rate)
$z^1 \leftarrow \gamma$
foreach *layer* $i \in nl_{AFE}$ **do**
    $F_x \leftarrow$ pass X_t through the convolution layer with z^i and Ω
    if *($i = 1$)* **then**
        $F_x \leftarrow$ apply regularization of rate α (dropout)
    end
    $F_x \leftarrow$ update F_x by passing through sub-sampling layers (max-pooling)
    $z^{i+1} \leftarrow z^i * \lambda$
end
return F_x

forwarded to the later layers. We employed dropout with a rate of α for the first convolution layer ($i = 1$), which is expressed in steps 5 to 7. Step 8 performs the sub-sampling operation using the max-pooling technique described in the previous subsection (Eq. 5.3). After that, we update the number of filters to be used by the reduction factor λ, in the next convolution layer in step 9. Finally, in step 11, the extracted feature matrix is returned for the next module to use.

In the inference phase, the proposed DL-LAC algorithm is exhibited in the Algorithm 7. It receives the location of test ECG data for inference and returns the predicted class labels (y_{pred}) for the corresponding sample. After loading the testing ECG data from step 1, the pre-trained model (M_t) is loaded in the subsequent step. Step 3 is responsible for updating the test data by validating the data length from Algorithm 5. In step 4, the model M_t is used to predict the probabilities for a sample ECG test data to belong in each of the four classes. In step 5, the class with the highest probability is selected as the classified class for each sample data. Ultimately, in the last step, the collection of predictions for all the data is returned.

Algorithm 7: Inference phase of DL-LAC

Input: $path_{test}$ (test data location)
Output: y_{pred} (predictions by the model)
1 X_{test}, y_{test} $\leftarrow$ load all test ECG data and corresponding class labels from $path_{test}$
2 $M_t \leftarrow$ load the pre-trained model
3 X_{test}, y_{test} $\leftarrow$ call DSV(X_{test}, y_{test}) from algo. 5
4 $y_{prob} \leftarrow$ predict the probabilities for each sample of X_{test} employing the model M_t
5 $y_{pred} \leftarrow$ argmax(y_{prob})
6 return y_{pred}

5.5.3 Computational complexity analysis in terms of mathematical operation

This section investigates the algorithm's complexity and the time cost to run the proposed deep learning-based lightweight ECG monitoring system to detect arrhythmia. We analyze the complexity of the proposed model's training and inference steps in terms of the number of different operations required by various stages of the model. The analysis primarily encompasses the mathematical analysis of the algorithm complexity in the training phase and inference phase through determining the recurrence of each operation (e.g., addition, subtraction, multiplication, and division, etc.). We express the addition and multiplication operations as ADD and MUL, respectively. In addition, we also analyze the occurrence of comparisons denoted as COMP.

5.5.3.1 Training phase

The training phase comprises the DSV algorithm for data augmentation and the DL-LAC training phase for the proposed CNN model. In the training phase of DL-LAC, depicted in Algorithm 4, we perform computational overhead analysis, considering that the appropriate hyperparameters of the proposed models are already selected after hyperparameter tuning employing the grid search technique. We divide the overall analysis of the training phase, mainly into three different fragments, such as the required data size validation phase, feature extraction phase, and the classification phase. Therefore, the total computational complexity can be expressed as Eq. 5.4:

$$C(Training) = C(DSV) + C(AFE) + C(AC). \tag{5.4}$$

Here, $C(DSV), C(AFE)$, and $C(AC)$ indicate the required computational overhead in the data size validation, automated feature extraction, and automated classification phases, respectively. For each of these three phases, the computation complexity is divided into three parts: the required number of additions, multiplications, and comparisons. In the first stage, to calculate the complexity of the data size validation phase, we mainly analyze the complexity of the Algorithm 5, which is invoked from the training procedure (Algorithm 4). The first four steps of the training algorithm are initializing steps; hence these do not require any mathematical operations (i.e., addition and multiplication). In step 5, the Algorithm 5 is invoked for validating the ECG data size. If the length of considered training ECG trace is $len(X_{train})$, then the required number of comparisons is also $len(X_{train})$ as the condition will be validated for each ECG trace.

In the next phase, the computational overhead is determined for the feature extraction phase manifested in the Algorithm 6 of the training procedure. For a particular layer (l^{th} layer) of the AFE module, if we consider that there are N^l number of nodes for the convolution layer, then the number of required operations can be defined as Eqs. 5.5 and 5.6.

$$\begin{aligned} C(AFE_{ADD}) = nl_{AFE} * \xi * N^l * (len(x^l)/B) * ((len(k^l) * len(x^{l-1})) - (len(k^l) - \eta) + 1) \\ - (len(x^{l-1}) - (len(k^l) - \eta)) * z^l)), \end{aligned} \tag{5.5}$$

$$\begin{aligned} C(AFE_{MUL}) = nl_{AFE} * \xi * N^l * (len(x^l)/B) \\ * (z^l * ((len(k^l) * len(x^{l-1})) - (len(k^l) - \eta) + 1)) + (nl_{AFE} - 1). \end{aligned} \tag{5.6}$$

Here, x^l, k^l, and z^l indicate the input, kernel, and the number of filters of layer l. The striding window length, the number of epoch, and batch sizes are denoted by η, ξ, and B, respectively.

Also, in terms of the AFE phase, the number of comparisons required for nl_{AFE} layers can be denoted as eq. 5.7. Here, x^l and z^l implies the input and the number of filters in the l^{th} layer of the AFE phase. For z^l number of filters, the number of comparisons required at the layer l due to passing the input x^l through the activation function (Ω) is $((z^l * len(x^l))$. In the sub-sampling layer (i.e., max-pooling layer), the number of comparisons required is $(len(x^l) - (z^l - 1))$.

$$C(AFE_{COMP}) = \sum_{l=1}^{nl_{AFE}} (\xi * N^l * (len(x^l)/B) * ((z^l * len(x^l)) + (len(x^l) - (z^l - 1)))). \quad (5.7)$$

The extracted features set (F_x) of the AFE phase will be relinquished to the AC module of the proposed model for the classification task. For a particular layer denoted as l, if the output of the i^{th} layer is γ^i, then the computational complexity for the i^{th} layer can be $(len(\gamma^i) * (len(F_x) - 1))\ ADD,\ (len(\gamma^i) * len(F_x))\ MUL$. Thus considering the number of fully-connected layers to be nl_{AC}, the computational complexity of this phase can be denoted as Eqs. 5.8 and 5.9.

$$C(AC_{ADD}) = \sum_{i=1}^{nl_{AC}} (\xi + (len(\gamma^i)) * (len(F_x) - 1), \quad (5.8)$$

$$C(AC_{MUL}) = \sum_{i=1}^{nl_{AC}} (len(\gamma^i)) * len(F_x) * \xi. \quad (5.9)$$

In terms of the number of comparisons required in the AC phase, considering nl_{AC} layers, the cumulative comparisons due to the comparisons as are necessary for computing the activation functions can be denoted as Eq. 5.10.

$$C(AC_{COMP}) = \sum_{i=1}^{nl_{AC}} (\xi * len(\gamma^i)). \quad (5.10)$$

Hence, by substituting the equations, as mentioned earlier in the Eq. 5.4, the overall computational complexity in terms of the number of mathematical operations required in the training phase of the proposed DL-LAC algorithm can be expressed as Eq. 5.11. The number of comparisons needed in different stages of the DL-LAC algorithm's training phase is also considered in this equation.

$$C(Training) = \begin{cases} ADD: & C(AFE_{ADD}) + C(AC_{ADD}) \\ MUL: & C(AFE_{MUL}) + C(AC_{MUL}) \\ COMP: & len(X_{train}) + C(AFE_{COMP}) \\ & + C(AC_{COMP}). \end{cases} \quad (5.11)$$

5.5.3.2 Inference phase

The inference/running phase is conducted to infer classes of each testing ECG data employing the pre-trained lightweight model (M_t) and then evaluating it using the unseen data. In correspondence with Algorithm 7, if we consider the test data to be X_{test}, and the size of test data after validating ECG signal is $len(X_{test})$, then the computational complexity of the inference phase can be denoted as follows:

$$C(Inference) = \begin{cases} ADD: & \sum_{i=1}^{nl_{AFE}}(len(x^i) - (\eta+1)) \\ & + \sum_{j=1}^{nl_{AC}}(len(\gamma^j) - 1) \\ MUL: & \sum_{i=1}^{nl_{AFE}}(len(x^i) - \eta) \\ & + \sum_{j=1}^{nl_{AC}}(len(\gamma^j)) \\ COMP: & \sum_{i=1}^{nl_{AFE}}(len(x^i)) \\ & + \sum_{j=1}^{nl_{AC}}(len(\gamma^j)) \\ & + len(X_{test}) \end{cases} \tag{5.12}$$

Eq. 5.12 illustrates that, in the inference phase, the pre-trained model is able to produce results with considerably lower computational operations (i.e., upper bound time-complexity of $O(len(X_{test})$, in Big O notation). The complexity analysis indicates that it can be utilized for lightweight arrhythmia classification at the resource-constrained ultra-edge IoT node.

5.6 Performance Evaluation

This section manifests the simulation results to establish the algorithmic analysis of the proposed lightweight DL-LAC method that is estimated in the previous section. As we have employed four different datasets with single-lead ECG, no existing research considered these many datasets with one lead ECG. The methodologies in the current literature can be slightly are mostly heavyweight, which makes them slower and not suitable for real-time analysis at the ultra-edge nodes. Most contemporary work utilizes extensive pre-processing and feature extraction phases with either binary or multi-class labels (i.e., more than four heartbeat classes) by analyzing multi-lead ECG data [113]. Therefore, we could not conduct a comparative analysis from the literature due to the lack of identical experimental settings (i.e., ECG lead usage and heartbeat class labels). Thus, the proposed method (DL-LAC) is compared with the traditional techniques (employing single-lead ECG) in terms of classification performance, memory consumption, and required inference time, using these four datasets. As the analyzed datasets comprise already segmented heartbeats, our analysis did not consider the cost of heartbeat segmentation during the inference phase, which may be required as a fixed cost in real-life use-cases during the running stage. In running phase,

standard heartbeat segmentation techniques (e.g., Pan and Tompikins) can be employed before passing the ECG data to the AI model [114]. However, we did not consider the fixed cost for heartbeat segmentation in our analysis as the considered dataset already contained segmented heartbeats.

5.6.1 Performance Indicators

To evaluate the classification results, we adopted the combination of three measurement indicators, accuracy, weighted precision, and weighted F1 score. The accuracy of a test is its ability to correctly differentiate the three cases. Considering, C = Number of classes in the considered classification task, $len(y_i)$ = number of samples in the i^{th} class, TP_i = the number of cases correctly identified to be in the i^{th} class, and $len(Y)$ = total number of samples in all the class, the accuracy can be denoted as Eq. 5.13:

$$\text{Accuracy} = \frac{\sum_{i=1}^{C}(TP_i)}{len(Y)}. \tag{5.13}$$

The weighted precision can be expressed as Eq. 5.14. It addresses how precise the model is out of those predicted to be in i^{th} class, how many of them are actually in i^{th} class, and the value is multiplied by the weight of the i^{th} class as follows:

$$\text{Weighted precision} = \sum_{i=1}^{C}(\frac{len(y_i)}{len(Y)} * \frac{TP_i}{TP_i + FP_i}). \tag{5.14}$$

Here, FP_i represents the number of cases incorrectly identified to be in the i^{th} class. Weighted F1 score is the weighted average of precision and recall. Hence, although we did not use recall directly as a performance measure, because of using the F1 score, it is implicitly used. The weighted F1 score can be obtained as follows:

$$\text{Weighted F1 score} = \sum_{i=1}^{C}(\frac{len(y_i)}{len(Y)} * 2\frac{P_i * R_i}{P_i + R_i}). \tag{5.15}$$

In the above equation, the precision and recall of i^{th} class are indicated by P_i and R_i, respectively. P_i can be expressed as $TP_i/(TP_i + FP_i)$ and R_i can be denoted as $TP_i/(TP_i + FN_i)$. FN_i denotes the number of cases incorrectly identified as a class other than the i^{th} class.

5.6.2 Results and Discussion

We have conducted comprehensive experiments in a systematic approach to identify the optimal model. Here, the experimental results can be summarized as follows:

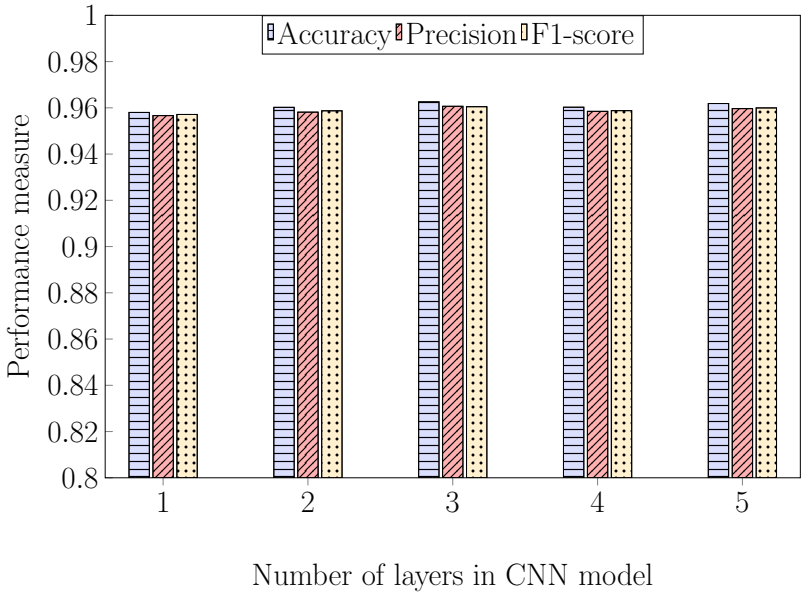
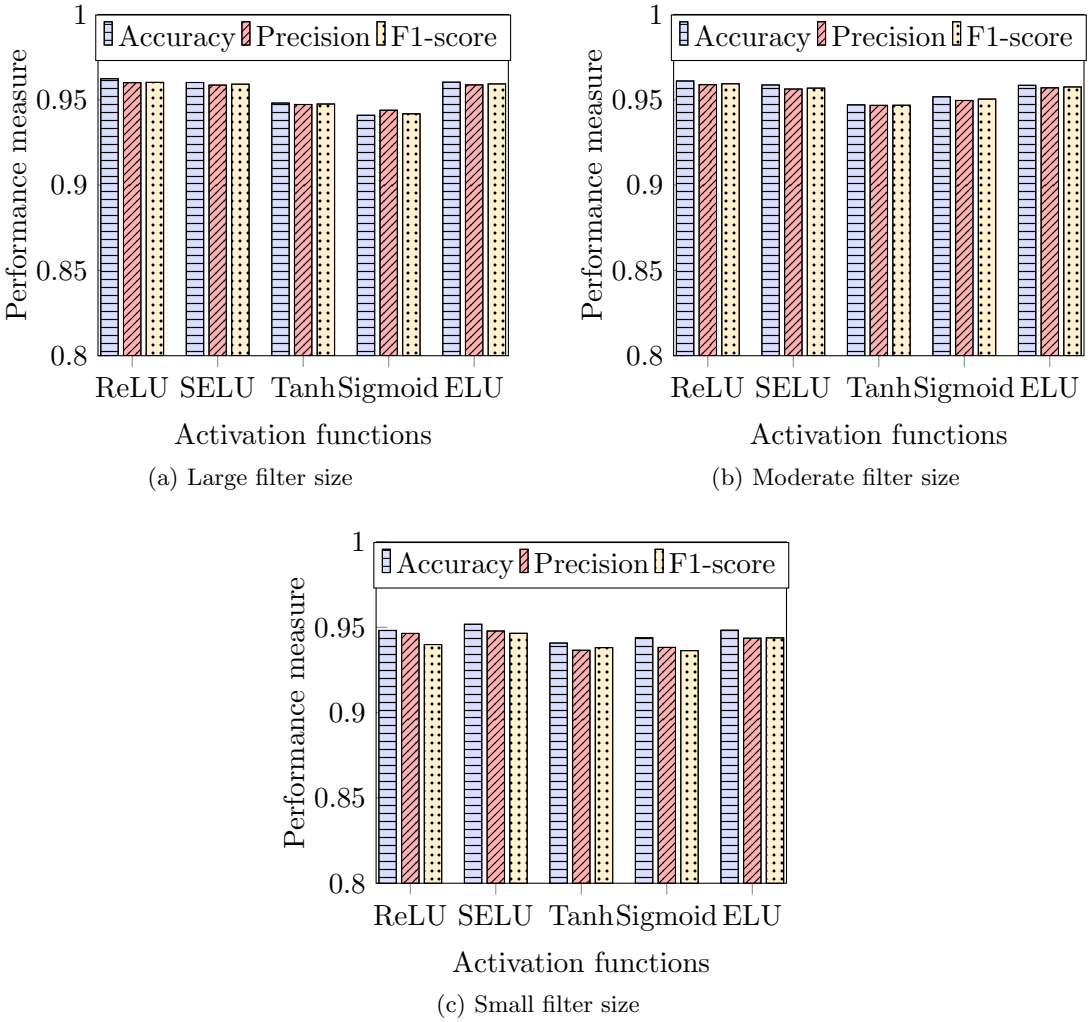


Figure 5.4: Performance variation of the proposed/custom CNN model with varying numbers of layers.

- The first phase of the experiment encompasses the hyperparameter tuning to find the optimal structure of the model. The selected hyperparameters were applied in the proposed DL-based model.
- In the second phase, we measured the model's performance employing the trained model obtained from DS1 and then tested it using MIT-BIH Arrhythmia Database (DS2), St Petersburg INCART 12-lead Arrhythmia Database (DS3), and Sudden Cardiac Death Holter Database (DS4).
- In the third phase of the experiment, we evaluated the proposed model's generalization ability by utilizing each of the four datasets individually for training and testing purposes using k-fold cross-validation.
- Finally, numerical analysis is carried out to assess the proposed CNN models' effectiveness in terms of execution time required and memory consumption in various IoT devices and compared to the traditional ML techniques.

5.6.2.1 Hyperparameter Tuning

We performed hyperparameter tuning to select the optimal parameters for the proposed CNN-based model in the initial phase of the experiment. Figure 5.4 demonstrates the results of manual tuning for the number of convolution layers used in the model by varying the number from one to five. The experimental results illustrate that, for three convolution layers, the best performance is achieved with 96.26%, 0.9606, and 0.9604 accuracy, precision,



(a) Large filter size

(b) Moderate filter size

(c) Small filter size

Figure 5.5: Performance comparison for different activation functions with respect to different filter size of the proposed CNN.

and F1-score, respectively. Therefore, we conducted further analysis using three number of convolution layers in the proposed DL-based architecture.

Furthermore, to select the optimal activation function (Ω) and the number of the initial filter size (γ), we performed a grid search technique. Figure 5.5 demonstrates the results obtained from the grid search where 5.5a, 5.5b, 5.5c represents the initial number of filter equals to large, moderate, and small, respectively. For the grid search, we considered three sizes for the filters of the first convolution layer, such as large, moderate, and small, with the value of 300, 150, and 50, respectively. For selecting activation function (Ω), we experimented with a set of five activation functions: ReLU, SELU, ELU, Tanh, and Sigmoid. According to performance, the best combination is evident when the activation is ReLU,

and the number of filters is large with the accuracy, precision, and F1-score, respectively 96.23%, 0.96004, and 0.9601.

Additionally, to elect the optimal optimizer, batch size, dropout, and epochs, we performed a grid search, which is manifested in Table 5.3. We have conducted the grid search among six widely used optimizers such as Adadelta, Nadam (Nesterov-accelerated Adaptive Moment Estimation), SGD (Stochastic Gradient Descent), RMSprop (Root Mean Square Propagation), Adagrad (Adaptive Gradient Algorithm), and Adam (Adaptive Moment Estimation). For the batch size, we tuned the value employing a set of three different values, such as 1000, 2500, and 5000. For selecting the optimal dropout rate (α), we considered values from 0.1 to 0.5. We varied the number of epochs (ξ) using three values (i.e., 10, 50, and 100). The best performing combination is obtained for the Adam optimizer along with batch size 5000, dropout rate 0.5, and the number of epochs 10. Hence, for further analysis of the experiment, we employed these parameter values for the model.

5.6.2.2 Inference Results

In the second phase of the experiment, we utilized DS1 as the training dataset and then tested the model's performance using DS2, DS3, and DS4 as test datasets. Table 5.4 illustrates the results for this phase. In all three test datasets, the proposed model outperformed the traditional ML methods (i.e., random forest, KNN) in terms of accuracy, precision, and F1-score. The proposed CNN achieved an accuracy of 94.07%, 92.04%, and 95.83%, while DS2, DS3, and DS4 are harnessed as the test dataset, respectively. The proposed custom CNN model is showing superior performance over traditional ML techniques because the combination of Convolution, sum-sampling, and regularization layers are able to capture the detailed features from the ECG signal automatically. Furthermore, due to the adaptive filter reduction in the deep convolution layers, the proposed model can identify significant points from the ECG with higher efficiency, and because of the use of the regularization layer, the proposed approach is able to avoid overfitting during training. However, the traditional methods are lacking the ability to automatically retrieve significant features

Table 5.3: Selected parameters for each optimizer after employing grid search.

Optimizer	**Selected paramters**			**Accuracy**
	Batch size	Dropout rate	Epochs	
Adadelta	5000	0.2	100	88.55%
Nadam	2500	0.2	10	88.67%
SGD	5000	0.4	100	88.81%
RMSprop	1000	0.4	10	89.19%
Adagrad	1000	0.5	10	89.56%
Adam	5000	0.5	10	**91.85%**

Table 5.4: Performance comparison of CNN with traditional ML methods for the second experimental setting using DS1 as the training dataset.

Method	**Test Dataset**	**Accuracy**	**Precision**	**F1-Score**	**Noise filtering**	**Feature extraction**	**ECG type**
KNN	DS2	89.83%	0.8541	0.8646	3-phased noise filtering	Required	Single lead
	DS3	89.76%	0.9281	0.9124			
	DS4	56.53%	0.7383	0.6991			
RF	DS2	89.31%	0.901	0.8957	3-phased noise filtering	Required	Single lead
	DS3	90.21%	0.8921	0.8844			
	DS4	85.77%	0.9126	0.8947			
CNN	DS2	94.07%	0.9071	0.9176	**Not required**	**Not required**	Single lead
	DS3	92.04%	0.8991	0.9018			
	DS4	95.83%	0.9563	0.9573			

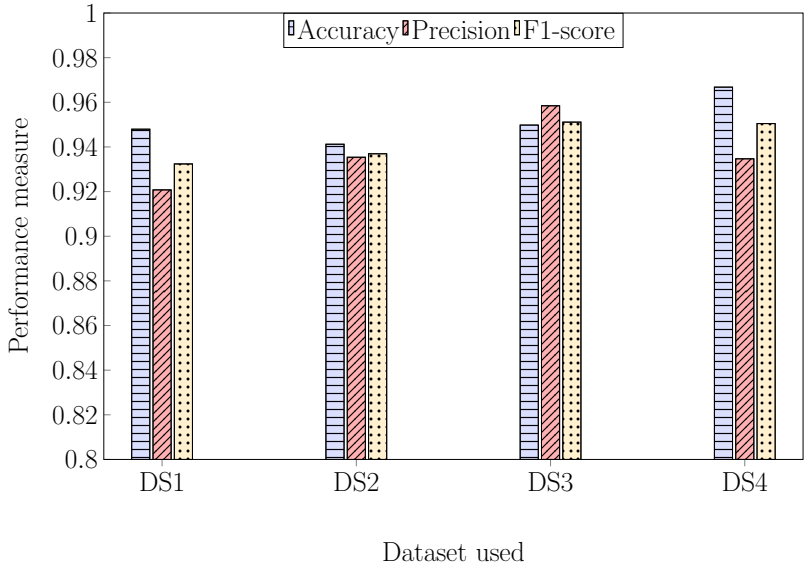
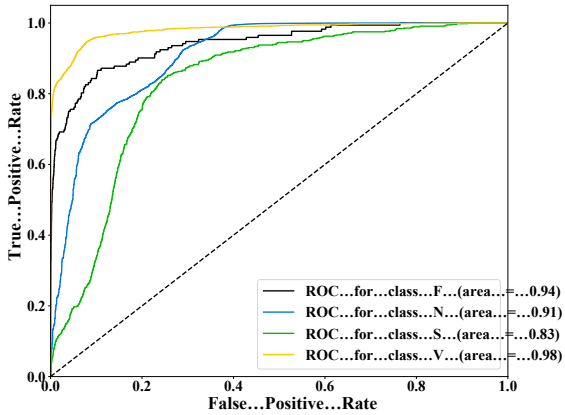


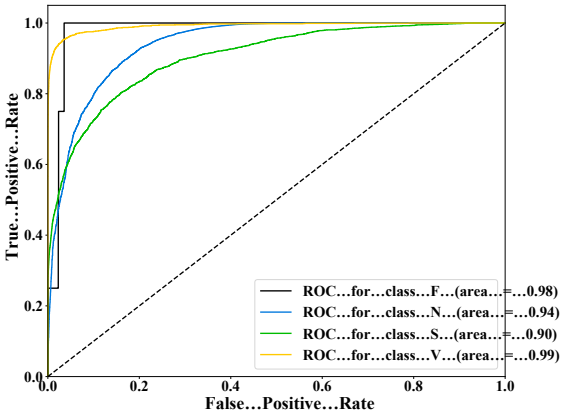
Figure 5.6: Performance of the proposed model for the third experimental setting employing the four datasets individually (3-fold stratified cross-validation). Here, DSi means the i^{th} dataset.

from the ECG even after extensive noise-filtering stages. The proposed model outperforms the traditional methods in terms of performance, but the vital part is that the proposed technique can achieve great accuracy even with raw ECG signals, without adopting noise-filtering and typical feature extraction of the ECG. Although the proposed technique does not require noise-filtering and feature extraction phases, in the inference phase, some pre-processing costs will be needed due to the standard heartbeat segmentation phase, and this heartbeat segmentation part results in a fixed cost for all the methods. The results reveal that the custom CNN-based model is robust in detecting arrhythmia with high accuracy and lightweight because of using raw single-lead ECG.

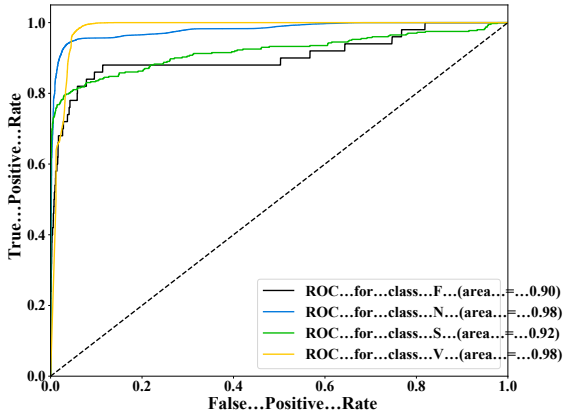
In the penultimate experimental phase (third phase), we experimented using each dataset



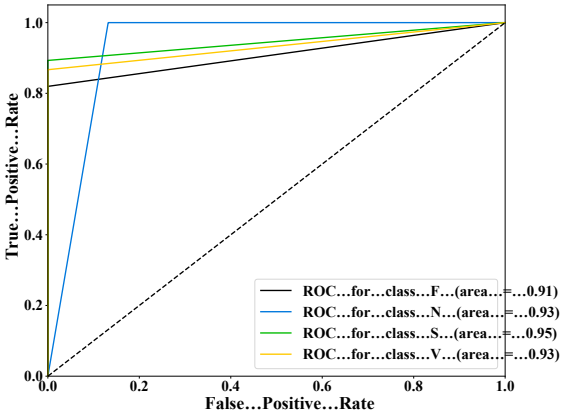
(a) ROC curve employing DS1 (AUC Score: 0.9113)



(b) ROC curve employing DS2 (AUC Score: 0.9406)



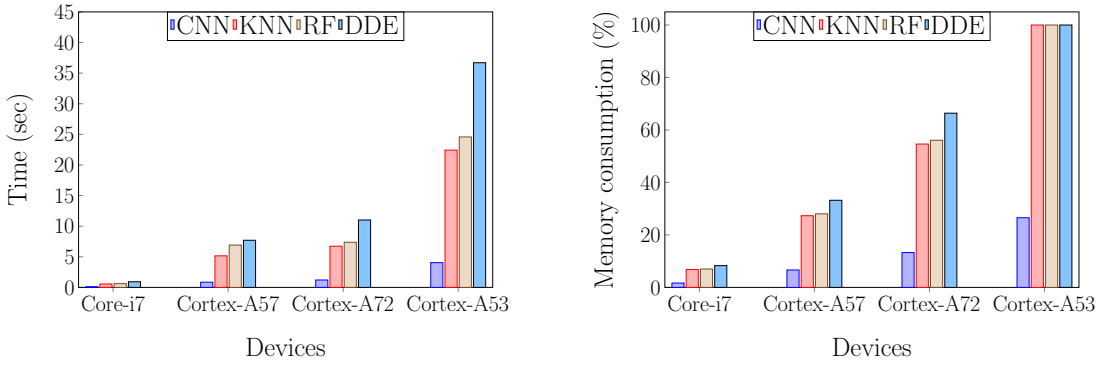
(c) ROC curve employing DS3 (AUC Score: 0.9796)



(d) ROC curve employing DS4 (AUC Score: 0.9340)

Figure 5.7: Area Under the Receiver Operating Characteristic (AUROC) curve derived for the third experimental settings utilizing 3-fold stratified cross-validation.

individually, as manifested in Fig. 5.6, employing 3-fold stratified cross-validation to validate the generalization capability of the proposed model. Stratification is a method in which the samples are rearranged to have a stable representation of the whole dataset by preserving the portion of samples for each class [115]. The cross-validation is performed after splitting each of the four datasets into 80%-20% for training and testing purposes. On the testing part of the dataset, the accuracy values of the model are 94.79%, 94.12%, 94.97%, and 96.67%, respectively, for DS1, DS2, DS3, and DS4. The encouraging results illustrate the model's ability to generalize diverse types of ECG signals to classify arrhythmias. To investigate the classification efficiency for each class, we manifested the AUROC curve for each class. Figure 5.7 exhibits the ROC curves where 5.7a, 5.7b, 5.7c, and 5.7d corresponds to the ROC curves for DS1, DS2, DS3, and DS4, respectively. For each dataset, the model



(a) Time required for different devices (in seconds)

(b) Memory consumption (%) by different methods for different devices

Figure 5.8: Required execution time and memory consumption of various methods on a workstation and different micro-controllers used as a proof-of-concept for the smart sensor.

demonstrated a high AUC score. The AUC scores for the four datasets are 0.9113, 0.9406, 0.9796, and 0.9340 for DS1 to DS4, respectively. The promising results prove the model's efficiency in distinguishing different classes of heartbeats to classify arrhythmia by employing a raw ECG signal.

5.6.2.3 Numerical Analysis

Finally, we conducted the numerical analysis in terms of time delay and memory consumption (in percentage) of the proposed model and compared it with traditional ML approaches (i.e., KNN, RF). Here, we have numerically calculated the approximated time and memory requirements for the proposed and traditional methods, considering that the ECG heartbeat segmentation is already performed because the segmentation was already completed in the adopted datasets. However, in the real-life arrhythmia monitoring use-case, the automatic heartbeat segmentation is expected to consume a fixed time and memory for all the methods. Figure 5.8 illustrates the results obtained from the analysis. The initial experiment was conducted on a workstation with Intel Core i7, 3.00GHz CPU, 16 GB RAM, powered by Nvidia RTX 2060 GPU. We approximated the time and memory consumption required for different IoT devices to determine the model's potential to integrate with the logic-in-sensor. The microprocessor-based IoT devices we considered for the numerical analysis are Jetson Nano (Quad-core ARM A57 @ 1.43GHz), Raspberry Pi 4 (Quad-core Cortex-A72 @ 1.5GHz), and Raspberry Pi 3 (Quad-core Cortex-A53 @ 1.4GHz). Figs. 5.8a and 5.8b exhibit that the proposed CNN-based model can be beneficial for real-time analysis of the ECG signal as the model can perform efficiently with limited resources due to employing raw-ECG signal without any manual feature extraction.

The complexity analysis explained in Sec. V and the experimental outcomes presented in

this section precisely confirm that the proposed lightweight ECG classification method can be considered as a viable solution for embedding intelligence into the resource-constrained ultra-edge IoT nodes. The proposed DL-LAC method's generalization aptitude was evaluated on four separate, publicly available real datasets by adopting multiple experimental settings. Promising experimental results signify that the proposed method performed with efficiency in all the experiments. Therefore, the model can be utilized for the ultra-edge IoT sensors to enhance healthcare services.

5.7 Summary

Centralized cloud-based analytics and edge analytics on smart devices are the traditional health automatic monitoring approaches. To make smart health even effective and secure, in this chapter, we focus on the necessity to go beyond the realms of conventional methods and investigate how to incorporate intelligence into the ultra-edge IoT nodes. As an example of smart ultra-edge health monitoring, we selected arrhythmia (a cardiovascular disease) classification by analyzing the ECG signal. As the sensors are resource-constrained, we designed a DL-based lightweight heartbeat classification model named DL-LAC that utilizes raw single-lead ECG to classify arrhythmia with encouraging efficiency, eliminating the necessity of pre-processing (i.e., noise-filtering) and feature extraction. We compared the proposed method with traditional machine learning (e.g., KNN, random forest) and the DDE-based optimization technique. The proposed method's generalization ability was evaluated using four different datasets. The datasets contain segmented heartbeats of the ECG signal; hence, we did not consider the expected cost (i.e., time and memory requirements) of the heartbeat segmentation. However, in a real-life use case for all the methods analyzed in this chapter, there would be a fixed cost for the heartbeat segmentation. By adopting multiple experimental settings, the promising results manifest that the proposed DL-LAC technique has the potential to be coupled with smart IoT sensors for ultra-edge computing to enhance the existing ECG monitoring system. Therefore, this research can be considered as a pioneering footprint to encourage the sensor foundries to consider embedding intelligence into IoT devices, and if it can be produced in mass production, the fabrication cost of the intelligent sensors can be significantly reduced.

Chapter 6

Asynchronous Federated Learning-Based ECG Analysis for Arrhythmia Detection: A Remote Health Monitoring Use-case

With the rapid elevation of technologies such as the Internet of Things (IoT) and Artificial Intelligence (AI), the traditional cloud analytics-based approach is not suitable for a long time and secure health monitoring. The privacy issues of the acquired ECG data and other health data of the patients have also arisen much concern in the cloud analytics approach. Although the recent push towards moving intelligence closer to the edge has the potential to minimize the massive privacy concern of the conventional cloud-based method, it still lacks the ability to enhance itself via online learning. To accelerate the healthcare system for remote patient monitoring, we have considered a critical use-case of cardiac activity monitoring by detecting arrhythmia from analyzing Electrocardiogram (ECG). We have envisioned an asynchronous Federated Learning (FL) architecture for arrhythmia classification using the local ECG data acquired within each mobile smart sensor, deployed at the Ultra-Edge Nodes (UENs). The envisioned paradigm allows privacy-preservation as well as the ability to accomplish online knowledge sharing by performing localized and distributed learning in a lightweight manner. Our proposed federated learning architecture for ECG analysis is further customized by asynchronously updating the shallow and deep model parameters of a custom Convolutional Neural Network (CNN)-based lightweight AI model to minimize valuable communication bandwidth consumption. By considering four different heartbeats classes, the proposed model's performance and generalization abilities are assessed, employing four distinct publicly available datasets. The experimental results demonstrate that the proposed asynchronous federated learning approach can obtain identical classification

performance compared to the existing techniques while also ensuring privacy, adaptability to different subjects, and minimizing the network bandwidth consumption.

6.1 Introduction

The Internet of Things (IoT) is apprehended to be an indispensable enabler of the next generation smart society. With the growing demand for remote health monitoring, the need for a paradigm shift from Artificial Intelligence (AI)-aided centralized remote cloud computing toward edge analytics for biomedical devices is rapidly growing [116]. Conventionally, the IoT devices and wearables have been utilized to collect and trace various well-being indicators such as Electrocardiogram (ECG), Electroencephalogram (EEG), and so forth. For implementing remote health monitoring, this traditional cloud-based analytics paradigm of regular IoT monitors contributes to prolonged delay, massive bandwidth consumption, and privacy concerns associated with the user's health data. In recent research, these challenges of traditional analytics were addressed and overcame with the emergence of a lightweight AI model deployed at the ultra-edge IoT nodes [117]. However, the ultra-edge health monitoring system lacks online learning capability, which cannot be achieved without re-training. Hence, in this paper, we address this pressing need of designing a distributed online learning technique (i.e., Federated Learning) using the Ultra-Edge Node (UENs) as the participating users, as depicted in Fig. 6.1, to bring localized intelligence to biomedical edge devices as well as ensuring the online learning capability.

We can summarize the main contributions of the paper as follows:

- Firstly, as depicted in Fig. 6.1, we enunciate the inherent shortcoming of the traditional cloud analytics-based health data analysis and reveal the pressing need for

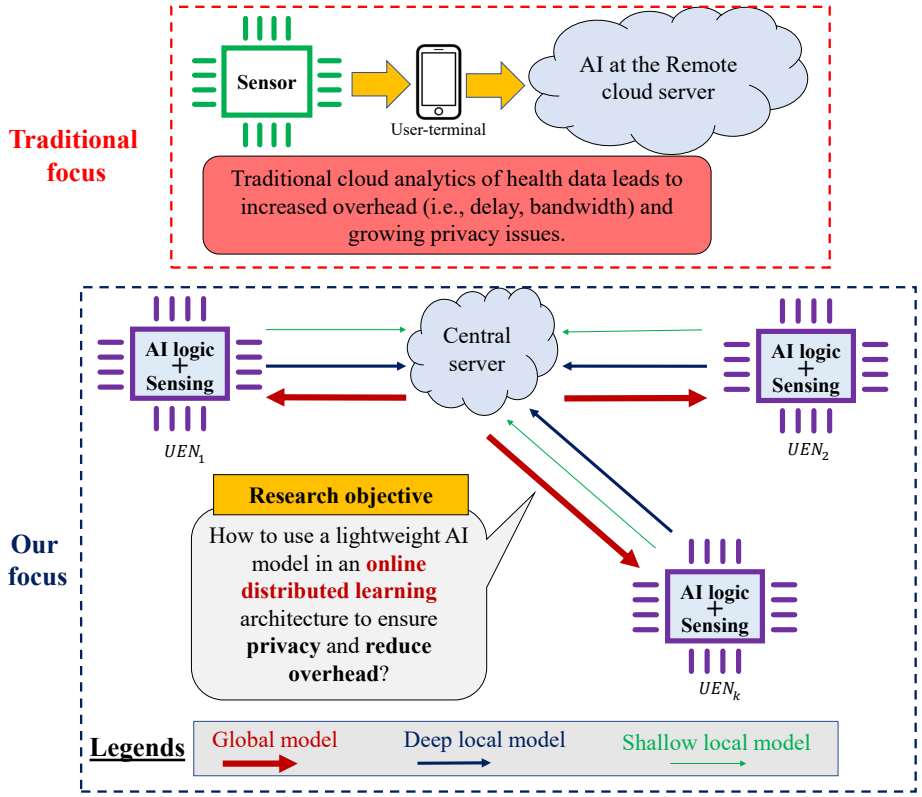


Figure 6.1: Our main focus is to develop an asynchronously federated learning-based ECG analytic methodology at the distributed Ultra-Edge nodes (UENs) to classify irregular heartbeats while preserving patient-data privacy.

deploying lightweight AI solutions in a collaborative, distributed, and online paradigm to move the AI analytics from the cloud to the ultra-edge nodes to facilitate uninterrupted remote health monitoring with enhanced privacy.

- Among a diverse set of use-cases, we have chosen the use-case of cardiac arrhythmia monitoring by analyzing ECG data. Arrhythmia is a significant contributor to cardiovascular diseases (CVD). Despite various technological advancements in the healthcare system, cardiovascular disease such as arrhythmia serves a noteworthy global public health predicament. It is still the most prominent life-threatening disease worldwide, with 15–20% of global mortalities [83]. According to American Heart Association (AHA), CVD costs will rise to approximately $750 billion by 2035. However, due to the privacy-sensitive of the ECG and the complexities of collecting ECG, obtaining a significant quantity of ECG to train a centralized machine learning (ML)-based model of arrhythmia detection is quite complicated and even not feasible under some circumstances such as remote monitoring of subjects. To ensure remote and secure cardiac monitoring in a lightweight manner with online learning ability, we employ two vari-

ations of the federated learning paradigm: asynchronous federated learning (referred to as Async-FL) and synchronous federated learning (referred to as Sync-FL). We propose the Async-FL as the most suitable distributed learning architecture for ECG analysis. It ensures less overhead and can obtain identical classification efficiency compared to the Sync-FL.

- As depicted in the second part of Fig. 6.1, our focus is to develop an agile data acquisition system for ECG for arrhythmia detection with mobile and deployable ultra-edge nodes (UENs) acting as edge computing nodes. A UEN can be considered a sensing node with a localized AI model to collect and gain knowledge from the subject's ECG data. As the AI model, we have used a (1-D) convolutional neural network (CNN)-based deep learning model (DL-LAC model) previously designed in one of our earlier papers [117]. We had presented that the proposed model can be used to classify heartbeats employing raw single-lead. The AI model does not require any pre-processing (i.e., noise-filtering) of the ECG signal, making the system lightweight and easy to integrate with the ultra-edge node. To design a distributed learning setup, the model will be initially trained and deployed to the UENs; upon deploying, the model will be updated asynchronously using the Async-FL architecture. After updating its local AI model, the UEN shares the model parameters with neighboring UENs and the cloud by efficiently scheduling the shallow and deep parameters to reduce the communication overhead. By employing this unique concept of asynchronously updating AI model parameters (i.e., model weights) of the UENs, the privacy of the acquired medical data and network efficiency are jointly preserved.

- Rigorous experimental analysis of our proposed Async-FL approach is presented using four publicly open and clinically graded ECG datasets. The encouraging results of the proposed method illustrate its potential to implement the envisioned concept in remote cardiac activity monitoring as a practical solution. To the best of our knowledge, there is no existing work in the literature that focuses on the distributed and online machine learning architecture for remote cardiac monitoring (i.e., arrhythmia) employing the ultra-edge IoT nodes (UENs) and Async-FL.

The remainder of the chapter is organized as follows. Section 6.2 surveys the relevant research work on combating arrhythmia with AI techniques. The formal problem is described in section 6.3. Next, our considered distributed learning focused asynchronously updating federated learning-based system model and proposed asynchronously updating federated learning algorithm is presented in section 6.4. The performance of our proposal is evaluated in section 6.5. Finally, section 6.6 concludes the chapter.

6.2 Related Work

In cloud-based ECG monitoring systems, several techniques are utilized, including feature extraction and classification. Discrete Wavelet Transformation (DWT) and Artificial Neural Network (ANN) were adopted for feature extraction, and for binary classification of heartbeat (normal or abnormal) are used in [118]. The authors in [98] trained a neural network (i.e., convolutional neural network) to classify 12 cardiac rhythm categories using the single-lead ECGs and achieved a higher classification efficiency than the traditional cardiologists. DWT and non-linear delay differential equations (DDE)-based optimization techniques [103], proposed in the literature for time-series ECG monitoring task, are unable to infer the system models in varying heart conditions adequately. An exhaustive exploration needs to be conducted to choose the most proper structure for the classification task using these approaches. For choosing an optimal DDE-based classification model, a Genetic Algorithm (GA) was applied in [105]. In another research [119], sparse decomposition was adopted for efficient feature extraction, and K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Radial Basis Function Neural Network were applied for classification. In another work, [120] the authors employed different separate manual feature extraction techniques to determine features such as P-wave interval, QRS interval, and QT interval from 12-lead ECGs and applied the support vector machine model to detect myocardial infarction. However, most of the existing literature works are not focusing on designing lightweight AI models and are utilizing a centralized ML algorithm to ignore the privacy leaks in collecting data. This means that private ECG data need to be collected and shared to train a data-driven ML model to detect the arrhythmia events. Hence, these approaches are impractical in ultra-edge nodes due to their significantly high computational complexity.

The edge computing-focused ECG analytics was investigated in a few studies. Authors in [121] developed an ECG analysis algorithm with noise-filtering and manual feature extraction phases and implemented it on an IoT-based embedded platform. To press the need for collaborative online learning, the authors in [122] introduced a federated learning-based distributed algorithm that allows each medical hospital to participate in the AI model's training locally cooperatively. However, in the existing literature, the viability of the decentralized online federated learning technique for deployment in the ultra-edge nodes for remote ECG monitoring is still yet to be explored elaborately, and hence we focus on this research gap in this paper.

6.3 Problem Description

Cardiac arrests and arrhythmias are indirectly related to pandemics such as COVID-19 infection, especially among the more critical subjects [123, 124]. Along with the rapidly

changing variations of the COVID-19, their effect on cardiac activities needs to be monitored in a distributed manner to obtain the varying impact of different strains and a wide range of data distributions on the cardiac status. In rural areas where healthcare services are challenging to access for long-time cardiac monitoring, the UENs need to provide service coverage for subjects. The cardiac state (i.e., irregular heartbeats) of each subject can be obtained from each UEN. Multiple UENs can summarize a particular area so that the medical service providers can get an overall representation of the region. Thus, the initial challenge is to expedite distributed learning among the UENs to adapt and enhance itself with varying data distribution acquired from different UENs via the collaborative and online learning approach. Furthermore, there is a massive privacy requirement for the private data of the users as they are not willing to share the raw health-related data with a remote cloud server to avoid potential security threats.

Moreover, for each UEN, the private data should be utilized securely without transmitting raw data somewhere else and then removed upon local automatic decision making is completed. Thereby, the UENs need a distributed learning architecture to extract distinctive ECG features with enhanced privacy and learning efficiency. This task of decentralized or distributed collaborative learning can be a critical challenge because the UENs will not have the global status of all the other subjects. To resolve this challenge, we have investigated the different variations of federated learning techniques such as typical synchronous federated learning and asynchronous federated learning. The UENs who will act as the edge users in this collaborative learning scenario could also exchange knowledge gained from private data with the server to obtain global knowledge and ensure online learning capability. This approach will ensure data privacy, reduced latency, and adaptation ability to varying data distribution among diverse subjects. After receiving the local models from the UENs, the server will update the global model to obtain an optimized and efficient arrhythmia detection model. In other words, the purpose of loss function minimization can be observed as follows:

$$\min \sum_{i=1}^{n} \frac{\mathcal{X}_i}{\mathcal{X}} f_i(w), \tag{6.1}$$

where $f_i(w)$, $\mathcal{X}_i$, and $\mathcal{X}$ denote the loss function of UEN_i, the private ECG sample data of UEN_i, and the ECG data used by the cloud for training, respectively. The weight vector (w) of each UEN is denoted by w, indicating the parameters of the local AI model. With the increasing prediction error, the value of the loss function rises. Thereby, during the communication rounds in the learning phase, the constraint here is to ensure consistent weight parameters of the local AI models upon receiving the updated model from the server for learning convergence. The UENs and the server can utilize the global model for local decision-making without sharing raw ECG samples of each subject.

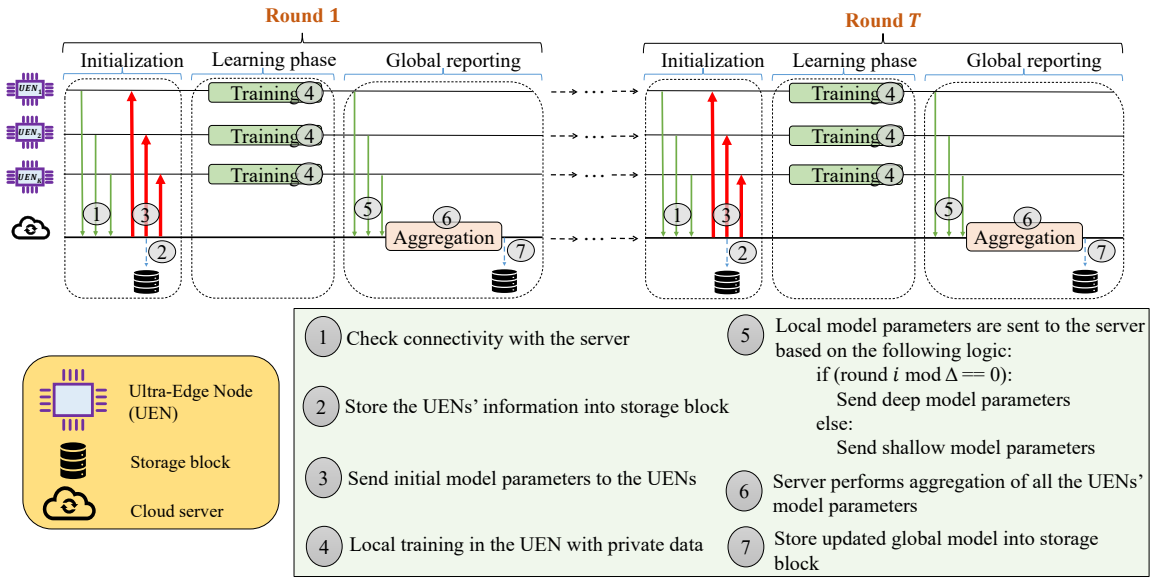


Figure 6.2: Ultra-edge Node (UENs)-based Distributed System Design.

In summary, the research challenge is to devise a system so that each UEN is competent in training a distributed global model to update its local model by adopting its own private raw ECG data. Each UEN can also broadcast its updated local model to the neighboring UENs and the cloud server for asynchronous updates of the global model. This collaborative learning process will continue as long as the loss function is not minimized and the global model accuracy reaches a pre-defined performance threshold. Hence, in this paper, the primary research challenge is to develop such a decentralized learning architecture that will ensure effective decision-making (i.e., arrhythmia detection) with enhanced data privacy and minimize network overhead.

6.4 System Design and Proposed Asynchronous Federated Learning-Based Algorithm

In this section, firstly, we present our envisioned asynchronous federated learning model for privacy-preserving arrhythmia screening in Fig. 6.2. After that, we describe the algorithm of the proposed method that contains Algorithms 8 and 9. The global and local AI models, constructed at the cloud and UENs, are facilitated by a customized lightweight CNN model. The deep learning model (referred to as Deep Learning-Based Lightweight Arrhythmia Classification, DL-LAC) was conceptualized in our earlier work in [117] for a centralized arrhythmia prediction at the ultra-edge sensors. The adopted lightweight AI model which will be deployed at the UENs for ECG analysis accepts a single-lead raw ECG heartbeat as input, represented as $X = [x_1, x_2, x_3, \ldots x_n]$, and outputs predicted class labels $Y =$

$[y_1, y_2, y_3, \ldots y_k]$. Here, $y_k \in \{0, 1\}$, which indicates whether that particular heartbeat belongs to k_{th} class or not.

In Fig. 6.2, we depict the fundamental steps of the AI-enabled asynchronous federated learning-based system model. As the AI model, we utilize the automated deep learning-based one-dimensional (1-D) custom lightweight CNN model discussed in the paper [117] as it does not necessitate any noise-filtering and manual feature extraction and it detects unique patterns automatically from the raw single-lead ECG signal. The Async-FL architecture essentially consists of 7 stages. There are three components in the system model: Ultra-Edge Nodes or the UENs, storage block, and cloud server. Firstly, during the initialization phase, at time t, each UEN checks the connectivity with the cloud server. Upon getting the connectivity with the server, the server stores the UEN's information in the storage block and sends the initially trained AI model to the UEN so that it can update its local AI model. After the UEN obtains the AI model, in the learning phase, it utilizes the learning process model using the private ECG data acquired at the UEN. In the global reporting phase, the local model parameters (shallow or deep depending) of each UEN are delivered to the server for aggregation. Later on, the server aggregates and updates the global model with all the information obtained from the UENs. The newly updated global model is then stored into the storage block for future use of the UENs. This overall process can be performed T times at each UEN.

Algorithm 8 depicts the algorithm that takes place at the remote cloud server to update the global model. As input, the algorithm takes the set of local model parameters or weights of all UENs, denoted as W, and it delivers the updated global model (M_g). Steps 1 and 2 of this algorithm are the initialization phases. In steps 3 to 6, the transmitted local model parameters (W) of all UENs are aggregated to update the global model, M_g. The learning performance of the model is accessed in step 5, and during every iteration, it is compared with the previously defined loss threshold (ξ). Therefore, the overall learning process at the remote cloud server takes place as long as the current loss value is higher than ξ.

Algorithm 8: Local models' aggregation at the remote server

Input : W (Collection of local model parameters/weights of all UENs)
Output: M_g (global model)

1 $\xi \leftarrow$ define the minimum loss threshold
2 $M_g \leftarrow$ load the existing global model from storage
3 **while** *($curr_{loss} > \xi$)* **do**
4 | $W \leftarrow$ load local model parameters of all UENs update M_g by aggregating all the local parameters (W)
5 | $curr_{loss} \leftarrow$ compute current loss of M_g
6 **end**
7 return M_g

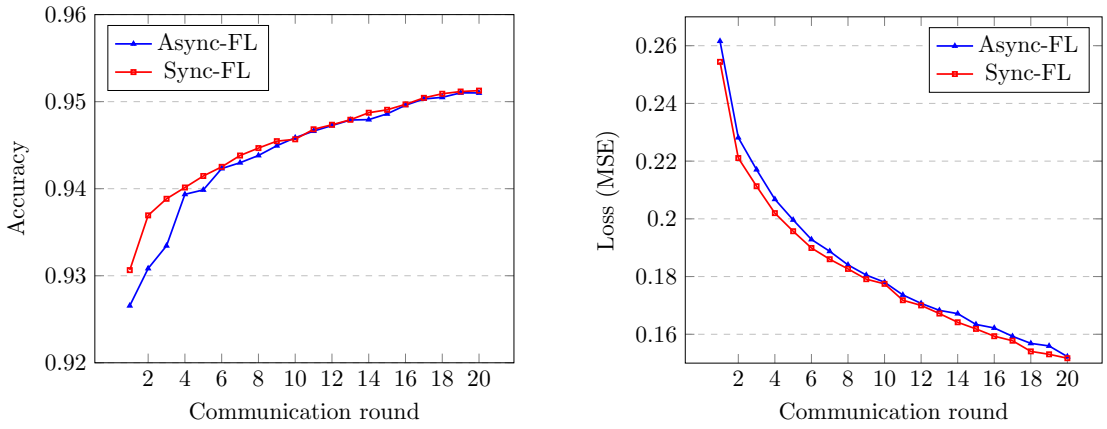
Algorithm 9: Learning at each UEN

Input : j (the current UEN), M_j (local model of UEN j), Δ (time-round), T (total number of communication rounds), α (deep parameter ratio, $0 < \alpha < 1$)

1 initialize M_j with the primarily trained AI model
2 $\delta \leftarrow$ initialize ECG data size threshold
3 initialize $timestep_j$ and $block_j$
4 $X_j \leftarrow$ obtain new data
5 $X_j \leftarrow$ validate the size of X_j by comparing with δ
6 **for** *(t=1 to T)* **do**
7 **if** *(t mod Δ)=0* **then**
8 t is assigned to $timestep_j$
9 $W_j \leftarrow$ extract local weights of α from M_j
10 deliver W_j, $block_j$, and $timestep_j$ to the server
11 **else**
12 $W_j \leftarrow$ extract local weights of $(1 - \alpha)$ from M_j
13 deliver W_j to the server
14 **end**
15 $M_j \leftarrow$ obtain updated model from the server
16 $block_j \leftarrow$ store global model state and data access information of time t
17 **end**
18 delete X_j from storage

The algorithm that runs on the UEN's side is manifested in Algorithm 9. We have considered the algorithm that runs on one UEN (referred to as UEN j) in the Algorithm 9. The algorithm initiates with the inputs; these inputs' details are demonstrated in the algorithm's input fragment. First, in step 1, the local model of UEN j, M_j is initialized with the initially trained AI model. In steps 2 and 3, the initialization of different necessary parameters takes place. Step 4 loads the new private data, X_j, and its size by comparing it with the ECG data size threshold δ in step 5.

The iteration starting at step 6 denotes that the overall training process at each UEN occurs from iterations $t = 1$ to $t = T$. The parameter T can be considered as the number of communication rounds during which the whole process at each UEN is executed. After Δ time rounds, when this condition is satisfied in step 7, time t is stored in $timestep_j$ list. Also, the local weight (W_j) with deep parameters and relevant access and time-step information ($block_j$, $timestep_j$) are communicated to the cloud in steps 9 and 10. Here, α denotes the deep parameter exchange ratio, indicating the deep parameter ratio contributing to the deep exchange. The parameter $timestep_j$ holds the iteration information when the deep parameter exchange with the cloud takes place. In steps 11 to 14, the local weight (W_j) shallow parameters of the $(1 - \alpha)$ ratio from the local model (M_j) are transferred to the



(a) Learning accuracy

(b) Value of loss function

Figure 6.3: The performance comparison of two federated learning architectures during the learning/training phase over varying communication rounds (employing DS1).

server. Step 15 updates the local model of UEN j by using the aggregated model obtained from the server. In step 16, all the global model states and information regarding time-step t is stored in the $block_j$ for future reference. Finally, in step 18, after training for T rounds, the utilized data X_j are permanently removed from the cache to enhance the security of the user's data.

6.5 Performance Evaluation

6.5.1 Data Preparation

We evaluated the generalization potential of the adopted FL-based distributed cardiac monitoring architecture employing four publicly available arrhythmia detection datasets. We utilized the MIT-BIH Supraventricular Arrhythmia database, referred to as DS1 [106] for the learning/training purpose. Then, the adopted techniques were extensively tested using three other public datasets from the Physionet repositories, such as the MIT-BIH Arrhythmia database, INCART 12-lead Arrhythmia database, and Sudden Cardiac Death Holter database, referred to as DS2 [107], DS3 [43], and DS4 [108], respectively. The recommendation of the Association for the Advancement of Medical Instrumentation (AAMI) is utilized for the arrhythmia classification task. We have considered four classes of heartbeats, namely N, S, V, and F, in developing this multi-class classification task, which represents normal, supraventricular ectopic, ventricular ectopic, and fusion beats, respectively [85].

Table 6.1: Classification performance of adopted FL architectures over varying number of Ultra-Edge Nodes (UENs) using three different test datasets.

Method	Dataset	Metrices	Number of Ultra-Edge nodes				
			2	4	6	8	10
Sync-FL	DS2	Accuracy	**0.88874**	**0.88979**	0.88969	0.87833	0.87717
		Precision	0.87830	0.87949	0.87440	0.86868	0.87193
		F1-score	0.88061	0.88151	0.88122	0.87185	0.87198
	DS3	Accuracy	**0.89293**	**0.88947**	0.88612	0.89517	0.86304
		Precision	0.88463	0.86772	0.85985	0.88153	0.83830
		F1-score	0.86715	0.86181	0.86750	0.86573	0.84906
	DS4	Accuracy	**0.76714**	**0.77717**	**0.75305**	0.73222	0.74874
		Precision	0.94192	0.95222	0.94839	0.95070	0.95052
		F1-score	0.61370	0.78300	0.83498	0.82060	0.79105
Async-FL	DS2	Accuracy	0.86853	0.87911	**0.89147**	**0.87903**	**0.87837**
		Precision	0.87277	0.87818	0.88315	0.87529	0.87749
		F1-score	0.86915	0.87554	0.88723	0.87520	0.87505
	DS3	Accuracy	0.87094	0.88612	**0.89478**	**0.89524**	**0.89892**
		Precision	0.84107	0.85985	0.88966	0.88134	0.89130
		F1-score	0.85204	0.86750	0.86772	0.86862	0.87387
	DS4	Accuracy	0.74033	0.73773	0.73240	**0.76935**	**0.76287**
		Precision	0.95062	0.94869	0.94417	0.95335	0.94796
		F1-score	0.81196	0.82400	0.73588	0.84454	0.69541

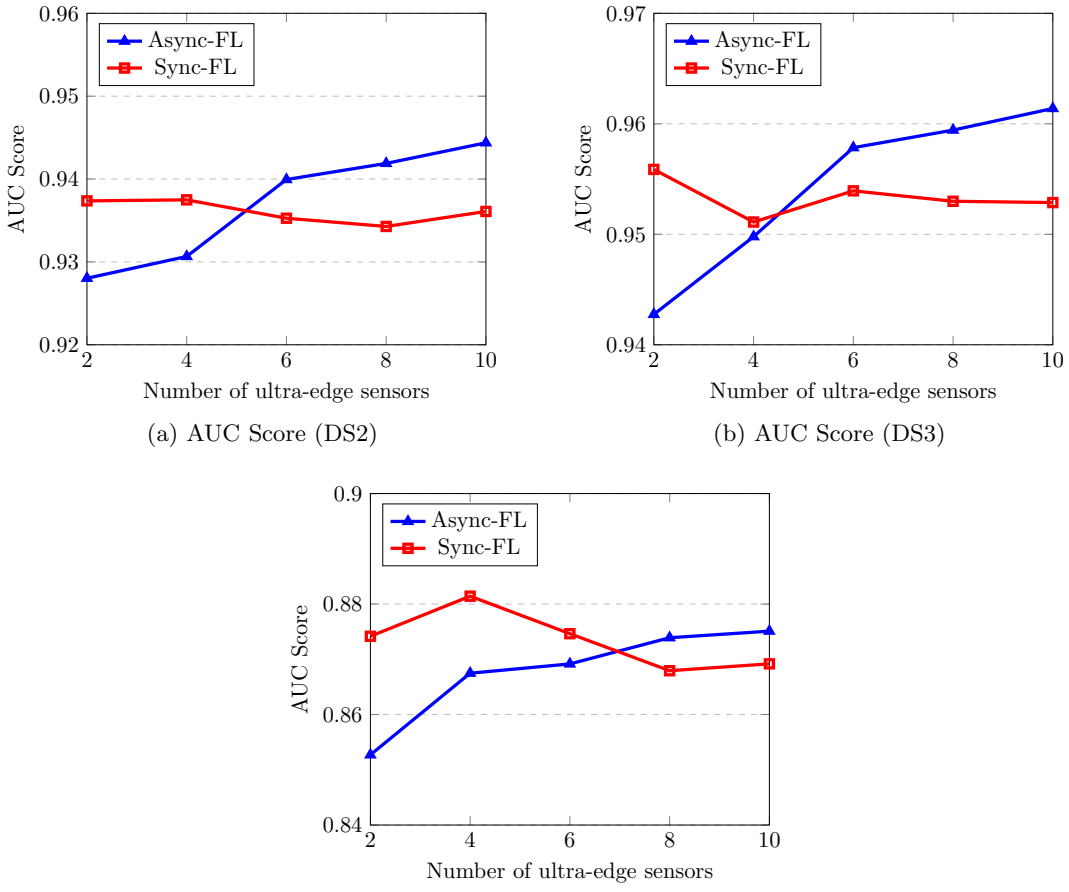
6.5.2 Simulation Setup

We have evaluated the performance of the FL architectures on four different classification performance metrics such as accuracy, weighted precision, weighted recall, and area under the receiver operating characteristic curve (AUC score). To identify the architectures' time and memory requirements, we have also determined the required time in seconds and memory requirement percentage for varying numbers of UENs. In the experimental analysis, the number of UENs contains different values such that $UEN \in \{2, 4, 6, 8, 10\}$.

Independent of the underlying AI model, two variations of the proposed federated learning architecture based on a custom lightweight CNN model, are simulated that are referred to as Async-FL and Sync-FL, respectively, for brevity. Also, we define the number of communication rounds/iterations and time-round to evaluate our proposal's performance. A communication round or iteration refers to the number of times the whole process runs (e.g., the local training for the UENs and exchanging parameters). In terms of the Async-FL, a time-round consists of multiple iterations/communication rounds, after which the cloud is updated with the deep weights. Note that the concept of time-round is exclusive to the Async-FL only, and in terms of the Sync-FL, deep model exchange occurs in every communication rounds. In other words, an iteration is an atomic unit relative to the time-round. For instance, in the Async-FL architecture, if the number of iterations is 20 and the number of time rounds is 5, then for the 5th, 10th, 15th, and 20th iteration, the deep model update is triggered in our proposed approach while the shallow model parameter update takes place during the other iterations. The number of iterations or communication rounds

is set to 20 for both FL variations. For the Async-FL, the value of time-round is 5. In the CNN model, the epoch and batch-size are fixed to 10 and 5000, respectively. The Rectified Linear Unit (ReLU) is employed as the activation function. At the same time, Adam (Adaptive Moment Estimation) is used as the optimizer to select and regularly update a preconditioned stochastic gradient descent. The adopted CNN model is adopted from the proposed DL-LAC model presented in paper [117]. It consists of three convolution layers, followed by three fully connected layers and the output layer.

6.5.3 Results and Discussion



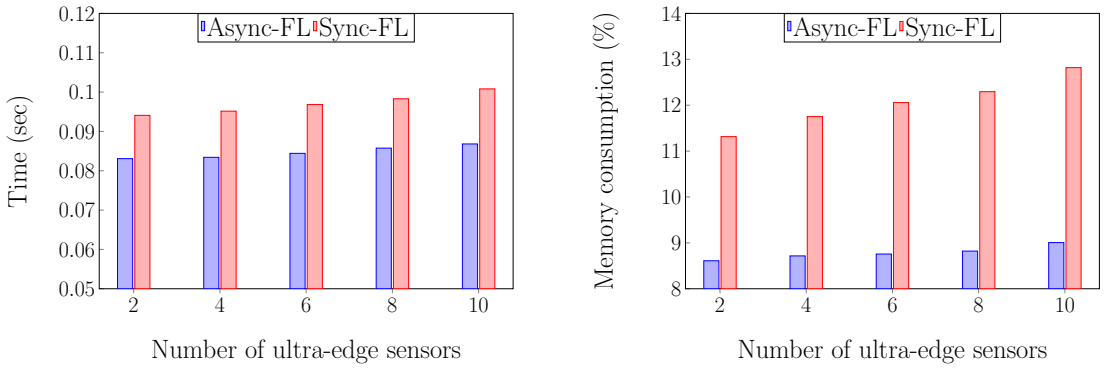
(a) AUC Score (DS2)

(b) AUC Score (DS3)

(c) AUC Score (DS4)

Figure 6.4: AUC score values acquired in the inference phase of the Sync-FL and Async-FL methods using different test datasets (i.e., DS2, DS3, and DS4).

Firstly, employing the DS1, we train the synchronous and asynchronous federated learning architectures, referred to as Sync-FL and Async-FL. Fig. 6.3 compares the learning accuracy and loss of the Sync-FL and Async-FL architectures over a varying number of it-



(a) Time required for different devices (in seconds)

(b) Memory consumption (%) by different methods for different devices

Figure 6.5: Required execution time and memory consumption for varying number of UENs (inference phase).

erations/communication rounds. In Fig. 6.3 we have demonstrated the mean accuracy and loss curve for different numbers of UENs (i.e., 2, 4, 6, 8, and 10). Note that the UENs start the first communication rounds with a centrally trained AI model, hence even in the first iteration/round, it obtains more than 90% accuracy. Notice from Fig. 6.3a that both models' learning accuracy reaches almost 95% over time. Both the variations of our proposal gain higher accuracies with the increasing number of communication rounds. During the initial rounds, the Sync-FL showed better performance than the Async-FL; however, both architectures obtain almost identical learning performance as the communication rounds progress. A similar trend is also illustrated in the learning loss, manifested in terms of the mean squared error (MSE), in Fig. 6.3b.

After the learning phase, we investigate the FL architectures generalization ability in the inference phase on three datasets (e.g., DS2, DS3, and DS4). Table 6.1 demonstrates the classification performance of adopted FL architectures over a varying number of UENs using the different test datasets. The results indicate that both the adopted FL architectures achieved encouraging classification accuracy in the test datasets, especially in DS2 and DS3. The Sync-FL shows superior performance than Async-FL when the number of UEN is relatively low. However, as the number of UEN increased, the classification performance showed better results in the case of the Async-FL for each of the adopted test datasets. A similar trend is also observed in terms of the other two performance metrics (i.e., precision and f1-score).

Next, in Fig. 6.4 we explore the AUC score to identify the area under the ROC curve by using DS2, DS3, and DS4 as the test dataset of the inference phase. Both FL architectures obtained robust AUC scores for all three adopted test datasets. A similar trend to the accuracy values demonstrated in Table 6.1 is also observed in the AUC score values as the

Sync-FL obtains better scores when the number of UEN is low. Whereas, for the increasing number of UENs, the Async-FL's AUC score values are superior to the Sync-FL. The results prove that even though the Async-FL architecture transmits learned deep model parameters after time-round (Δ), as the number of users (i.e., UEN) grows higher, the classification performance becomes more robust. The encouraging arrhythmia detection performance ensures the proposed Async-FL's viability for deploying in an online collaborative learning paradigm.

In the next simulation setup in Fig. 6.5, we compare the required execution time and memory consumption for the inference phase for the proposed Sync-FL and Async-FL architectures. For a varying number of UENs, the average execution time (in seconds) needed in the inference phase by both FL paradigms is shown in Fig. 6.5a, and the memory consumption (in percentage) is displayed in Fig. 6.5b. Although both systems can generate results in a low amount of time and memory requirements, due to regulating the deep model exchange ratio in the Async-FL technique, the execution time and memory consumption are lower than the counter-part Sync-FL approach. The low time and memory demand of the Async-FL architecture ensure its suitability to get used in a collaborative online learning approach with the resource-constrained ultra-edge IoT nodes.

6.6 Summary

In this paper, we have proposed an asynchronously updating federated learning architecture (Async-FL) for mobile and deployable Ultra-Edge Nodes (UENs) to build decentralized and collaborative arrhythmia detection without the need for direct ECG data exchange with the cloud. We employ raw single-lead ECG data to design the distributed federated learning architecture, preserve patient data privacy, and mitigate the network overhead. Extensive experimental results demonstrated the viability of the proposed Async-FL in terms of lightweight operation (e.g., low execution time and memory consumption) while attaining a considerable arrhythmia detection accuracy. Our proposal also leads to lower network overhead (i.e., bandwidth consumption, time, and memory requirements) for an increasing number of UENs. As the demand for remote patient monitoring is escalating with the emergence of pandemics such as the novel coronavirus, this particular ECG monitoring use-case using the asynchronous federated learning paradigm can lead the way to implement the envisioned future generation smart and remote health monitoring system at a mass scale.

Chapter 7

Conclusions and Future Works

This chapter summarizes the contributions of the dissertation work and manifests potential future research directions.

7.1 Contributions

In this thesis, we investigate the development of different lightweight artificial intelligence-based models, specifically deep learning-based models, so that the logic can be integrated with the resource-constrained ultra-edge IoT nodes for localized decision-making in diverse challenges associated with remote health monitoring. Along with the rapid advancements of enabling technologies such as AI, IoT, and 5G, B5G networks, the demand for remote health monitoring is also escalating on a regular basis. Especially in this time of urgency of the pandemics such as the COVID-19, remote health monitoring has become even more demanding. Therefore, to address this growing demand, we have first identified why the traditional cloud analytics-based architectures are not suitable for remote health monitoring at a mass scale. We shed light on the necessity of moving intelligence towards the ultra-edge IoT nodes. The ultra-edge IoT nodes are resource-constrained; hence, they need lightweight AI models for decision making and data delivery. In this goal, we propose different custom deep learning-based models to achieve lightweight and efficient AI solutions and compare the results with typical machine learning models. We mainly explore the implications of supervised learning tasks from three considered layers/planes of IoT, namely, sensing, communication, and computing layer. The empirical results are obtained by analyzing publicly available real datasets that have been adopted for evaluating the proposed methodologies on several performance indicators.

Towards conducting the research experiments, there were several challenges associated with each of the research works that future researchers can anticipate in the future. In chapter 3, for MCG denoising use-case, we observed a lack of MCG dataset in the literature,

Table 7.1: Summary of the contributions in sensing layer of IoT

Chapter no.	**Chapter 3**
Considered use-case	Noise-removal from the MCG signal to obtain the ECG for continuous monitoring of cardiovascular activities
Task type	Supervised learning (regression)
Dataset	Publicly available real datasets from PhysioNet [42–44]
Utilized AI framework	Reservoir Computing (RC)
Method used for comparison	Deep learning, Moving average technique
Performance indicators	RMSE, PSD, and Time-Memory requirments

Table 7.2: Summary of the contributions in the communication layer of IoT

Chapter no.	**Chapter 4**
Considered use-case	Estimate channel conditions in the multi-band relay of B5G networks for efficient delivery of medical data
Task type	Supervised learning (regression)
Dataset	Publicly available real datasets from CRAWDAD [78–80]
Utilized AI framework	Convolutional Neural Network (CNN)
Method used for comparison	Linear Regression (LR), Auto Regression (AR), and Artificial Neural Network (ANN)
Performance indicators	Average RMSE, Processing time, Memory requirements, Network throughput

Table 7.3: Summary of the contributions in computing layer of IoT

Chapter no.	**Chapter 5**	**Chapter 6**
Considered use-case	Develop lightweight AI model to detect arrhythmia, which can be deployed to the ultra-edge IoT nodes	Design a decentralized, collaborative, and online learning architecture to deploy the lightweight AI model that can detect arrhythmia
Task type	Supervised learning (classification)	
Dataset	Publicly available real dataset from PhysioNet [43, 106–108]	
Utilized AI framework	Convolutional Neural Network (CNN)	CNN model with an Asynchronous Federated Learning
Method used for comparison	K-Nearest Neighbor (KNN) and Random Forest (RF)	Synchronous federated learning
Performance indicators	Accuracy, Precision, F1-score, ROC Curve, AUC score, Processing time, Memory requirements	Accuracy, Precision, F1-score, Loss (MSE), AUC score, Processing time, Memory requirements

which is open for public use. Thereby, we adopted synthesize MCG data to analyze data in our experiments. Thus collection and documentation of MCG data can be a future research direction to exploit a real dataset towards the research outcomes. Although we have used multiple public datasets in chapter 4, 5, and 6, before actually implementing them in real-life health monitoring uses, we need to conduct more simulations on data with various distributions. Furthermore, in the experiments conducted at the computing layers of IoT, the arrhythmia datasets we employed comprise ECG data with heartbeat segmentation; hence, we could not incorporate the cost needed in a typical segmentation phase. Thereby, future researchers may encounter a lack of clinically-graded completely raw ECG data (without heartbeat segmentation) for the arrhythmia detection task.

The main contributions and adopted methodologies of the considered use-cases in the three IoT planes are listed in Table 7.1, 7.2, and 7.3, respectively indicating the contributions in the sensing, communication, and computing layers. This thesis can be considered a proof-of-concept for the next generation of smart and connected remote healthcare with the help of distributed in the ultra-edge IoT node utilizing lightweight artificial intelligence models for decision-making with enhanced privacy and AI-aided B5G relay networks for efficient health data transmission.

7.2 Future Directions

In this sub-section, we conclude the thesis by shedding light on some possible future research directions:

- **Collection of real MCG data:** Due to the high cost of the TMR-based MTJ sensors, we could not collect actual MCG data for our experiments. So, we adopted synthetic MCG data for the analysis. However, before incorporating this technique in clinical trials, we have to ensure that the investigations are conducted on real collected datasets. Thereby, future research can be conducted on collecting MCG data from a diverse set of populations and then perform the analysis again to verify the consistency of the results obtained with our work.

- **Detection of Arrhythmia from MCG:** In the sensing layer of IoT, we analyzed the MCG signal to obtain the ECG by noise-filtering using the RC method, and in the computing layer, we picked the use-case of detecting arrhythmia by analyzing ECG data. An exciting future research direction can be to connect these two use-cases from sensing and computing layers by analyzing the viability of using the MCG signal obtained from the ultra-edge IoT nodes to detect arrhythmia.

- **Reinforcement Learning for optimal channel selection:** In the communication layer of IoT, we have proposed a deep learning-based robust channel selection

method for efficient massive IoT data transmission. However, the deep learning-based systems need a lot of historical training data, which can be a challenge in some scenarios. Thereby, an interesting research direction can be to analyze the viability of Reinforcement Learning techniques in selecting the optimal or most suitable channels in multi-band channel selection tasks. The reinforcement learning algorithm is an independent, self-teaching system suitable for dynamic situations, making it a potential solution for network condition prediction or channel selection.

- **Exploration of other types of cardiac irregularities and severity level:** In the computing layer of IoT, we considered four classes of heartbeats to detect arrhythmia. In future research, we can extend this work to consider a more broad range of heartbeats along with the already used ones. Another exciting research direction could be to use one lightweight AI model to detect irregular heartbeats and severity level of cardiac abnormalities.

- **Extending the concept of the ultra-edge IoT in other health monitoring use-cases:** Ultra-high sensitive TMR-based MTJ sensors can be used to measure both heart's MCG, and brain's MagnetoEncephaloGraphy (MEG) signals at room temperature. In this thesis, we investigated the viability of MCG in cardiac monitoring using lightweight AI models with ultra-edge IoT nodes. This research can be extended towards brain activity monitoring by analyzing the MEG signal and exploring its relationship with the brain signals such as EEG.

Bibliography

[1] Z. M. Fadlullah, Y. Kawamoto, H. Nishiyama, N. Kato, N. Egashira, K. Yano, and T. Kumagai, "Multi-hop wireless transmission in multi-band WLAN systems: Proposal and future perspective," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 108–113, Feb. 2019.

[2] A. AAMI and A. EC57, "(R) 2008-Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," 2008.

[3] L. Greco, G. Percannella, P. Ritrovato, F. Tortorella, and M. Vento, "Trends in IoT based solutions for health care: Moving AI to the edge," *Pattern Recognition Letters*, vol. 135, pp. 346–353, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865520301884

[4] S. Sakib, T. Tazrin, M. M. Fouda, Z. M. Fadlullah, and M. Guizani, "DL-CRC: Deep Learning-Based Chest Radiograph Classification for COVID-19 Detection: A Novel Approach," *IEEE Access*, vol. 8, pp. 171 575–171 589, 2020, doi: 10.1109/ACCESS.2020.3025010.

[5] A. Haleem and M. Javaid, "Medical 4.0 and its role in healthcare during COVID-19 pandemic: A review," *Journal of Industrial Integration and Management*, vol. 5, no. 4, 2020.

[6] B. A. Jnr, "Use of telemedicine and virtual care for remote treatment in response to COVID-19 pandemic," *Journal of Medical Systems*, vol. 44, no. 7, pp. 1–9, 2020.

[7] D. Laura, "20 healthcare moves from Amazon, Google, Microsoft & Apple in 2020," Becker's Hospital Review, Jan. 2021. [Online]. Available: https://www.beckershospitalreview.com/healthcare-information-technology/20-healthcare-moves-from-amazon-google-microsoft-apple-in-2020.html

[8] Y. Zhang, G. Chen, H. Du, X. Yuan, M. Kadoch, and M. Cheriet, "Real-Time Remote Health Monitoring System Driven by 5G MEC-IoT," *Electronics*, vol. 9, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/11/1753

[9] A. Ahad, M. Tahir, M. Aman Sheikh, K. I. Ahmed, A. Mughees, and A. Numani, "Technologies trend towards 5G network for smart health-care using IoT: A review," *Sensors*, vol. 20, no. 14, p. 4047, 2020.

[10] S. Qazi and K. Raza, "Chapter 4 - Smart biosensors for an efficient point of care (PoC) health management," in *Smart Biosensors in Medical Care*, ser. Advances in ubiquitous sensing applications for healthcare, J. Chaki, N. Dey, and D. De, Eds. Academic Press, 2020, pp. 65–85. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128207819000048

[11] A. Mohsen, M. Al-Mahdawi, M. M. Fouda, M. Oogane, Y. Ando, and Z. M. Fadlullah, "AI Aided Noise Processing of Spintronic Based IoT Sensor for Magnetocardiography Application," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, Dublin, Ireland, 7-11 Jun. 2020, doi: 10.1109/ICC40277.2020.9148617.

[12] X. Liu, K. H. Lam, K. Zhu, C. Zheng, X. Li, Y. Du, C. Liu, and P. W. T. Pong, "Overview of Spintronic Sensors With Internet of Things for Smart Living," *IEEE Transactions on Magnetics*, vol. 55, no. 11, pp. 1–22, Nov. 2019.

[13] K. Fujiwara, M. Oogane, A. Kanno, M. Imada, J. Jono, T. Terauchi, T. Okuno, Y. Aritomi, M. Morikawa, M. Tsuchida, N. Nakasato, and Y. Ando, "Magnetocardiography and magnetoencephalography measurements at room temperature using tunnel magneto-resistance sensors," *Applied Physics Express*, vol. 11, no. 2, Jan. 2018, doi: 10.7567/apex.11.023001.

[14] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, and P. Reynolds, "Cloud computing: A new business paradigm for biomedical information sharing," *Journal of Biomedical Informatics*, vol. 43, no. 2, pp. 342–353, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046409001154

[15] Y. Guo, M. Kuo, and T. Sahama, "Cloud computing for healthcare research information sharing," in *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, 2012, pp. 889–894.

[16] S. Sakib, T. Tazrin, M. M. Fouda, Z. M. Fadlullah, and N. Nasser, "An Efficient and Lightweight Predictive Channel Assignment Scheme for Multiband B5G-Enabled Massive IoT: A Deep Learning Approach," *IEEE Internet of Things Journal*, vol. 8, no. 7, pp. 5285–5297, 2021, doi: 10.1109/JIOT.2020.3032516.

[17] L. Chettri and R. Bera, "A Comprehensive Survey on Internet of Things (IoT) Toward 5G Wireless Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 16–32, 2020.

[18] W. Anani, A. Ouda, and A. Hamou, "A Survey Of Wireless Communications for IoT Echo-Systems," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, 2019, pp. 1–6.

[19] M. Emu and S. Sakib, "Species Identification using DNA Barcode Sequences through Supervised Learning Methods," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, 2019, pp. 1–6, doi: 10.1109/ECACE.2019.8679166.

[20] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A State-of-the-Art Survey on Deep Learning Theory and Architectures," *Electronics*, vol. 8, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/3/292

[21] M. A. Boyacioglu, Y. Kara, and Ömer Kaan Baykan, "Predicting bank financial failures using neural networks, support vector machines and multivariate statistical methods: A comparative analysis in the sample of savings deposit insurance fund (SDIF) transferred banks in Turkey," *Expert Systems with Applications*, vol. 36, no. 2, Part 2, pp. 3355–3366, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S095741740800078X

[22] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, Antalya, Turkey, Aug. 2017, pp. 1–6.

[23] J. Gu, Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, L. Wang, G. Wang, J. Cai, and T. Chen, "Recent Advances in Convolutional Neural Networks," 2017.

[24] R. Z. Cabada, H. R. Rangel, M. L. B. Estrada, and H. M. C. Lopez, "Hyperparameter optimization in CNN for learning-centered emotion recognition for intelligent tutoring systems," *Soft Computing*, vol. 24, no. 10, pp. 7593–7602, 2020.

[25] "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013709000173

[26] M. Lukoševičius, *A Practical Guide to Applying Echo State Networks.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 659–686. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_36

[27] B. J. Grzyb, E. Chinellato, G. M. Wojcik, and W. A. Kaminski, "Which model to use for the Liquid State Machine?" in *2009 International Joint Conference on Neural Networks*, Atlanta, GA, USA, 14–19 Jun. 2009, pp. 1018–1024.

[28] G. Tanaka *et al.*, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, Jul. 2019.

[29] D. Elsarraj, M. A. Qisi, A. Rodan, N. Obeid, A. Sharieh, and H. Faris, "Demystifying echo state network with deterministic simple topologies," *Int. J. Comput. Sci. Eng.*, vol. 19, pp. 407–417, 2019.

[30] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," 2017.

[31] Y. F. Suprunenko, P. T. Clemson, and A. Stefanovska, "Chronotaxic Systems: A New Class of Self-Sustained Nonautonomous Oscillators," *Physical Review Letters*, vol. 111, no. 2, p. 024101, Jul. 2013.

[32] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

[33] H. Jaeger and H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[34] A. Hirohata, K. Yamada, Y. Nakatani, L. Prejbeanu, B. Diény, P. Pirro, and B. Hillebrands, "Review on spintronics: Principles and device applications," *Journal of Magnetism and Magnetic Materials*, vol. 509, p. 166711, Sep. 2020.

[35] S. Sakib, M. M. Fouda, Z. M. Fadlullah, and N. Nasser, "Migrating Intelligence from Cloud to Ultra-Edge Smart IoT Sensor Based on Deep Learning: An Arrhythmia Monitoring Use-Case," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, Jun. 2020, pp. 595–600, doi: 10.1109/IWCMC48107.2020.9148134.

[36] F. N. Hooge, T. G. M. Kleinpenning, and L. K. J. Vandamme, "Experimental Studies on 1/f Noise," *Reports on Progress in Physics*, vol. 44, no. 5, 1981, doi: 10.1088/0034-4885/44/5/001.

[37] P. Wisniowski, J. M. Almeida, and P. P. Freitas, "1/f Magnetic Noise Dependence on Free Layer Thickness in Hysteresis Free MgO Magnetic Tunnel Junctions," *IEEE Transactions on Magnetics*, vol. 44, no. 11, pp. 2551–2553, Nov. 2008.

[38] Z. Q. Lei, G. J. Li, W. F. Egelhoff, P. T. Lai, and P. W. T. Pong, "Review of Noise Sources in Magnetic Tunnel Junction Sensors," *IEEE Transactions on Magnetics*, vol. 47, no. 3, pp. 602–612, Mar. 2011.

[39] N. Schaetti, M. Salomon, and R. Couturier, "Echo State Networks-Based Reservoir Computing for MNIST Handwritten Digits Recognition," in *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*, 2016, pp. 484–491, doi: 10.1109/CSE-EUC-DCABES.2016.229.

[40] Z. Tong and G. Tanaka, "Reservoir Computing with Untrained Convolutional Neural Networks for Image Recognition," in *2018 24th International Conference on Pattern Recognition (ICPR)*, Aug. 2018, pp. 1289–1294.

[41] K. Zhang, G. Chuai, W. Gao, X. Liu, S. Maimaiti, and Z. Si, "A new method for traffic forecasting in urban wireless communication network," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, Mar. 2019, Art. no. 66. [Online]. Available: https://doi.org/10.1186/s13638-019-1392-6

[42] R. Bousseljot, D. Kreiseler, and A. Schnabel, "Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet," *Biomedical Engineering/Biomedizinische Technik*, vol. 40, no. s1, pp. 317–318, 1995.

[43] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. E215–220, Jun. 2000.

[44] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, Jun. 2018, doi: 10.1109/ICHI.2018.00092.

[45] S. Sakib, T. Tazrin, M. M. Fouda, Z. M. Fadlullah, and N. Nasser, "A Deep Learning Method for Predictive Channel Assignment in Beyond 5G Networks," *IEEE Network*, vol. 35, no. 1, pp. 266–272, 2021, doi: 10.1109/MNET.011.2000301.

[46] A. Arfaoui, S. Cherkaoui, A. Kribeche, and S. M. Senouci, "Context-Aware Adaptive Remote Access for IoT Applications," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 786–799, Jan. 2020.

[47] S. Oteafy and H. Hassanein, "Leveraging Tactile Internet Cognizance and Operation via IoT and Edge Technologies," *Proceedings of the IEEE*, vol. 107, no. 2, pp. 364–375, Feb. 2019.

[48] Y. Han, B. D. Rao, and J. Lee, "Massive Uncoordinated Access With Massive MIMO: A Dictionary Learning Approach," *IEEE Transactions on Wireless Communications*, vol. 19, no. 2, pp. 1320–1332, Feb. 2020.

[49] J. Lianghai, B. Han, M. Liu, and H. Schotten, "Applying Device-to-Device Communication to Enhance IoT Services," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 85–91, Jul. 2017.

[50] X. Chen, D. W. K. Ng, W. Yu, E. G. Larsson, N. Al-Dhahir, and R. Schober, "Massive Access for 5G and Beyond," *arXiv preprint arXiv:2002.03491*, Feb. 2020.

[51] A. Moubayed, A. Shami, and H. Lutfiyya, "Wireless Resource Virtualization With Device-to-Device Communication Underlaying LTE Network," *IEEE Transactions on Broadcasting*, vol. 61, no. 4, pp. 734–740, 2015.

[52] A. Ettefagh, M. Kuhn, I. Hammerstrom, and A. Wittneben, "On the range performance of decode-and-forward relays in IEEE 802.11 WLANs," in *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, Helsinki, Finland, Sep. 2006.

[53] P. Zhang, J. Yuan, J. Chen, J. Wang, and J. Yang, "Analyzing amplify-and-forward and decode-and-forward cooperative strategies in Wyner's channel model," in *2009 IEEE Wireless Communications and Networking Conference*, Budapest, Hungary, Apr. 2009.

[54] A. Khina, O. Ordentlich, U. Erez, Y. Kochman, and G. W. Wornell, "Decode-and-forward for the Gaussian relay channel via standard AWGN coding and decoding," in *2012 IEEE Information Theory Workshop*, Lausanne, Switzerland, Sep. 2012.

[55] N. Egashira, K. Yano, S. Tsukamoto, J. Webber, M. Sutoh, Y. Amezawa, and T. Kumagai, "Integrated synchronization scheme for WLAN systems employing multiband simultaneous transmission," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, San Francisco, CA, USA, Mar. 2017.

[56] F. Gholami, H. Meghdadi, and A. Shahzadi, "Throughput Analysis for Decode-and-Forward Relaying Protocol with Wireless Energy Harvesting and Information Processing," in *2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC)*, Dec. 2018, pp. 132–135.

[57] A. Chaaban and A. Sezgin, "Multi-hop relaying: An end-to-end delay analysis," *IEEE Transactions on Wireless Communications*, vol. 15, no. 4, pp. 2552–2561, Apr. 2016.

[58] G. Cerar, H. Yetgin, M. Mohorčič, and C. Fortuna, "Machine Learning for Link Quality Estimation: A Survey," *arXiv preprint arXiv:1812.08856*, Nov. 2019.

[59] A. R. Abdellah, O. A. K. Mahmood, A. Paramonov, and A. Koucheryavy, "IoT traffic prediction using multi-step ahead prediction with neural network," in *2019 11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, Oct. 2019, pp. 1–4.

[60] Y. Kim, P. Wang, and L. Mihaylova, "Structural Recurrent Neural Network for Traffic Speed Prediction," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 5207–5211.

[61] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2595–2621, Jun. 2018.

[62] Y. Lin, X. Dai, L. Li, and F.-Y. Wang, "An efficient deep reinforcement learning model for urban traffic control," *arXiv preprint arXiv:1808.01876*, 2018.

[63] M. Chen, U. Challita, W. Saad, C. Yin, and M. Debbah, "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *arXiv preprint arXiv:1710.02913*, 2017.

[64] M. Nasri and M. Hamdi, "LTE QoS Parameters Prediction Using Multivariate Linear Regression Algorithm," in *2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, Feb. 2019, pp. 145–150.

[65] Y. Zhou and J. Li, "Research of Network Traffic Anomaly Detection Model Based on Multilevel Autoregression," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, Oct. 2019, pp. 380–384.

[66] C. Parera, A. E. C. Redondi, M. Cesana, Q. Liao, and I. Malanchini, "Transfer Learning for Channel Quality Prediction," in *2019 IEEE International Symposium on Measurements Networking (M&N)*, Jul. 2019, pp. 1–6.

[67] M. Iqbal, M. Zahid, D. Habib, and L. John, "Efficient Prediction of Network Traffic for Real-Time Applications," *Journal of Computer Networks and Communications*, vol. 2019, pp. 1–11, Feb. 2019, Art. no. 4067135. [Online]. Available: ttps://doi.org/10.1155/2019/4067135

[68] L. Nie, X. Wang, L. Wan, S. Yu, H. Song, D. Jiang, and K. Zhang, "Network Traffic Prediction Based on Deep Belief Network and Spatiotemporal Compressive Sensing in Wireless Mesh Backbone Networks," *Wireless Communications and Mobile Computing*, vol. 2018, Jan. 2018, Art. no. 1260860. [Online]. Available: https://doi.org/10.1155/2018/1260860

[69] T. Liu and A. E. Cerpa, "Temporal adaptive link quality prediction with online learning," *ACM Transactions on Sensor Networks (TOSN)*, vol. 10, no. 3, pp. 1–41, May 2014.

[70] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," in *IEEE International Conference on Computer Communications (INFOCOM'17)*, Atlanta, GA, USA, May 2017.

[71] L. Liu, B. Yin, S. Zhang, X. Cao, and Y. Cheng, "Deep learning meets wireless network optimization: Identify critical links," *IEEE Transactions on Network Science and Engineering*, Apr. 2018.

[72] W. Rehan, S. Fischer, and M. Rehan, "Machine-learning based channel quality and stability estimation for stream-based multichannel wireless sensor networks," *Sensors Journal*, vol. 16, no. 9, p. 1476, Sep. 2016.

[73] J. D. Herath, A. Seetharam, and A. Ramesh, "A Deep Learning Model for Wireless Channel Quality Prediction," in *IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019.

[74] A. Hanyu, Y. Kawamoto, H. Nishiyama, N. Kato, N. Egashira, K. Yano, and T. Kumagai, "Adaptive Frequency Band and Channel Selection for Simultaneous Receiving and Sending in Multiband Communication," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 460–463, Apr. 2019.

[75] H. Singh, J. Hsu, L. Verma, S. S. Lee, and C. Ngo, "Green operation of multi-band wireless LAN in 60 GHz and 2.4/5 GHz," in *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, Las Vegas, USA, Jan. 2011.

[76] M. Li, S. Salinas, P. Li, X. Huang, Y. Fang, and S. Glisic, "Optimal Scheduling for Multi-Radio Multi-Channel Multi-Hop Cognitive Cellular Networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 1, pp. 139–154, Jan. 2015.

[77] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, p. 1929–1958, Jan. 2014.

[78] R. Parasuraman, S. Caccamo, F. Baberg, and P. Ogren, "CRAWDAD dataset kth/rss," Downloaded from https://crawdad.org/kth/rss/20160105, Feb. 2020.

[79] S. Fu and Y. Zhang, "CRAWDAD dataset due/packet-delivery," Downloaded from https://crawdad.org/due/packet-delivery/20150401, Feb. 2020.

[80] B. Meixner, J. W. Kleinrouweler, and P. Cesar, "4G/LTE Channel Quality Reference Signal Trace Data Set," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: Association for Computing Machinery, Jun. 2018, p. 387–392.

[81] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," *arXiv preprint arXiv:1811.03378*, Nov. 2018.

[82] S. Mendis, P. Puska, and B. Norrving, "Global atlas on cardiovascular disease prevention and control. WHO," *World Heart Federation and World Stroke Organization*, 01 2011.

[83] S. Berrouiguet, M. L. Barrigón, J. L. Castroman, P. Courtet, A. Artés-Rodríguez, and E. Baca-García, "Combining mobile-health (mHealth) and artificial intelligence (AI) methods to avoid suicide attempts: the Smartcrises study protocol," *BMC psychiatry*, vol. 19, no. 1, pp. 1–9, 2019.

[84] Q. Yao, R. Wang, X. Fan, J. Liu, and Y. Li, "Multi-class Arrhythmia detection from 12-lead varied-length ECG using Attention-based Time-Incremental Convolutional Neural Network," *Information Fusion*, vol. 53, pp. 174–182, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253518307632

[85] S. Sahoo, M. Dash, S. Behera, and S. Sabut, "Machine Learning Approach to Detect Cardiac Arrhythmias in ECG Signals: A Survey," *IRBM*, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1959031819301654

[86] P. Kamble and A. Birajdar, "IoT Based Portable ECG Monitoring Device for Smart Healthcare," in *2019 Fifth International Conference on Science Technology Engineering and Mathematics (ICONSTEM)*, vol. 1, 2019, pp. 471–474.

[87] H. Kim, S. Kim, N. V. Helleputte, A. Artes, M. Konijnenburg, J. Huisken, C. V. Hoof, and R. F. Yazicioglu, "A Configurable and Low-Power Mixed Signal SoC for Portable ECG Monitoring Applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 2, pp. 257–267, Apr. 2014.

[88] C. H. Tseng, "Coordinator Traffic Diffusion for Data-Intensive Zigbee Transmission in Real-time Electrocardiography Monitoring," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 12, pp. 3340–3346, Dec. 2013.

[89] M. Bansal and B. Gandhi, "IoT & Big Data in Smart Healthcare (ECG Monitoring)," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon).* IEEE, 2019, pp. 390–396.

[90] Z. Yang, Q. Zhou, L. Lei, K. Zheng, and W. Xiang, "An IoT-cloud based wearable ECG monitoring system for smart healthcare," *Journal of medical systems*, vol. 40, no. 12, p. 286, 2016.

[91] A. Rahman, T. Rahman, N. H. Ghani, S. Hossain, and J. Uddin, "IoT Based Patient Monitoring System Using ECG Sensor," in *2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST)*, 2019, pp. 378–382.

[92] J. Bogatinovski, D. Kocev, and A. Rashkovska, "Feature Extraction for Heartbeat Classification in Single-Lead ECG," in *Proc. MIPRO*, May 2019.

[93] M. Alfaras, M. C. Soriano, and S. Ortín, "A Fast Machine Learning Model for ECG-Based Heartbeat Classification and Arrhythmia Detection," *Frontiers in Physics*, vol. 7, p. 103, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2019.00103

[94] H. Shi, H. Wang, Y. Huang, L. Zhao, C. Qin, and C. Liu, "A hierarchical method based on weighted extreme gradient boosting in ECG heartbeat classification," *Computer Methods and Programs in Biomedicine*, vol. 171, pp. 1–10, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016926071831900X

[95] M. Baza, A. Salazar, M. Mahmoud, M. Abdallah, and K. Akkaya, "On Sharing Models Instead of Data using Mimic learning for Smart Health Applications," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 2020, pp. 231–236.

[96] H. Yang and Z. Wei, "Arrhythmia Recognition and Classification Using Combined Parametric and Visual Pattern Features of ECG Morphology," *IEEE Access*, vol. 8, pp. 47 103–47 117, 2020.

[97] C. Lainscsek and T. Sejnowski, "Electrocardiogram classification using delay differential equations," *Chaos (Woodbury, N.Y.)*, vol. 23, p. 023132, 06 2013.

[98] A. Hannun, P. Rajpurkar, M. Haghpanahi, T. Geoffrey, C. Bourn, M. Turakhia, and A. Ng, "Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network," *Nature Medicine*, vol. 25, Jan. 2019.

[99] M. Wess, P. D. S. Manoj, and A. Jantsch, "Neural network based ECG anomaly detection on FPGA and trade-off analysis," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4.

[100] S. Saadatnejad, M. Oveisi, and M. Hashemi, "LSTM-Based ECG Classification for Continuous Monitoring on Personal Wearable Devices," *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 2, pp. 515–523, 2020.

[101] S. Singh, S. K. Pandey, U. Pawar, and R. R. Janghel, "Classification of ECG Arrhythmia using Recurrent Neural Networks," *Procedia Computer Science*, vol. 132, pp. 1290–1297, 2018, international Conference on Computational Intelligence and Data Science. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050918307774

[102] P. Pławiak and U. R. Acharya, "Novel deep genetic ensemble of classifiers for arrhythmia detection using ECG signals," *Neural Computing and Applications*, pp. 1–25, 2019.

[103] A. Daamouche, L. Hamami, N. Alajlan, and F. Melgani, "A wavelet optimization approach for ECG signal classification," *Biomedical Signal Processing and Control*, vol. 7, no. 4, pp. 342–349, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1746809411000772

[104] S. Sakib, M. M. Fouda, and Z. M. Fadlullah, "A Rigorous Analysis of Biomedical Edge Computing: An Arrhythmia Classification Use-Case Leveraging Deep," in *2020 International Conference on Internet of Things and Intelligence System (IoTaIS 2020)*, BALI, Indonesia, Jan. 2021, doi: 10.1109/IoTaIS50849.2021.9359721.

[105] C. Lainscsek, P. Rowat, L. Schettino, D. Lee, D. Song, C. Letellier, and H. Poizner, "Finger tapping movements of Parkinson's disease patients automatically rated using nonlinear delay differential equations," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 1, p. 013119, 2012.

[106] S. D. Greenwald, R. S. Patil, and R. G. Mark, "Improved detection and classification of arrhythmias in noise-corrupted electrocardiograms using contextual information," in *[1990] Proceedings Computers in Cardiology*, Sep. 1990, pp. 461–464.

[107] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH Arrhythmia Database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, May 2001.

[108] S. D. Greenwald, "The development and analysis of a ventricular fibrillation detector," Ph.D. dissertation, Massachusetts Institute of Technology, 1986.

[109] F. Bouaziz, D. Boutana, and H. Oulhadj, "Diagnostic of ECG Arrhythmia using Wavelet Analysis and K-Nearest Neighbor Algorithm," in *Intl. Conf. on Applied Smart Systems*, Nov. 2018, pp. 1–6.

[110] G. Chen, Z. Hong, Y. Guo, and C. Pang, "A cascaded classifier for multi-lead ECG based on feature fusion," *Computer Methods and Programs in Biomedicine*, vol. 178, pp. 135–143, 2019. [Online]. Available: https://doi.org/10.1016/j.cmpb.2019.06.021

[111] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv preprint arXiv:1803.08375*, 2018.

[112] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Chau, "CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization," *arXiv preprint arXiv:2004.15004*, 2020.

[113] M. Wu, Y. Lu, W. Yang, and S. Y. Wong, "A Study on Arrhythmia via ECG Signal Classification Using the Convolutional Neural Network," *Frontiers in Computational Neuroscience*, vol. 14, p. 106, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fncom.2020.564015

[114] E. J. da S. Luz, W. R. Schwartz, G. Cámara-Chávez, and D. Menotti, "ECG-based heartbeat classification for arrhythmia detection: A survey," *Computer Methods and Programs in Biomedicine*, vol. 127, pp. 144–164, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169260715003314

[115] R. Kohavi *et al.*, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Ijcai*, vol. 14, no. 2. Montreal, Canada, 1995, pp. 1137–1145.

[116] R. K. Pathinarupothi, P. Durga, and E. S. Rangan, "IoT-Based Smart Edge for Global Health: Remote Monitoring With Severity Detection and Alerts Transmission," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2449–2462, 2019.

[117] S. Sakib, M. M. Fouda, Z. M. Fadlullah, N. Nasser, and W. Alasmary, "A Proof-of-Concept of Ultra-Edge Smart IoT Sensor: A Continuous and Lightweight Arrhythmia Monitoring Approach," *IEEE Access*, vol. 9, pp. 26 093–26 106, 2021, doi: 10.1109/ACCESS.2021.3056509.

[118] K. Balaskas and K. Siozios, "ECG Analysis and Heartbeat Classification Based on Shallow Neural Networks," in *Proc. MOCAST)*, May 2019.

[119] R. Sandeep and K. C. Ray, "Sparse representation of ECG signals for automated recognition of cardiac arrhythmias," *Expert Systems with Applications*, vol. 105, pp. 49–64, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417418301842

[120] A. K. Dohare, V. Kumar, and R. Kumar, "Detection of myocardial infarction in 12 lead ECG using support vector machine," *Applied Soft Computing*, vol. 64, pp. 138–147, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494617307159

[121] D. Azariadi, V. Tsoutsouras, S. Xydis, and D. Soudris, "ECG signal analysis and arrhythmia detection on IoT wearable medical devices," in *2016 5th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, 2016, pp. 1–4.

[122] M. Zhang, Y. Wang, and T. Luo, "Federated Learning for Arrhythmia Detection of Non-IID ECG," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020, pp. 1176–1180.

[123] A. Bhatla, M. M. Mayer, S. Adusumalli, M. C. Hyman, E. Oh, A. Tierney, J. Moss, A. A. Chahal, G. Anesi, S. Denduluri *et al.*, "COVID-19 and cardiac arrhythmias," *Heart Rhythm*, vol. 17, no. 9, pp. 1439–1444, 2020.

[124] A. N. Kochi, A. P. Tagliari, G. B. Forleo, G. M. Fassini, and C. Tondo, "Cardiac and arrhythmic complications in patients with COVID-19," *Journal of Cardiovascular Electrophysiology*, vol. 31, no. 5, pp. 1003–1008, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/jce.14479