CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

Implementation and Performance Evaluation of Smartphone vs Raspberry Pi as Edge Devices in IoT Based Healthcare System

by

Waqas Mehmood

A thesis submitted in partial fulfillment for the degree of Master of Science

in the

Faculty of Computing
Department of Computer Science

2025

Dedicated to my better half, Zoha Qamar.

CERTIFICATE OF APPROVAL

Implementation and Performance Evaluation of Smartphone vs Raspberry Pi as Edge Devices in IoT Based Healthcare System

by

Waqas Mehmood

(MCS223010)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Rafia Mumtaz	NUST, Islamabad
(b)	Internal Examiner	Dr. Nayyer Masood	CUST, Islamabad
(c)	Supervisor	Dr. Amir Qayyum	CUST, Islamabad

Dr. Amir Qayyum
Thesis Supervisor
April, 2025

Dr. Abdul Basit Siddiqui
Head
Dept. of Computer Science
April, 2025

Dr. M. Abdul Qadir
Dean
Faculty of Computing
April, 2025

Author's Declaration

I, **Waqas Mehmood** hereby state that my MS thesis titled "**Implementation and Performance Evaluation of Smartphone vs RaspberryPi as Edge Devices in IoT Based Healthcare System**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my MS Degree.

(Waqas Mehmood)
Registration No: MCS223010

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "**Implementation and Performance Evaluation of Smartphone vs RaspberryPi as Edge Devices in IoT Based Healthcare System**" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and that complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS Degree, the University reserves the right to withdraw/revoke my MS degree and that HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Waqas Mehmood)

Registration No: MCS223010

Acknowledgement

All praises to Allah Who created me and taught me what I didn't know. I am grateful to my supervisor Dr. Amir Qayyum who rectified the framework of this thesis and guided me throughout my research. I also thank him for the support and motivation he provided during the hard times of this research. He helped in the implementation of proposed methodology and continuously backed me up to improve the results.

I am thankful to the Head of my department, Dr. Abdul Basit Siddiqui, who allowed me to complete my research. I also want to thank Dr. Nayyer Masood who gave me the opportunity to visit United Kingdom on a research training program at University of the West of Scotland and supported me throughout the degree duration. Thanks to my friends and colleagues at CUST for being well wishers. Finally, special thanks to my family who allowed me to complete this research without creating disturbances.

This research was conducted as part of the DigiHealth-Asia project, co-funded by Erasmus+ Programme of the European Union. The findings, interpretations, and conclusions expressed in this thesis are those of the author and do not necessarily reflect the views of the funding organization or its partners.

Waqas Mehmood

Abstract

This study evaluates the feasibility and performance of using a smartphone and a Raspberry Pi as edge devices in an Internet of Things (IoT)-based healthcare system, with a specific focus on electrocardiogram (ECG) monitoring..

ECG data we are dealing with often contain noise components like Power Line Interference, Baseline Wander, Electrode Motion Artifacts and Electromyographic, which make it susceptible to degradation of the signal quality and accuracy. These noise components must be removed effectively to make it helpful in decision making. At first, we denoised ECG Signals from each sensor using filters like wavelet transform, band pass and outlier filter to improve signal quality. To perform accurate ECG inference, a deep neural network (DNN) was trained on the MIT-BIH dataset. The trained model was deployed on both the smartphone and Raspberry Pi. Their performance was assessed to determine the viability of these edge devices in real-time IoT-based ECG monitoring.

We conducted a comprehensive performance comparison under the identical testing conditions to ensure an equitable evaluation. The performance was evaluated through packet delivery ratio, delay metrics and throughput. After the system was up and running, the maximum delay times between packets were measured for different devices used in this system. It was found that average delay time was higher in smartphone. Moreover, greater discrepancies in maximum delay per packet were shown in Smartphone compared to the Raspberry Pi. This study highlights the trade-offs between processing speed, delay variance, and modularity among edge devices. Smartphones are better at user friendliness but Raspberry Pi is more efficient for the network. To make these findings more actionable, they will help improve IoT-based healthcare, enabling selection of appropriate edge devices for relevant use cases. The results of this study provide valuable insights for developers and researchers working on IoT-based solutions.

Contents

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Network
DNN	Deep Neural Network
EC-IoT	Edge-centric Internet of Things
ECG	Electrocardiogram
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
PDR	Packet Delivery Ratio
PLI	Power Line Interference
PLR	Packet Loss Ratio
RPI	Raspberry Pi

Chapter 1

Introduction

The rapid advancements in Internet of Things and edge computing have revolutionized the healthcare sector, allowing remote monitoring and real-time data processing [1]. Edge device has become the core component of IoT system that process and transfer important data like Electrocardiogram (ECG) data. In this study, we investigate the implementation of smartphones and Raspberry Pi devices as edge nodes in an IoT-based healthcare system that utilizes an Electrocardiogram (ECG) sensor [2].

In healthcare IoT, edge computing plays an important role in improvement of quality of service. It gives multiples advantages like reducing latency and enhanced privacy and security [2]. Automating the collection, delivery, and processing of patient vital data using edge devices, It accelerates the process of diagnosis and treatment [3]. This facilitates delivering more patient specific and timely treatment to patients .

To investigate the performance of smartphones and Raspberry Pi devices as edge nodes in an IoT-based healthcare system, we have designed and implemented a prototype system. Our system utilizes edge devices for remote monitoring and alerts transmission, targeting cardiac conditions.

In this work, a comparative evaluation of smartphones and Raspberry Pi devices for use as edge devices in IoT enabled healthcare system is presented. In addition,

we have also highlighted the performance evaluation of edge devices in terms of network latency, energy efficiency and cost effectiveness. Our work's findings show that smartphones and Raspberry Pi devices can serve both as edge node for IoT based healthcare systems with pros and cons. Typically, smartphones are very familiar devices, but can't compare with Raspberry Pi devices in terms of latency or customizability as you set it up.

Real-world ECG signals are susceptible to various noises, including power line interference (PLI), baseline wander, electrode motion artifacts, and electromyography (EMG) contamination. Figure 1.1 shows the ECG signal containing PLI and Baseline Wander noises. These noises can significantly affect the accuracy of AI model. Milestones of this research are:

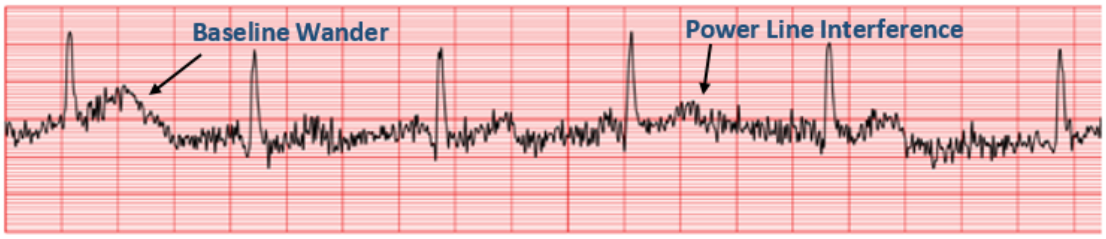

FIGURE 1.1: Raw ECG Signal

1. Collect and denoise ECG signals, addressing common artifacts such as power line interference (PLI), baseline wander, electrode motion artifacts, and electromyography (EMG) contamination.

2. Train a deep neural network (DNN) on the denoised ECG dataset for accurate ECG inference.

3. Deploy the trained DNN model on both smartphones and Raspberry Pi devices.

4. Assess the efficiency and feasibility of these edge devices based on several key performance metrics, including AI model performance, packet delivery ratio (PDR), packet loss ratio (PLR), delay, and throughput.

1.1 Research Objectives

This research aims to design, implement, and assess an IoT-based ECG monitoring system that uses edge computing to achieve real-time performance. The specific objectives are as follows:

1. **Develop and implement denoising techniques:** Implement techniques such as wavelet transforms, band-pass filters, and outlier detection to improve ECG signal quality.

2. **Deploy DNN on edge devices:** Train a deep neural network (DNN) on denoised ECG datasets and deploy the model on edge devices for real-time inference.

3. **Edge Device Testing:** Validate the suitability and performance of smartphones and Raspberry Pi as edge devices based on metrics such as packet delivery ratio, latency, and throughput.

4. **Trade-Off Analysis:** Analyze trade-offs between processing speed, latency variance, and modularity when using different edge devices for monitoring ECG in an IoT environment.

1.2 Rationale and Significance of Selected Research Topic

The selected research topic reflects on the critical challenges and prospects of the modern healthcare delivery. Indeed, choosing appropriate edge devices for systems built with new technologies is among the key concerns of the future. The reasoning behind the selection of the topic is as follows:

1.2.1 Rationale

The growing need for remote patient monitoring demands efficient and adaptable edge devices within IoT based healthcare systems. While smartphones offer widespread availability and user familiarity, their processing power can be limitations. Conversely, Raspberry Pis offer greater processing capabilities but lack the portability and user-friendliness of smartphones. This research addresses this trade-off by comparatively evaluating these two readily available platforms.

By incorporating a denoising pre-processing step before training DNN models, this research delves deeper into the practical applicability proposed AI and IoT based healthcare system. Analyzing performance encompassing both AI model accuracy on denoised data and network communication efficiency (PDR, PLR, Delay, Throughput) provides a more comprehensive understanding of their suitability for realworld healthcare deployments.

1.2.2 Significance:

1. Comparative Analysis: By comparing the performance of smartphones and Raspberry Pis as edge devices, this research will provide valuable insights into their suitability based on critical parameters like AI model performance, data transmission efficiency, and resource utilization.

2. Cost-Effective Solutions: Real-World Applicability: The findings will inform the development and deployment of cost-effective remote patient monitoring systems.

3. Future Research Directions: This research will pave the way for further exploration of using these devices for processing and analyzing other types of healthcare data, potentially leading to more comprehensive remote monitoring solutions.

Addressing each of these aspects requires a strong understanding of signal processing, machine learning, and network communication protocols, ensuring a rigorous and technically challenging MS thesis.

1.3 Scope of the Study

IoT-based ECG monitoring system: Performance evaluation of smartphones and Raspberry Pi. The scope includes:

1. **System Architecture:** Includes IoT sensors, edge devices, and cloud storage.
2. **On-Edge Signal Preprocessing and Noise Reduction:** Both on-edge signal preprocessing and noise reduction techniques are explored.
3. **DNN Model Development:** Developing and training a deep neural network (DNN) model for the classification and detection of ECG signal anomalies.
4. **Performance Evaluation:** Conducting performance evaluation by maintaining constant testing conditions to ensure a fair comparison between edge devices.

The findings of this work are intended to offer insights to developers and researchers for the strategic selection of edge devices used in hospital-based IoT applications, particularly for ECG monitoring.

1.4 Thesis Organization

This thesis is structured as follows:

1. **Literature Review:** Discusses existing research on IoT-based ECG monitoring systems, the role of edge computing, and the challenges of signal processing and device performance.

2. **Design and Implementation of Proposed Solution:** Describes the system architecture, hardware and software components, and the implementation of signal processing and DNN deployment on edge devices.

3. **Assessment and Results:** Presents the comparative performance evaluation of smartphones and Raspberry Pi based on key metrics such as latency, throughput, and packet delivery ratio.

4. **Discussion:** Analyzes the trade-offs between edge devices and their implications for real-time IoT-based healthcare applications.

5. **Conclusion and Future Work:** Summarizes the key findings, contributions, and limitations of the study, and proposes directions for future research.

By addressing the challenges and trade-offs in edge device performance, this study contributes to the advancement of IoT-based healthcare solutions, offering practical recommendations for real-world applications.

Chapter 2

Literature Review

Within the last decade, healthcare and Internet of Things technologies combined received much attention from the researchers. The integration of IoT-based healthcare systems is considered to have a positive impact on patient care by enhancing monitoring, diagnostics, and treatment [4]. Such systems are made up of connected devices and sensors that are responsible for collecting and transmitting patients' data, allowing them to be monitored and analyzed in real-time.

2.1 Edge Computing in IoT

Edge computing is an essential aspect of IoT and provides a solution to some of the challenges of conventional cloud computing. The data is processed at the "edge" of the network, meaning close to the source. Figure 2.1 shows the architecture of edge computing [5]. It reduces latency and bandwidth consumption and have the capability to enhances the data security and privacy [6]. The last aspect is crucial in healthcare, where data should be processed in a timely and secure manner. The utilization of smartphones and other edge devices, such as single-board computers like Raspberry Pi are becoming increasingly common to conduct such local processing [7]. The integration of IoT and edge computing in healthcare has been a growing area of research. Edge computing has proven to be the faster

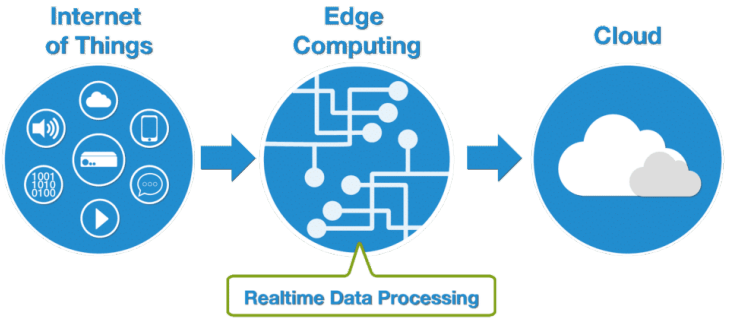

FIGURE 2.1: Edge Computing Architecture

transmission speed, better energy efficiency, and real-time feedback for healthcare applications. For example, in a healthcare system, the edge devices can collect data from various sensors, processes it locally and then send only the necessary information to the cloud, reducing network congestion and data transmission costs [8] .

Furthermore, edge computing can enhance the security and privacy of healthcare data by processing it locally, without the need to transmit sensitive information to the cloud.

2.1.1 Related Work

However, the limitations of traditional cloud computing in IoT environments motivates the use of edge computing as a promising paradigm [9]. As edge computing pushes processing and storage closer to the data source, latency is improved, quality of service is increased, and security and privacy are improved [10]. Some key advantages of edge computing in IoT-based healthcare systems include

1. Reduced latency and faster response times, enabling real-time monitoring and decision-making

2. Improved data security and privacy, as sensitive data can be processed locally without being transmitted to the cloud

3. Better energy efficiency, as less data needs to be sent over the network

Edge computing has been explored in various healthcare applications, such as remote patient monitoring, assisted living, and chronic disease management [11] [12].

2.2 Edge Devices - Smartphones and Raspberry Pi

As technologies have increased rapidly, the healthcare industry has been moving fast towards the implementation of edge computing solutions, including phones, and Raspberry Pi. Smartphones feature powerful processors, an ensemble of sensors as well as abundant connectivity options, have already become part of many healthcare applications[13]. Also, Raspberry Pi, a cheap single board computer has been popularized as an IoT device in view of its efficiency in collecting, processing and transmitting data.

The widespread availability, affordability and the performance capabilities of smartphones and Raspberry Pi have made them the preferred choices for edge computing in healthcare.

There are many advantages of integration of smartphones and Raspberry Pi in the healthcare industry. Smartphones can be used to for multiple applications including real time monitoring of patients health parameters, data collection as well as remote diagnosis [14]. In addition, smartphones are portable and accessible for the health professionals to access and manage patient information at any time and any place.

However, Raspberry Pi has occupied the healthcare sector using its versatility and cost effectiveness. In IoT based healthcare applications, these IoT devices are used where it can be used as a data collection, processing and transmission in Raspberry Pi – based systems [15] [16]. Such systems can deployed in remote

locations or a patients' homes and a solution Continue Health Monitoring can be provided in a scalable and flexible manner [16].

The adoption of smartphones and Raspberry Pi in healthcare has not been without its challenges. One of the primary concerns is the issue of data transmission and processing latency, which can hinder real-time response and emergency detection.

2.2.1 Comparison of Smartphone and Raspberry Pi as Edge Devices

TABLE 2.1: Comparison of Smartphone and Raspberry Pi as Edge Devices in IoT Systems

Metric	**Smartphone**	**Raspberry Pi**
Processing Power	High – Multi-core processors, GPUs	Moderate – ARM processors, limited GPU
Energy Efficiency	Highly optimized for low power consumption	Consumes more power, depends on peripherals
Portability	Highly portable and compact	Portable but bulkier with peripherals
Connectivity	4G/5G, Wi-Fi, Bluetooth	Wi-Fi, Ethernet, Bluetooth
Sensors	Integrated (GPS, accelerometer, gyroscope)	Requires external sensors
Cost	Varies (mid-high)	Low-cost (starting from $35)
Ease of Development	App development (Android/iOS)	Python, C, Linux-based development
Battery Life	Long battery life, optimized for mobility	Requires external power source or battery pack
Data Security	Built-in encryption, biometrics	Security depends on software implementation
Storage	High internal storage	SD card, external USB drives
Real-time Processing	Limited, primarily application-based	Good for lightweight real-time processing

The comparison in Table 2.1 highlights key differences between smartphones and Raspberry Pi as edge devices in IoT healthcare applications.

2.2.1.1 Comparative Studies of Edge Devices

1. **Smartphones**

   Advantages: User-friendly interfaces, high computational power, and extensive connectivity options.

   Challenges: Higher latency and variability in performance due to multitasking and resource allocation.

2. **Raspberry Pi**

   Advantages: Cost-effective, energy-efficient, and optimized for dedicated tasks. Suitable for networked applications with consistent performance.

   Challenges: Limited user interface and slightly lower computational capabilities compared to modern smartphones.

2.3 ECG Signal Processing

There are a variety of noise sources that make real world electrocardiogram signals susceptible to degradation of the signal quality and accuracy [17]. Besides power line interference, baseline wander, electrode motion artifacts and electromyographic contaminants are also present in ECG signals. But these different types of noise have a large detrimental impact on the reliability and clinical interpretation of ECG data, which is why it is so important to properly address these issues using appropriate signal processing and noise reduction techniques.

For example, power line interference that occurs as a result of electrical grid can cause 50/60 Hz noise in ECG signal [18]. On the contrary, baseline wander refers to a slow moving fluctuation of the ECG signal baseline caused by respiration or subject movement. The motion of the electrodes on the skin results in change of the measured electrical potential, which produces electrode motion artifacts. Electromyographic (EMG) noise, caused when the body moves during sleep, will powerfully contaminate the ECG signal and make the accurate analysis of cardiac activity difficult [19].

To address these noise sources, researchers have explored a variety of signal processing techniques. Adaptive filtering, empirical mode decomposition, and deep learning-based approaches have all shown promise in effectively removing noise from ECG signals and improving the accuracy of heartbeat detection and other diagnostic tasks [18].

By developing robust and automated methods for ECG denoising, researchers can help bridge the gap between the controlled environment of clinical studies and the more unpredictable real-world conditions encountered in practical applications. These advanced signal processing techniques can significantly enhance the reliability and clinical utility of ECG data, making it more suitable for real-world healthcare applications where noise and artifacts are often prevalent. [18]. Following famous methods have been developed to overcome these challenges:

1. Power Line Interference (PLI): Adaptive filtering and notch filters are the most commonly employed to deal with PLI.

2. Baseline Wander: Polynomial fitting, wavelet transform, and high-pass filtering techniques can be employed to fix baseline drift.

3. Electrode Motion Artifacts and EMG Contamination: Signal averaging and advanced filtering, such as Kalman filters and Independent Component Analysis (ICA), can be employed to decrease these artifacts

2.4 Deep Neural Network for ECG Analysis

Deep Neural Networks have shown an unprecedented performance in processing ECG signals since they can learn relevant complex patterns and features from data. DNNs can be adopted to classify arrhythmias from ECG signal evaluation, detect heart beat anomalies, or offer predictive information about a patient's cardiac output. It has been found that deep learning algorithms significantly surpass

the accuracy and performance of traditional machine learning techniques. Nevertheless, to make the DNN model workable and deployable in edge devices, it is necessary to design the model effectively and optimize it efficiently [20].

2.4.1 Key Aspects of DNNs in ECG Analysis

2.4.1.1 Dataset Preparation

ECG datasets are compiled, processed, and enhanced to achieve resilience against noise and artifacts. The initial steps in dataset preparation are crucial and include:

1. **Noise Reduction:** Removal of artifacts and external interference from the raw ECG signals.
2. **Feature Extraction:** Identifying and extracting relevant features such as heart rate, QRS complexes, and ST-segment deviations.

2.4.1.2 Training Algorithm

DNN models are trained using supervised learning techniques, where labeled ECG data serves as the input for learning. Some commonly used architectures include:

1. **Convolutional Neural Networks (CNNs):** Suitable for capturing spatial and temporal patterns in ECG signals.
2. **Recurrent Neural Networks (RNNs):** Ideal for handling sequential data and long-term dependencies in ECG waveforms.

2.4.1.3 Edge Deployment

Trained DNN models can be deployed on edge devices, enabling faster inference without reliance on cloud computing. Examples of such edge devices include:

1. **Raspberry Pi:** Compact and cost-effective for real-time analysis.
2. **Smartphones:** Widely accessible and capable of running lightweight models.

Edge deployment ensures low latency and enhanced privacy, as sensitive ECG data need not be transmitted to cloud servers.

2.5 Performance Metrics for Edge Computing in Healthcare

Evaluating the performance of edge devices in healthcare applications involves multiple metrics:

2.5.1 AI Model Performance

As artificial intelligence models become increasingly prevalent in edge computing environments, the need for comprehensive performance evaluation has become paramount. Metrics such as accuracy, precision, recall, F1-score, and inference time are crucial in assessing the effectiveness of these models when deployed on resource-constrained edge devices [21].

A measure of model performance is accuracy, the ability for the model to correctly classify or predict outcomes. Precision, however, measures the proportion of positive predictions that are true, and recall calculates the proportion of true positives that the model can find [22]. A balanced measure of the model performance is given by the F1 score, as the harmonic mean between precision and recall [21] . Edge deployment places high importance on the inference time, i.e. time taken by the model to make a prediction, as edge devices, generally, have constrained computational resources. For real time decision making and responsiveness in edge applications inference time must be optimized

2.5.2 Network Performance Metrics

One of the key performance metrics used to assess the quality of an IoT network is the Packet Delivery Ratio, which represents the ratio of successfully delivered data packets to the total number of packets transmitted. Another important metric is the Packet Loss Ratio, which measures the percentage of data packets that fail to reach their intended destination [23]. Delay and throughput are also critical factors, as they directly impact the timeliness and capacity of the network.

2.6 Comparative Analysis and Survey of Existing Techniques

2.6.1 Survey of Existing Techniques for ECG Signal Denoising

Effective ECG signal processing is critical for accurate diagnosis and monitoring of cardiovascular conditions. Several techniques have been developed to address common artifacts in ECG signals:

2.6.2 Power Line Interference (PLI)

1. Notch Filters: Keeping signal integrity is an often encountered problem in science and engineering applications in which an unwanted interference can substantially corrupt measures as well as analysis of data collected [24].Power line frequency, for example 50 or 60 Hz, is one of the typical common sources of interference that can pollute a broad variety of signals. To solve this problem, numerous studies have been conducted on notch filters to eliminate power line interference of this particular frequency [25]. Due to their simplicity and easy of implementation, Notch Filters are a commonly used

power line interference mitigation, a solution that works in many applications. Nevertheless, like notch filters, they can distort the signal of interest while removing the undesired interference [25].

The use of notch filters to suppress power line interference is widely known to be effective , but the introduction by notch filters of undesirable distortion into the signal itself is an area of increasing research interest . In [26] , the potential of the effects of electromagnetic compatibility filters, including notch filters, on narrowband power line communications have been characterized.

2. Adaptive Filters: In the field of signal processing, adaptive filters provide a powerful means of processing; they are superior to fixed filters in many respects. Adaptive filters are able to adjust their parameter values in real time, so that interference can be minimized and the signal characteristics changing over time can be adaptively followed [27]. The property of this repetition characterizes better performance since the filter can sequentially optimize its behavior to the input signal [28].

   On the other hand, these filters are adaptive in nature. Generally, adaptive filters are more computationally intensive than fixed filters, since their parameters must be updated continuously in order to be optimally performing. The tradeoff between performance and computational cost when choosing a filter for a particular application forms this core consideration.

   The Butterworth digital filter constitutes one adaptive filter example, as it is adaptive to changes in the frequency content of the signal [27]. A second adaptive filtering technique is the Kalman filter, which has found extensive application in integrated GPS/INS navigation systems. It has been shown that these Kalman based adaptive filters outperform conventional Kalman filters in terms of both accuracy and computational efficiency; thereby, very suitable for cost effective solutions.

   Although the performance advantages of adaptive filters are well documented, researchers have considered a number of approaches to increase their computational efficiency. The first proposed approach uses a fast adaptive Kalman

filtering algorithm to achieve high accuracy while reducing the execution time compared to traditional methods. Also, combining different adaptive filter structures can enhance the filter performance and the convergence properties, but the selection of the proper combination of filters may demand special caution (Street et al., 1968; Jer Nimo Arenas Garc'a, c5aed24f).

2.6.3 Baseline Wander:

1. Polynomial Fitting: A very powerful tool for removing slow drifts from signals is polynomial fitting: it can model arbitrarily complicated baseline variations well. Fitting a polynomial function on the baseline of the signal to fit the baseline and subtract to detrend the data. (Ref [29]). This approach can be very effective but comes with careful selection of the polynomial order to prevent being overfit or underfit from the data.

   Polynomial fitting is one of the key advantages for dealing with very different types of baseline variations including simple linear trends to more difficult non-linear drifts. Therefore, the polynomial function can be adapted to the features of a signal and hence can be used to better represent the baseline. Evolutionary Polynomial Regression is a data driven technique based on an evolutionary computing approach to search for an evolutionary computed polynomial model, while to get over the problems of traditional numeric regression [30].

   Additionally, polynomial fitting flexibility is increased in that a polynomial can be fit to the entire signal, or the signal can be broken up into pieces and a polynomial fit to each piece. The capability of capturing more localized baseline variation is enabled, which can be very useful in the case of non stationary signals.

   Yet, this technique requires an important choice on the appropriate polynomial order. If the order is too low, it is not possible to fit a polynomial that adequately captures the baseline and the drift removal is incomplete. On the other hand, if the order is too high, the polynomial can over fit the

data and introduces artifacts or distorting artifacts on the detrending signal [29][31].

In order to address this challenge, cross-validation and jackknife resampling techniques have been proposed [31].

2. Wavelet Transform: Wavelet signal processing techniques have recently surfaced as a powerful means for many applications such as digital signal processing [32] and geophysical data analysis. These are methods that can be used to decompose a signal into its component frequencies (which can be really useful for identifying, and removing if unwanted, drift/trend from the data).

   A major feature of wavelet based methods is their self attention to the time frequency resolution with respect to the signal. Unlike the short-time Fourier transform where a window function of fixed width is used, wavelet functions can be scaled and translated so that resolution in time and frequency are variable and flexible [33]. Flexibility in the time and frequency spectrum of the signal is critical for frequency selective analysis and processing of time varying signals.

   An important item, in wavelet based signal processing, is the choice of the wavelet function. Different parameters of the wavelet function can affect the conditions which the wavelet must satisfy, and the time and frequency spreads of the wavelet function can cause that the analysis results are highly influenced.

3. High-Pass Filtering: Baseline wander (also referred to as baseline drift) is an important source of low frequency noise with frequency components between 0.05 Hz and 3 Hz [34] that can highly affect the interpretation and analysis of the ECG signal. High-pass filtering is proposed to correct this problem and eliminate the low frequency ECG signal wander and the low frequencies of the signal.

   Among the principal benefits of high-pass filtering is the removal of the low frequency components of the ECG signal out of contact patient movement,

respiration and other physiological phenomena. In ambulatory and stress test settings, ECG signals can be subjected to higher levels of noise and baseline wander as acquisition condition is not as good [35] and therefore this filtering is very useful. Additionally, the suppression of low frequencies can improve visibility and interpretation of important ECG features, those P-waves and T-waves, that are significant for accurate diagnosis and monitoring of cardiac health [36].

However, a common form of high-pass filtering is not without its challenges. The choice of cutoff frequency of the filter is very critical since the wrong selection of the cutoff frequency may cause the loss of the important low frequency information or still the best option not able to get rid of the baseline wander. Moreover, design and implementation of high-pass filters is complex, involving judicious selection of filter order, stability, and group delay.

To address these challenges, recent research has explored the use of deep learning techniques for ECG signal denoising and baseline wander removal. These deep learning-based approaches have shown promising results in effectively removing baseline wander while preserving the important low-frequency components of the ECG signal.

In conclusion, high-pass filtering is a widely used technique for eliminating ECG signal wander and suppressing low-frequency components.

2.6.4 Electrode Motion Artifacts and EMG Contamination:

1. Signal Averaging: This is one way of filtering that can be achieved by averaging the residuals of multiple ECG cycles. Though a simple method, it does not adequately reduce periodic noise; moreover, it is completely inefficient against non-periodic noise and artifacts.

2. Kalman Filtering: This is an advanced filtering technique that estimates the true signal based on the signal and noise models. Kalman filtering in its linear variants is a form of the Wiener filter, where both the signal and the noise processes are Gauss-Markov processes.

3. Independent Component Analysis (ICA): ICA is a powerful method to remove noise. ICA takes the mixed signals and separates them to get independent component signals, and some of these components will be noise signals [37]. To suppress these components, they can be removed. The disadvantage of ICA is the high computational cost.

2.6.5 Performance Metrics for Evaluating Edge Devices

Evaluating the performance of edge devices in healthcare applications involves multiple metrics:

2.6.5.1 AI Model Performance:

1. Accuracy, Precision, Recall, F1-Score: These metrics assess the classification performance of machine learning models deployed on edge devices, ensuring they meet clinical requirements.

2. Inference Time: The time taken by the model to make predictions is crucial for real-time applications.

2.6.5.2 Network Performance Metrics:

1. Packet Delivery Ratio (PDR) and Packet Loss Ratio (PLR): These metrics evaluate the reliability and consistency of data transmission between devices.

2. Delay: The time delay in data transmission affects the responsiveness of the system, which is critical for real-time monitoring.

3. Throughput: The rate at which data is processed and transmitted by the edge device, indicating its capacity to handle large volumes of data.

2.6.6 Trade-Offs in Edge Computing for ECG Monitoring

2.6.6.1 Processing Speed vs. Latency

While smartphones offer faster processing speeds, they often exhibit higher and more variable latency compared to Raspberry Pi. This can impact real-time ECG monitoring, where consistent responsiveness is crucial.

2.6.6.2 Modularity and Scalability

Raspberry Pi systems are highly modular, allowing for customization and scalability in IoT networks. Smartphones, while less modular, provide greater integration with existing technologies.

2.6.6.3 Power Consumption

Power efficiency is a significant consideration in IoT healthcare devices. Raspberry Pi consumes less power, making it ideal for continuous monitoring applications. Smartphones, on the other hand, may require frequent recharging.

2.6.7 Identified Research Gaps

While IoT-based healthcare systems and edge computing have substantially developed, there is a considerable lack of in-depth comparative studies on edges, especially smartphones and Raspberry Pi, in terms of ECG signal processing and analysis. Specifically, some researchers pointed to the following gaps:

2.6.7.1 Comparative Performance Analysis:

The proliferation of the Internet of Things in the healthcare domain has revolutionized the way patient data is collected, transmitted, and analyzed [38]. Although smartphones and Raspberry Pi devices differ significantly in terms of their processing power, memory, and network capabilities, the impact of these characteristics on packet delivery and loss rates in healthcare IoT systems has not been thoroughly explored [39].

Several studies have highlighted the benefits and challenges of IoT-enabled healthcare systems. Research in this area could investigate the influence of CPU utilization, memory constraints, and network interface performance of these two device types on critical metrics such as Packet Delivery Ratio, Packet Loss Ratio, Delay, and Throughput. Understanding the relationships between device characteristics and packet reliability can help optimize data transmission protocols and buffer management schemes to minimize packet loss and improve delivery reliability [40].

2.6.7.2 Data Processing Techniques:

1. Signal Denoising Techniques: Although several signal denoising techniques have been proposed, there is a gap in research that systematically evaluates the effectiveness of these techniques.

2. Integration with Machine Learning Models: The integration of advanced denoising techniques with deep neural networks for electrocardiogram analysis on edge devices is an area that has not been extensively explored. There is a need for research that not only evaluates the denoising performance but also examines how these techniques can enhance the accuracy and reliability of subsequent machine learning inferences on edge devices [41] .

   Integration of advanced denoising techniques with these deep neural networks is an area that requires further investigation. The challenges of interpretability, scalability, and efficiency in deep learning models for ECG data analysis also need to be addressed [41].

Edge computing has emerged as a paradigm that deploys computational and storage capabilities at the network edge, in close proximity to end-users. This can potentially enhance service performance by reducing latency and increasing throughput compared to conventional data center-based cloud computing.

2.6.8 Sufficiency of Research Topic to Qualify as MS Thesis

This research topic, focusing on the feasibility and efficiency of smartphones and Raspberry Pis as edge devices in an ECG-based IoT healthcare system, offers sufficient depth and scope to qualify as an MS thesis & for several reasons:

Technical Depth and Complexity:

It involves multiple technical aspects:

1. Pre-processing ECG data for noise removal.
2. Implementing Deep Neural Network (DNN) models for analysis.
3. Evaluating the performance of these models on data processed by different edge devices.
4. Analyzing data transmission efficiency metrics like PDR, PLR, delay, and throughput

2.7 Problem Statement

“A detailed comparative study is needed to understand how smartphones and Raspberry Pi perform as edge devices in IoT-based healthcare systems. There is

almost no research considering implementation issues, which are crucial for continuous monitoring of healthcare, such as power consumption, network dependability, or the user interface. In addition, ECG signal denoising and AI model training on denoised datasets are also major points of concern. The obstacles include signal artifacts in ECG signals, such as baseline wander, power line interference, electrode motion artifacts, and electromyography (EMG) contamination."

2.8 Research Questions

1. What are the methods to be adopted for ECG signal denoising before training the model?
2. How do smartphones and Raspberry Pi compare in terms of performance and feasibility as edge devices for real-time inference in IoT-based healthcare systems regarding key metrics such as AI model performance, packet delivery ratio, packet loss ratio, delay, and throughput?

2.9 Objectives of the Research

The objectives of the research for the proposed topic can be defined as follows:

1. **ECG Signal Denoising:** Evaluate the effectiveness of ECG signal denoising techniques, performed prior to model training, on both smartphones and Raspberry Pi, ensuring accurate and reliable ECG analysis.
2. **Evaluate AI Model Performance:** Compare the accuracy and robustness of deep neural network (DNN) models deployed on smartphones and Raspberry Pi for real-time ECG inference.
3. **Assess Network Performance:** Measure and compare the packet delivery ratio (PDR), packet loss ratio (PLR), delay, and throughput of data transmission between edge devices and the central server in an IoT-based healthcare system.

2.10 Conclusion

This chapter provided a comprehensive review of the literature surrounding the integration of Internet of Things (IoT) and edge computing in healthcare, with a specific focus on electrocardiogram (ECG) monitoring systems. The review highlighted the transformative potential of IoT-based healthcare systems, which leverage connected devices and sensors to enable real-time patient monitoring. Edge computing emerged as a critical enabler of these systems, offering reduced latency and enhanced data security.

The chapter explored the role of edge devices, particularly smartphones and Raspberry Pi in IoT-based healthcare applications. Smartphones have become a popular choice for real-time health monitoring because of their powerful processors, integrated sensors, and widespread availability. On the other hand, Raspberry Pi with its cost-effectiveness and versatility, has proven to be a reliable option for data collection, processing, and transmission in remote or resource-constrained settings. A detailed comparison of these devices revealed their respective strengths and limitations, emphasizing the need for a balanced approach when selecting edge devices for healthcare applications.

The discussion on ECG signal processing underscored the challenges posed by various types of noise, including power line interference, baseline wander, electrode motion artifacts, and electromyographic (EMG) contamination. Advanced signal processing techniques, such as adaptive filtering, and wavelet transforms were identified as effective solutions for noise reduction and signal enhancement. These techniques are crucial for ensuring the accuracy and reliability of ECG data, particularly in real-world healthcare scenarios where noise and artifacts are prevalent.

The chapter also highlighted the growing role of deep neural networks (DNNs) in ECG analysis, particularly for tasks such as arrhythmia detection and heartbeat anomaly classification. The deployment of DNN models on edge devices, such as smartphones and Raspberry Pi, offers significant advantages in terms of

low latency and enhanced privacy, making them suitable for real-time healthcare applications.

Finally, the chapter identified key performance metrics for evaluating edge devices in healthcare systems, including AI model performance (accuracy, precision, recall, F1-score, and inference time) and network performance metrics (packet delivery ratio, packet loss ratio, delay, and throughput). These metrics are essential for assessing the feasibility and efficiency of edge devices in real-time healthcare applications.

In conclusion, this literature review has laid the groundwork for understanding the potential of IoT and edge computing in healthcare, particularly for ECG monitoring. It has also highlighted the need for further research to address existing gaps, such as the comparative performance analysis of smartphones and Raspberry Pi and the integration of advanced signal processing techniques for real-time healthcare applications. These insights will guide the subsequent chapters of this research, which aim to explore the feasibility and efficiency of smartphones and Raspberry Pi as edge devices in IoT-based healthcare systems.

Chapter 3

Design and Implementation of Proposed Solution

This chapter presents a comprehensive process model for designing, implementing, and evaluating an IoT based healthcare system that monitors ECG. In this chapter the focus was on the system architecture, signal processing techniques, neural network development and performance evaluation along with the experimental setup. Figure 3.1 shows the model training and testing pipeline. The main purpose is to determine the feasibility and efficiency of employing smartphones and Raspberry Pi devices as edges computing platforms for processing real time ECG in real time and to contribute to more reactive and reliable healthcare solutions.

3.1 Data Extraction and Preprocessing

We initially loaded MIT-BIH dataset [42] , which is a collection of a large number of records of ECG signals taken from various patients, and associated annotation files that describe the occurrence of different types of arrhythmias and other cardiac events.

Once we had loaded the dataset, we extracted the signal records and annotation files. The raw ECG data is present in the signal records while the annotation files

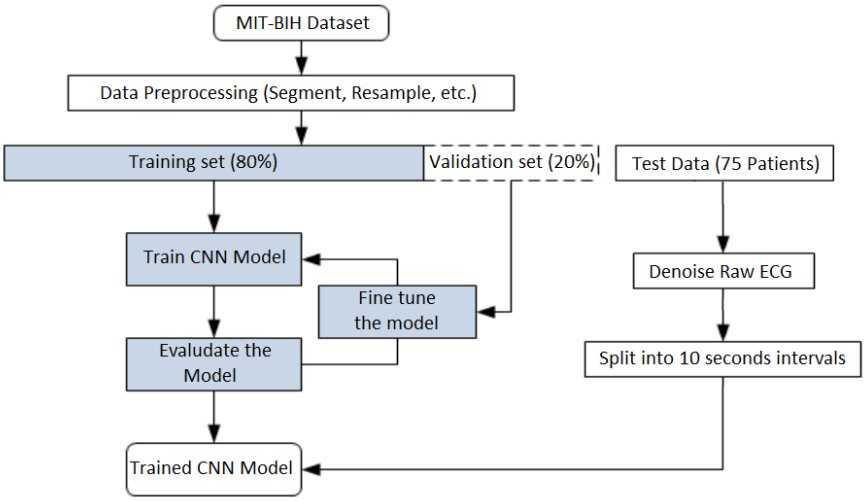

FIGURE 3.1: CNN Model Training & Testing Pipeline

carry essential information on the events at the occurrence of beats or abnormalities in the signals.

Next, we applied several preprocessing techniques to the extracted ECG signals in order to prepare them for model training and evaluation. This preprocessing involved several key steps:

3.1.1 Z-Score Normalization

The Z-Score normalization transforms the data to have a mean of 0 and a standard deviation of 1. The formula is:

$$z = \frac{x - \mu}{\sigma}$$

Where:

- x = original value
- μ = mean of the feature
- σ = standard deviation of the feature

ECG signals in MIT-BIH dataset have varying amplitudes and baseline shifts due to differences in recording conditions, patient physiology, or sensor characteristics.

ECG signals were normalized using z-score normalization, ensuring that amplitude values remained within a common range. Figure 3.2 shows the result of z-score nomalization. This normalization stabilizes the learning process and improves the performance of the model by mitigating scale differences across signals.

	Original Values	Normalized Values
0	-0.145	1.018899
1	-0.145	1.018899
2	-0.145	1.018899
3	-0.145	1.018899
4	-0.145	1.018899
...	...	...
63382	-0.220	0.591051
63383	-0.220	0.591051
63384	-0.250	0.419912
63385	-0.250	0.419912
63386	-0.240	0.476959

[63387 rows x 2 columns]

FIGURE 3.2: Z-Score Normalization on MIT-BIH Dataset

3.1.2 Segmentation

Continuous ECG recordings were segmented into fixed-length windows of 1 second and 360 samples. This segmentation ensured that the signals were compatible with the model's input requirements. Each segment represents a time window of the ECG, allowing the input data to remain uniform and easier to process.

3.1.3 Upsampling

The frequency of ECG signals was adjusted to meet the model's need by upsampling them. Model was trained on 360 Hz data, the signals were upsampled to match this frequency. This step ensured that the input data was in the correct format for the model to process effectively.

3.1.4 Label Encoding

Annotations were converted into categorical representations that the model could interpret. For example, arrhythmias were encoded as binary labels indicating their presence or absence. This transformation enabled the model to learn directly from the annotated data. Normal beats are encoded as 0 and abnormal as 1.

These preprocessing techniques were essential for optimizing the ECG signals for model training and evaluation. By applying these steps, the quality of the data was enhanced, ensuring more accurate and efficient processing during the model's learning phase. The reduction in quality of cardiac data by this preprocessing can be monitored at various reporting intervals.

3.2 Class Balancing

MIT-BIH dataset has total of 109494 heartbeats out of which 75052 are categorized as normal beats and 34,442 as abnormal beats [43]. The dataset is skewed toward normal beats (70%) compared to abnormal beats (30%). A model trained on this imbalanced data will likely overfit to the majority class (normal beats) and underperform on the minority class (abnormal beats).

To address the problem of class balancing in the dataset, two effective methods were employed to balance the classes, ensuring the model had sufficient data to learn from all categories, including underrepresented ones. The dataset was balanced to achieve a 1:1 ratio between the classes using following method.

3.2.1 SKLEARN Method - resample

The first approach involved utilizing a pre-implemented method from the Scikit-learn (SKLEARN) library, specifically the `resample` function. This method allows for rebalancing the data by either oversampling the minority class or undersampling the majority class.

1. **Oversampling:** Instances from the underrepresented class are duplicated to increase their prevalence in the training dataset.

2. **Undersampling:** Instances from the overrepresented class are reduced to balance the dataset.

By applying the `resample` function, the model was less likely to be biased toward the majority class, promoting more balanced predictions across all classes.

3.2.2 SMOTE with NearMiss

The second approach combined two techniques: SMOTE (Synthetic Minority Over-sampling Technique) and NearMiss. This hybrid approach effectively enhanced the model's learning from the minority class while preventing overfitting.

1. **SMOTE:** Instead of duplicating existing samples, SMOTE generates synthetic samples by interpolating between original data points of the minority class. This increases the diversity of the minority class and creates realistic samples that follow the patterns of the existing data.

2. **NearMiss:** NearMiss focuses on undersampling the majority class by selecting samples that are closest in proximity to the minority class. This technique emphasizes complex examples, ensuring the model can better separate the two classes, reducing misclassification errors.

By combining SMOTE and NearMiss, we created a more representative and balanced dataset. SMOTE's synthetic oversampling added new data points to the minority class, while NearMiss reduced the size of the majority class, enhancing the model's ability to differentiate between subtle differences in classes. This dual approach significantly improved the model's performance and resulted in a more diverse and balanced training set with ratio of 1:1. Table 3.1 shows the state of data before class balancing and Table 3.2 shows the state of data after class balancing.

State of Data Before Balancing

TABLE 3.1: State of Data Before Class Balancing

Class	**Number of Samples**
Normal	75,052
Abnormal	34,442
Total	**109,494**

State of Data After Balancing

TABLE 3.2: State of Data After Class Balancing

Class	**Number of Samples**
Normal	75,052
Abnormal	75,052
Total	**150,104**

3.3 Train-Test Split

The dataset was split in 80% for training and 20% as testing for evaluating the model's performance, and avoid overfitting. A large part of the data (the majority of the data in fact) goes into the training set to aid the model in figuring out patterns, features, and relationships in the ECG signals. Training the model on so much of our data allows it to be able to recognize complex features and understand things like normal and abnormal heart rhythm patterns. The other portion of the data, that the model has never seen before, is the testing set. Then we use testing set to evaluate generalization ability of the model, or how well will the model perform on unseen data. This makes sure the model is not simply memorizing the training data rather it's able to predict outcomes on new real world data.

Furthermore, the split was done in stratified fashion, so that the data class distribution was maintained. As a result, the training as well as the testing sets will have a representative share of classes, such as normal and abnormal ECG patterns

which is a requirement for imbalanced datasets such as the MIT-BIH ECG data. By doing so, the test set is not biased toward a specific class, leading to a fair and accurate inference about the model's performance across classes.

3.4 Model Training using CNN

The input is passed through sequential layers of convolution with activation function ReLu, AvgPooling, dropout and dense layers with the sequence shown in Figure 3.3:

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d_6 (Conv1D)           (None, 360, 16)           128

 average_pooling1d_6 (Avera  (None, 179, 16)           0
 gePooling1D)

 conv1d_7 (Conv1D)           (None, 179, 32)           4640

 average_pooling1d_7 (Avera  (None, 89, 32)            0
 gePooling1D)

 conv1d_8 (Conv1D)           (None, 89, 64)            22592

 average_pooling1d_8 (Avera  (None, 44, 64)            0
 gePooling1D)

 flatten_2 (Flatten)         (None, 2816)              0

 dropout_4 (Dropout)         (None, 2816)              0

 dense_4 (Dense)             (None, 35)                98595

 dropout_5 (Dropout)         (None, 35)                0

 dense_5 (Dense)             (None, 5)                 180

 softmax_2 (Softmax)         (None, 5)                 0

=================================================================
Total params: 126135 (492.71 KB)
Trainable params: 126135 (492.71 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

FIGURE 3.3: Neural Network Model Sequential Architecture of Our System

The Figure 3.3 shows the summary of a neural network model (named "sequential_2") implemented using a sequential architecture. The model was built up out of multiple layers designed to take part in some feature extraction and classification tasks. conv1d_6 is a 1D convolutional layer with 16 filters of 3 output size,

and output shape (None, 360, 16) with 128 trainable parameters. The None here is to denote no batch size and this layer is to process the input signal to provide basic features. Next, the average_pooling1d_6 is using 1D pooling with a standard pooling size of 2, reducing the dimensionality to cut the number in half and having an output shape of (None, 179, 16) with 0 more parameters.

The second layer, conv1d_7, is yet another 1D convolutional layer with 32 filters and involves a complex feature and it has an output shape of (None, 179, 32) and 4,640 parameters. Finally, we see the average_pooling1d_7 layer again reducing the input size by half and returns an output shape of (None, 89, 32) and no parameters. Finally, conv1d_8 produces the output shape (None, 89, 64) and the count of filters is increased to 64, extracting more detailed features and having 22,592 parameters. Next are average_pooling1d_8, of which reduces the dimensionality of the input into (None, 44, 64) without introducing any new parameters.

The flatten_2 layer reshares the output from the convolutional stack into a 1D vector and the output shape is (None, 2816) with no parameters. Dropout_4 layer in order to mitigate overfitting consists in applying dropout regularization by randomly removing a fraction of input units during training and keeping the same shape of the output. The second layer, dense_4, is a 35 neurons fully connected layer with 98,595 parameters which is trained to either perform further feature extraction or a classification task. Lastly, this dense layer, (None, 35), is the same output shape is applying dropout regularization on a second dropout layer, dropout_5.

The final layers are dense_5, a layer consisting of 5 neurons with 180 parameters in order to classify into 5 categories. Finally, we have the softmax_2 layer, which applies softmax activation to convert logits as probability distributions for the classification task. This network of layers constitutes an effective sequence of features extraction, dimensionality reduction, and classification, while also providing mechanisms for overfitting reduction and generalization improvement for the model.

3.4.1 Training Configuration

The model was trained using the following configuration:

3.4.1.1 Loss Function

The categorical cross-entropy loss function works well for multi-class classification problems such as ECG signal classification. In this work, it was used to quantify the performance of the classification model, where the output is a probability between 0 and 1 for each possible class. The model is trained to produce predictions that closely match the actual distribution of the data by minimizing the cross-entropy between the actual labels and the predicted probabilities.

3.4.1.2 Optimizer

The Adam (Adaptive Moment Estimation) optimizer was employed to optimize the model. Adam is one of the most widely used optimizers due to its momentum, adaptive learning rate, and efficiency in handling sparse gradients. Unlike traditional gradient descent methods that compute the same learning rate for each parameter, Adam dynamically adjusts the learning rate for each parameter individually. This leads to faster convergence and more stable training.

Adam effectively combines the strengths of two other gradient descent extensions: AdaGrad and RMSProp, resulting in enhanced adaptability and performance during training.

3.4.1.3 Batch Size

A batch size of 36 was used, meaning that 36 samples were processed by the model in each iteration before updating the model's weights. The batch size significantly influences the speed and stability of training. A batch size of 36 was selected to balance system resource usage and stable gradient updates, leading to efficient

training. While experiments with Adversarial Imitation Learning did not yield statistically significant results, a batch size of 36 was found to provide a reasonable trade-off between computational efficiency and performance.

3.4.1.4 Epochs

The model was trained for 50 epochs. An epoch refers to one complete pass through the entire training dataset. Over the course of 50 epochs, the model iterated over the data multiple times, updating its parameters in each iteration. By epoch 50, the model had likely converged sufficiently, with diminishing returns in performance improvements observed in subsequent epochs. This duration was found to be adequate to achieve robust classification performance while avoiding overfitting.

In this training configuration, the model was optimized to maximize the learning of useful statistical features from ECG data, ensuring strong classification performance and generalization to new, unseen data.

3.5 Testing Model on Self-Collected ECG Dataset

For testing the model, we utilized our own ECG dataset, which was recorded from 75 patients. Each patient's recording lasted approximately 5 minutes. These recordings were segmented into 10-second intervals, resulting in a total of 1894 samples. All samples in the dataset were classified as abnormal.

The segmented ECG samples were then fed into a pre-trained machine learning model based on Convolutional Neural Networks (CNNs) to evaluate its performance on real-world data.

3.5.1 Self-Collected ECG Upsampling and Denoising

The test data was collected at a frequency of 200Hz. However, since the model was trained on data sampled at 360Hz, it was necessary to up-sample the test data to match the training frequency. This upsampling ensured compatibility between the test and training datasets, allowing for accurate evaluation. Figure 3.4 shows the result of the applied denoising technique on upsampled dataset.

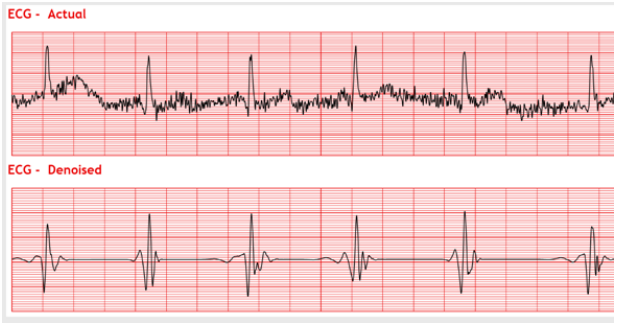

FIGURE 3.4: Actual and Denoised ECG Signal

1. **Power Line Interference (PLI):**

   Power line interference shown in Figure 3.5 is a common type of noise in ECG signals, typically resulting from electrical mains at 50/60 Hz. This interference can mask critical ECG features, and its removal is necessary to improve signal clarity.

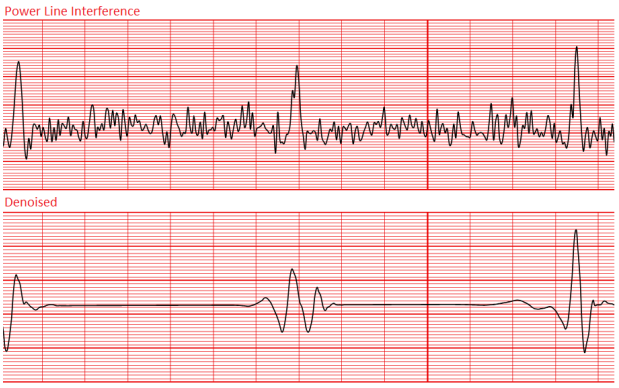

FIGURE 3.5: PLI observed in received ECG signal and denoised

2. **Baseline Wander:**

   Baseline wander shown in Figure 3.6 is characterized by low-frequency noise

that shifts the ECG signal baseline. This noise is often caused by patient movement, respiratory patterns, or electrode displacement. Baseline correction techniques are applied to realign the signal to its true baseline.

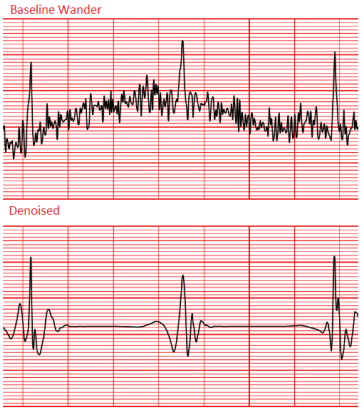

FIGURE 3.6: Baseline Wander observed in received ECG signal and denoised

3. **Electrode Motion Artifacts and Electromyography (EMG) Contamination:**
   Electrode motion artifact shown in Figure 3.7 s arise from physical movement or poor electrode contact, resulting in abrupt changes in the ECG signal. EMG contamination, on the other hand, stems from muscle contractions that introduce high-frequency noise into the ECG signal. Artifact suppression methods are essential to ensure the model correctly interprets ECG features without interference.

Addressing these noise factors through preprocessing enhances the ECG signal quality, resulting in better model performance and more accurate classification of cardiac conditions.

3.5.1.1 Denoiser Interface

We designed a web interface to test the denoising result of signal in real time by tweaking different parameters. Figure 3.8 shows the interface layout that performs denoising in realtime.

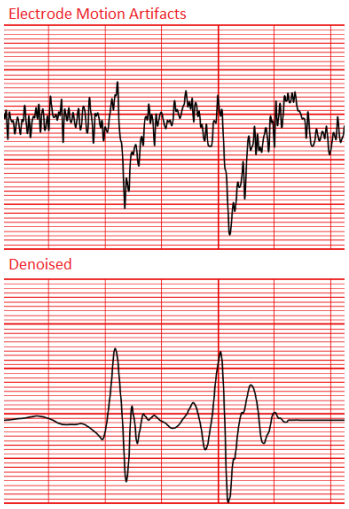

FIGURE 3.7: Electrode Motion Artifacts observed in received ECG signal and denoised

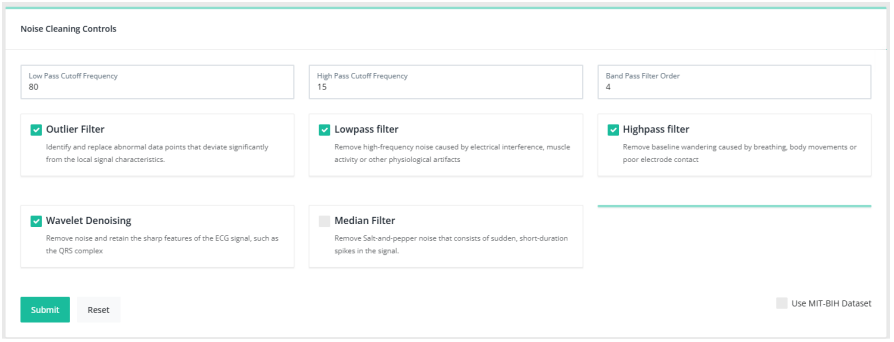

FIGURE 3.8: ECG Signal Denoiser Interface

Filter Settings

The following filter settings are applied to preprocess ECG signals, ensuring enhanced signal quality by removing various types of noise and artifacts.

1. **Low Pass Cutoff Frequency**

   A low-pass filter allows frequencies below this threshold to pass, effectively blocking higher-frequency noise that may interfere with the ECG signal.

2. **High Pass Cutoff Frequency**

   A high-pass filter allows frequencies above this threshold to pass, eliminating low-frequency noise such as baseline wander caused by breathing or patient movement.

3. **Band Pass Filter Order**

   Determines the complexity of the bandpass filter, which affects the steepness of the cutoff and the precision of the filter in isolating desired frequency bands.

4. **Outlier Filter**

   This filter identifies and replaces abnormal data points that significantly deviate from the local signal characteristics, ensuring smoother and more reliable ECG signals.

5. **Lowpass Filter**

   The lowpass filter removes high-frequency noise caused by electrical interference, muscle activity, or other physiological artifacts, preserving the integrity of the primary ECG waveform.

6. **Highpass Filter**

   The highpass filter removes baseline wander, which is often caused by factors such as body movements, breathing, or poor electrode contact. This helps stabilize the ECG signal.

7. **Wavelet Denoising**

   This technique applies a wavelet transform to remove noise while retaining sharp features of the ECG signal, such as the QRS complex. This is particularly effective in preserving key diagnostic components of the signal.

8. **Median Filter**

   The median filter, though currently disabled, can be used to remove salt-and-pepper noise, which consists of short-duration spikes in the signal. This filter is useful for eliminating sudden, isolated noise points.

This interface allows users to preprocess ECG signals by applying various filtering and denoising techniques. The primary objective is to enhance the quality of the ECG signal by effectively removing noise sources that can interfere with accurate signal interpretation and analysis.

By adjusting filter frequencies and selecting specific techniques, users can tailor the preprocessing pipeline to the needs of their data. This customization improves the accuracy of downstream ECG analysis or machine learning models by ensuring that the signal is clean and interpretable.

The interface provides options for applying multiple noise reduction techniques, such as:

1. **Low-pass and High-pass Filters:**
   Designed to remove high-frequency and low-frequency noise, ensuring that only the desired signal band is retained.

2. **Wavelet Denoising:**
   A technique that preserves the sharp features of the ECG signal (e.g., QRS complex) while eliminating noise.

3. **Outlier Detection:**
   Identifies and replaces abnormal data points that deviate significantly from the local signal characteristics.

This preprocessing step is crucial for improving the quality of ECG signals, ultimately leading to better performance and reliability in ECG classification models or diagnostic applications.

3.6 System Architecture and Deployment

3.6.1 System Architecture

The proposed Internet of Things (IoT)-based electrocardiogram (ECG) monitoring system leverages edge computing to process ECG signals closer to their source. The ECG monitoring system was designed to send spliced signals from the ESP32 ECG sensor to edge devices (such as smart phones and Raspberry Pis) using MQTT protocol. The edge devices served as MQTT brokers and performed local

data processing and the results were send to a cloud hosted server for additional analysis and storage.

The system architecture integrates IoT devices such as sensors, smartphones, and Raspberry Pi to achieve real-time monitoring and data analysis. The design objectives focus on minimizing latency, ensuring data privacy, and improving the overall system responsiveness.

The system consists of the following core components:

1. **ECG Sensors:** These devices capture raw ECG signals from the patient and transmit them to the edge devices for processing.

2. **Edge Devices:** Smartphones and Raspberry Pi are used as edge devices to perform data preprocessing, denoising, and analysis. These devices execute computationally intensive tasks locally, reducing the need to transfer large amounts of data to cloud servers.

3. **Cloud Server:** While most processing is performed on the edge, the cloud server stores long-term patient data for historical analysis and remote access by healthcare professionals.

4. **User Interface:** A web application deployed on cloud provides real-time visualization of ECG signals and alerts for anomalies, enhancing usability and accessibility for patients and clinicians.

3.6.1.1 Data Flow

1. **Signal Acquisition:** ECG sensors capture cardiac signals and forward them to the nearest edge device.

2. **Preprocessing:** The edge device applies denoising techniques such as wavelet transforms, band-pass filters, and outlier detection to improve the quality of the ECG signals.

3. **Inference:** A trained deep neural network (DNN) deployed on the edge device analyzes the denoised signals for real-time anomaly detection.

4. **Data Storage and Alerts:** Critical anomalies trigger instant alerts on the web application, while the processed data is periodically uploaded to the cloud for further analysis and long-term storage.

3.6.2 Deployment

3.6.2.1 Hardware Components

1. **ECG Sensors:** Configured AD8232 ECG sensor with ESP32 kit for ECG signal acquisition. Figure 3.9 is the designed ECG sensor which uses ESP32 kit and AD8232 ECG sensor to collect ECG from patient. We designed another ECG sensor to make a more compact version which is shown in Figure 3.10.

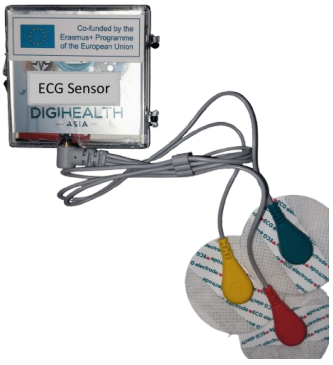

FIGURE 3.9: Designed Compact ECG Sensor

FIGURE 3.10: Designed ECG Sensor

2. **Edge Devices:**

   (a) **Smartphone:** High-performance smartphones with Android operating systems are used as edge device. Figure 3.11 is the Smartphone that is being used as edge device in our system.

FIGURE 3.11: Smartphone as Edge Device

   (b) **Raspberry Pi:** Model 4B is employed due to its efficient processing capabilities, low power consumption, and flexibility for IoT integration. Figure 3.12 is the Raspberry Pi kit being used as edge device.

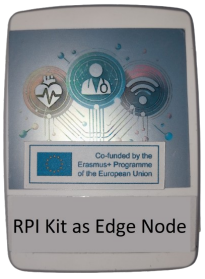

FIGURE 3.12: Raspberry Pi Kit as Edge Device

3.6.2.2 Software Components

1. **Data Processing and Denoising:**

   (a) Developed a web application using Python and PHP for denoising ECG data. The denoising pipeline includes wavelet transforms to isolate

signal frequencies, band-pass filters to eliminate irrelevant noise, and outlier detection algorithms to identify and remove transient anomalies.

(b) The implementation is done using Python libraries such as NumPy, SciPy, and PyWavelets.

2. **Deep Neural Network (DNN):**

(a) A convolutional neural network (CNN) architecture is trained on a labeled dataset of denoised ECG signals to classify patterns and detect arrhythmias.

(b) The trained model is optimized using TensorFlow Lite for deployment on edge devices.

3. **Smartphone Application:**

(a) Developed using Java, the application enables smartphone to act as edge device using MQTT protocol. Figure 3.13 is the interface of Smartphone edge application which is showing that edge device is currently receiving data from sensor node.

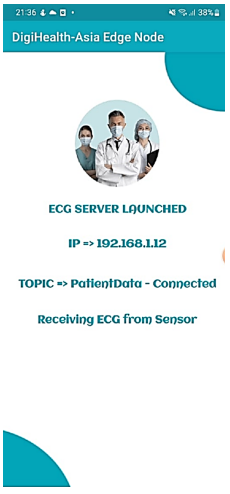

FIGURE 3.13: Smartphone as Edge Device Interface

4. **Cloud Integration:**

(a) Data that was aggregated at edge devices are sent to cloud for further long-term storing and to show patients ECG profile to doctors and other

paramedics staff. Figure 3.14 is the cloud interface for doctors, patients and paramedics staff and Figure 3.16 shows the list of registered patients alognwith their attached ECG sensor information. ECG sensor and be associated with patients by doctor or hospital admin. After doctor login, list of patients associated to that doctors are shown and doctor can view the profile of patient including ECG data.

Figure 3.17 is the cloud interface for managing ECG sensors. Super adminitrator or the device admin can login and add or remove sensor from database. These sensros are then attached to the patients to receive ECG data. Figure 3.15 is the cloud interface for adding or editing patient record. New patient can be added by hospital admin and they can also associate sensor and doctor to the patient while adding or editing the patient record.

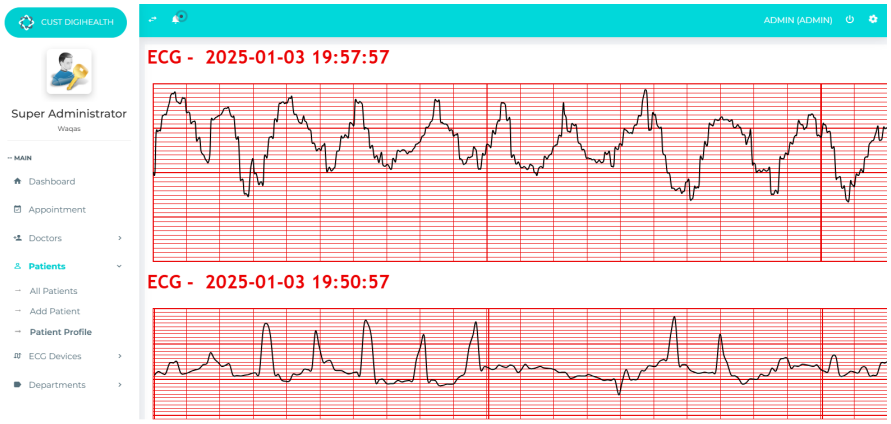

FIGURE 3.14: Cloud Interface for Doctors and Patients

3.6.3 Implementation Steps

3.6.3.1 ECG Signal Collection:

ECG signals are collected from sensors under controlled conditions to build a robust testing dataset.

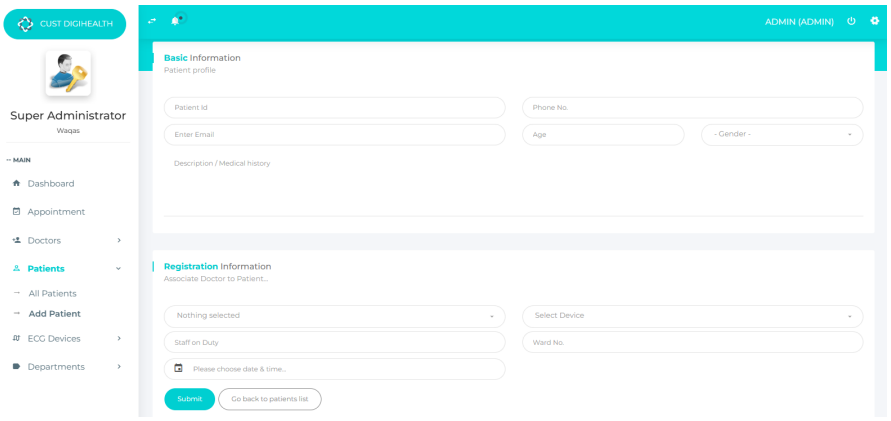

FIGURE 3.15: Cloud Interface for Adding New Patients With Sensor and Doctor Association

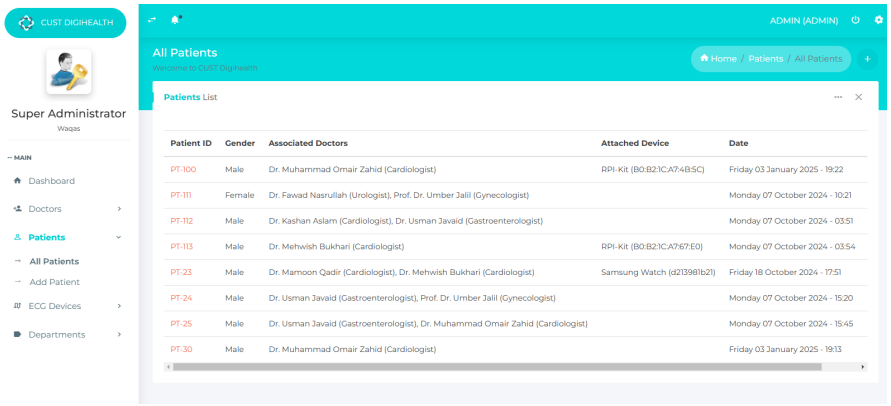

FIGURE 3.16: Cloud Interface - List of Patients alognwith their attached ECG sensors information

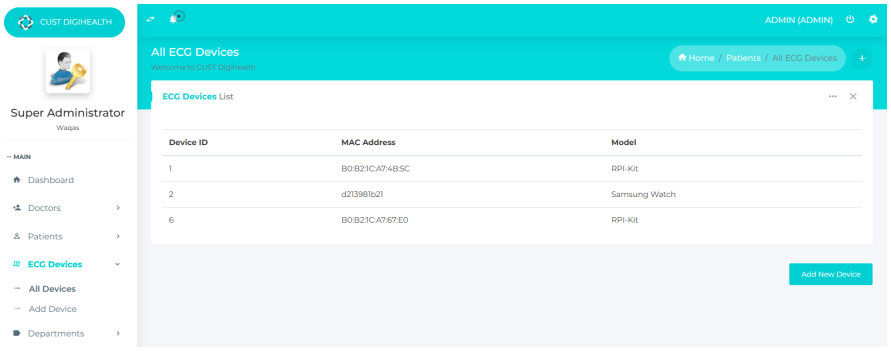

FIGURE 3.17: Cloud Interface - ECG Sensors Management

3.6.3.2 Model Training:

The DNN model is trained using supervised learning techniques, with validation to prevent overfitting.

3.6.3.3 Edge Device Deployment:

The trained DNN model is converted to a lightweight format compatible with edge devices and deployed on both smartphones and Raspberry Pi.

3.6.3.4 Performance Optimization:

Device-specific optimizations, such as threading and hardware acceleration, are implemented to improve inference speed and reduce energy consumption.

These edge devices were configured in real-time to test ECG signals that determine any possible abnormalities in real time. On ECG sensor bootup, it first searches for the edge devices on network with SSID "digihealth" and password "digihealth", first it looks for the RPI kit, if not found then it searches for smartphone and repeats the cycle until any edge device is found on the network. When connected, sensors makes the MQTT connection with edge MQTT broker and starts sending ECG data.

When ECG reading is complete means the sensor's power is removed, edge device stops receiving the ECG data from sensor and waits for 5 seconds considering if it is intentional disconnection or any error in connection, when 5 seconds are elapsed, edge devices denoises and slices the ecg in 360 sample sizes and passes it to the trained ML model which gives the inference. The system, in turn, could also create visual feedback for the healthcare provider and could alert them by sending notifications through web application deployed on cloud, so they could react promptly, as well.

3.7 Performance Evaluation Metrics

To evaluate the performance of smartphones and Raspberry Pis as edge devices, the following metrics were analyzed:

1. **Packet Delivery Ratio (PDR):** Measures the successful transmission of packets from sensor to edge device by measuring the percentage of packets successfully delivered from sensors to the edge device. A higher PDR indicates better reliability in communication.

2. **Packet Loss Ratio (PLR):** Evaluates the percentage of data packets lost during transmission. It is a key indicator of network stability and the effectiveness of the transmission process. Lower PLR values signify better performance.

3. **Throughput:** Determines the volume of data successfully transmitted and processed per unit of time. It serves as a critical performance indicator for the overall capacity and efficiency of the edge device in handling data.

4. **Delay:** Assesses the time taken for ECG data to reach to edge device and processed by it and then forwarded to the cloud. This metric is crucial for applications requiring real-time or near-real-time data processing, such as healthcare monitoring systems.

3.8 Conclusion

This cpahter detailed the design and implementation of an IoT-based healthcare system for real-time ECG monitoring by leveraging edge computing with smartphones and Raspberry Pi as edge devices. The chapter outlined the system architecture, data preprocessing techniques, and the development of a Convolutional Neural Network (CNN) model for ECG signal classification. Key steps included noise reduction, Z-score normalization, segmentation, upsampling, and label encoding to prepare the ECG data for model training. Class balancing techniques, such as SMOTE and NearMiss, were employed to address dataset imbalances, ensuring robust model performance.

The CNN model was trained using categorical cross-entropy loss and the Adam optimizer, achieving strong classification accuracy. The system was tested on

a self-collected ECG dataset, demonstrating the model's ability to generalize to real-world data. The deployment of the system involved hardware components like ECG sensors and edge devices, along with software components for data processing, denoising, and cloud integration. Performance evaluation metrics, including packet delivery ratio, packet loss ratio, throughput, and delay were used to assess the system's efficiency.

Chapter 4

Assessment and Results

4.1 Introduction

This presents the assessment and results of the proposed IoT-based ECG monitoring system, focusing on the performance evaluation of the CNN model and the edge devices. The chapter begins with a detailed analysis of the classification report and confusion matrix, providing insights into the model's accuracy, precision, recall, and F1-score. The loss and accuracy curves are examined to understand the model's learning behavior and generalization capabilities.

The chapter also discusses the results of denoising the self-collected ECG signals, highlighting the effectiveness of the preprocessing techniques in enhancing signal quality. Performance metrics such as packet delivery ratio, packet loss ratio, throughput, and delay are analyzed to evaluate the efficiency of smartphones and Raspberry Pi as edge devices. The findings reveal that Raspberry Pi achieves lower latency, while smartphones offer greater portability, making them suitable for different deployment scenarios.

Finally, the chapter concludes with a discussion of the trade-offs between processing power, energy consumption, and portability, providing valuable insights for optimizing IoT-based healthcare systems. The results underscore the potential of

edge computing in improving real-time ECG monitoring and enhancing patient care.

4.2 Classification Report and Confusion Matrix

Classification Report

In machine learning, a classification report is a performance evaluation metric that provides detailed insights into the classification results for tasks such as binary classification. It includes key metrics like precision, recall, F1-score, and support for each class, offering a comprehensive overview of the model's performance.

The classification report evaluates how well the model predicts each class and highlights any imbalances or misclassifications. This is essential for diagnosing issues with model performance and improving prediction accuracy.

The Figure 4.1 presents the classification report on test data using the CNN model:

classification report

	precision	recall	f1-score	support
N	0.9970	0.9980	0.9975	2015
A	0.9980	0.9970	0.9975	1985
accuracy			0.9975	4000
macro avg	0.9975	0.9975	0.9975	4000
weighted avg	0.9975	0.9975	0.9975	4000

FIGURE 4.1: Classification Report on Test Data using CNN

Confusion Matrix

The confusion matrix is a tabular representation of a model's binary classification performance, comparing predicted values with actual values. By analyzing the confusion matrix, one can assess the model's sensitivity (recall), specificity, and overall accuracy. This visualization is crucial for understanding misclassification patterns and areas for improvement.

The Figure 4.2 illustrates the confusion matrix generated using the CNN model:

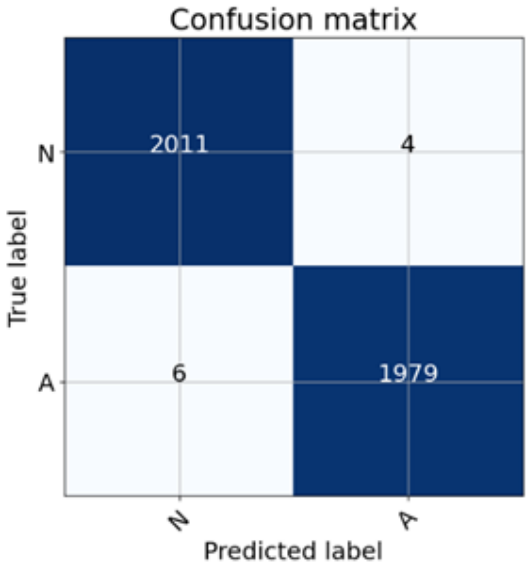

FIGURE 4.2: Confusion Matrix using CNN

The classification report and confusion matrix collectively provide a robust evaluation of model performance, guiding further refinement and optimization of the classification model.

4.2.1 Loss and Accuracy Curve

We plot the binary classification loss and accuracy curves for how our model did while training. These curves provide insights into the model's learning process over multiple epochs by tracking two essential metrics: loss and accuracy.

4.2.1.1 Loss

The loss curve shows that the model is learning from the training data, minimizing the error by improving. For such tasks known as binary classification tasks, a model typically optimizes for a loss function (e.g. binary cross entropy) to improve prediction accuracy.

A model is said to be learning effectively when it shows that loss curve decreases. On the contrary, if the loss stalls or starts to increase, this could be a sign of

overfitting or the possibility of an problem with the model's architecture. The Figure 4.3 illustrates the loss curve for both the training and test datasets:

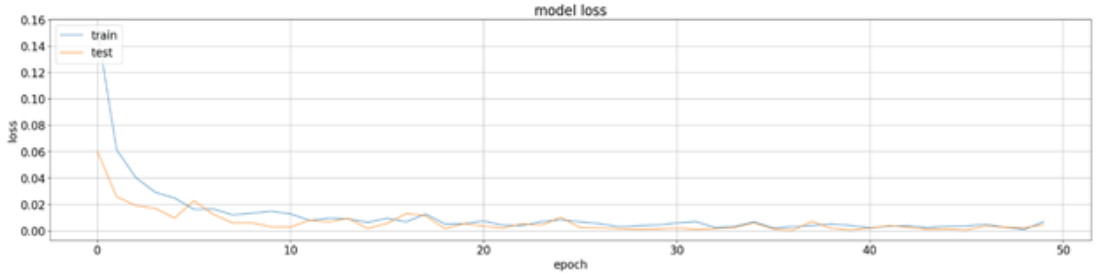

FIGURE 4.3: Loss Curve for Training and Test Datasets

4.2.1.2 Accuracy

Accuracy curve is a way to display how good your model classifies binary labels (0, 1 of course). Accuracy is the total number of correct predictions divided by total number of predictions. A curve rising measures the accuracy of the model implies that it is getting better at generalizing and properly classifying data.

Monitoring the accuracy curve over epochs helps identify if the model continues to improve or reaches a saturation point. A flattening or decreasing accuracy curve may indicate that the model requires further tuning. The Figure 4.4 shows the accuracy curve for the training and test datasets:

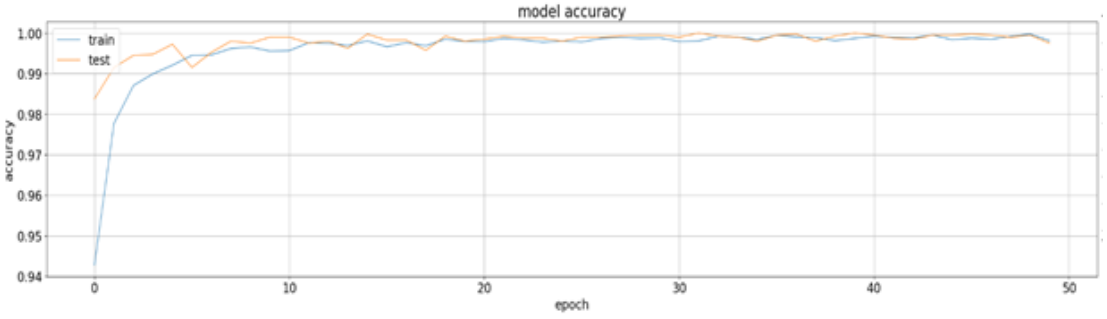

FIGURE 4.4: Accuracy Curve for Training and Test Datasets

These curves are crucial in evaluating the model's performance and diagnosing potential issues during training, ensuring the model achieves optimal generalization on unseen data.

4.3 Actual vs Denoised Signal Results

By using interface mentioned in Figure 3.8, we denoised the self collected ECG signal, results of denoising are showin in Figure 4.5 which shows that QRS complexes are clearly visible after denoising.

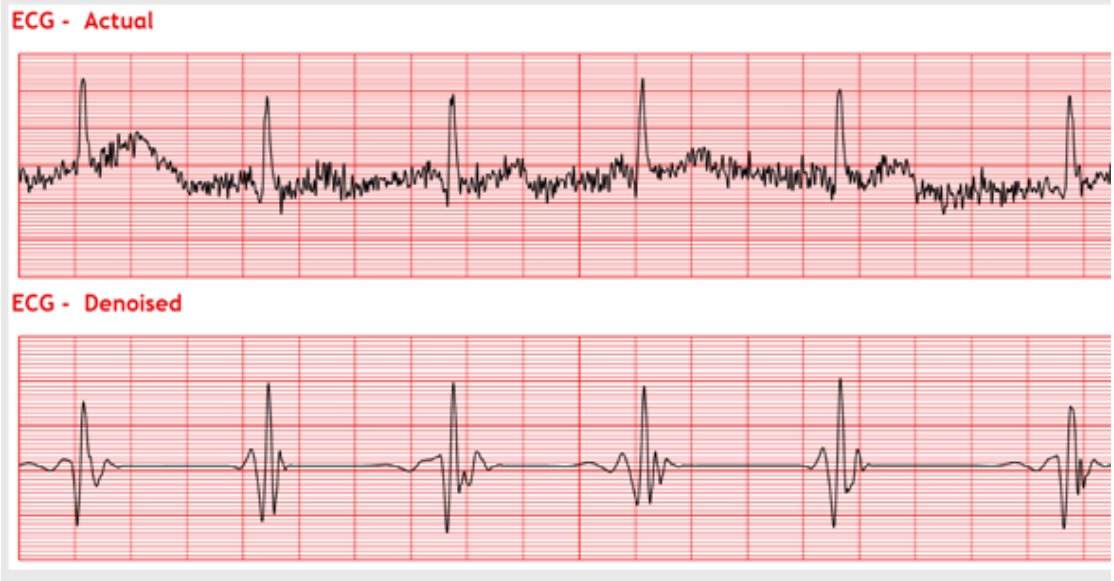

FIGURE 4.5: Actual and Denoised Signal

The model demonstrated strong predictive performance, achieving an accuracy of 74% on the self-collected ECG dataset. This result highlights the CNN model's capability to recognize patterns in abnormal ECG signals effectively.

The classification report on the test data confirmed the model's ability to generalize and classify ECG segments accurately, further validating its robustness and reliability for real-world applications.

The Figure 4.6 presents the classification report, showcasing the achieved accuracy on the self-collected dataset:

The successful application of the model to our own dataset demonstrates the effectiveness of CNNs in detecting abnormal ECG patterns, reinforcing the potential for clinical application and further research.

```
Normal ECG Samples : 500

Abnormal ECG Samples : 1394
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.74      0.85      1894

    accuracy                           0.74      1894
   macro avg       0.50      0.37      0.42      1894
weighted avg       1.00      0.74      0.85      1894
```

FIGURE 4.6: Classification Report on Self-Collected ECG Dataset

4.3.1 Performance Analysis

Raspberry Pi was shown to achieve lower delays than smartphone, with average. latencies of 70–120 ms for the Raspberry Pi and 100–400 ms (up to 1 s peak delays) for the smartphone under high traffic conditions. Table 4.1 shows the minimum, maximum and average delays in single transmission of ECG data from sensor to both edge devices. This difference is because of smartphones being so multitasking and there being many background processes competing for resources.

TABLE 4.1: Delay Measurements of RPI and Smartphone Edge Devices

Edge Device	Lowest Delay	Highest Delay	Avg. Delay	Readings
RPI	12 ms	684 ms	89 ms	35
Smartphone	660 ms	3382 ms	534 ms	35

The Figure 4.7 show the delay comparison between both edge devices. Highest delay of RPI device was around 700 ms and average delay for RPI was less than 100 ms where as for smartphone, highest delay was around 3400 ms and average delay was around 530 ms which shows that there is a significantly higher delay on Smartphone edge as compared to RPI.

On both these platforms, the packet delivery and loss ratios performed consistently. Throughput measurements showed that both devices were capable of processing data rates required for real-time ECG monitoring without drastic performance degradation. Throughput was measure to be between 780 bps to 820 bps on both devices and the average of througput over 35 readings on Smartphone resulted in 807 bps and on Raspberry Pi, it resulted in 815 bps.

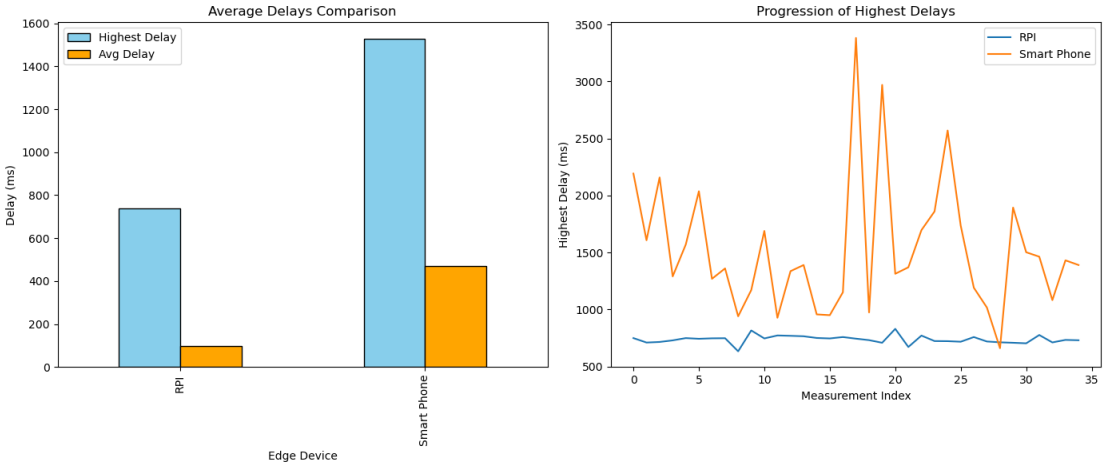

FIGURE 4.7: Delay comparison of Smartphone and RPI as Edge Devices

The energy consumption turned out to be an interesting observation. Even though Raspberry Pi is a fine machine in terms of processing, it needed more battery and was power hungry, because when the smartphone is not in use, it can put the edge application to sleep mode. The trade-off between processing power and energy was crucial for determining the appropriate deployment scenario per device.

4.4 Discussion of Findings

This increase in delay on smartphones is primarily due to computational resource contention from running multiple applications concurrently on the device.Smartphones, as multi-purpose devices, are designed to run variety of applications at once. This resource excess is usually found in the form of Processing power being shared over several different tasks that can induce latency in applications that require high-priority/real-time computations like ECG data processing is.

On the other hand, towards a single, dedicated task, the Raspberry Pi displays consistently low latencies. Raspberry Pi do not experiences identical resource contention in regards to the processing of the ECG data like a smartphone would when configured. By limiting its processing to just ECG data, computational

resources are completely used for this purpose, allowing for very low latencies and ultimately real-time system performance.

But even though the Raspberry Pi outperforms in maintaining low latencies, the inherent portability of smartphones means that they are the ideal candidate for patient-centric deployments, where proximity and ease of access are paramount. Smartphones are small, light, and ubiquitous devices that patients can carry with them during their daily routine. This is crucial for continuous monitoring applications, as it allows the patient to incorporate monitoring into their daily life without the need for additional hardware. So, even if a Raspberry Pi guarantees performance reliability, it cannot compete with a smartphone for patient-focused deployments, which require portability and convenience.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this study, we evaluated the smartphones and Raspberry Pi as edge devices for an edge-centric IoT based ECG monitoring system in terms of feasibility and performance and provide important insights on their suitability for such smart health applications. Complete pipeline of IoT based healthcare system was implemented by designing and developing a prototype which includes the design of ECG sensor, configuration of RPI and development of an android application that enables smartphone to act as edge device. Web interface for doctors, patients and paramedics was also developed and deployed on cloud. Doctors can login the system and can view the patient profile including the ECG values sent by the edge node to cloud.

Multiple ECG reading were taken on both edge devices to evaluate the results. The results of findings show that smartphones as edge resulted in comparatively higher delays as compared to Raspberry Pi, however, their wide use and user friendly interfaces can make them available and practical for many of the healthcare applications. With the fact that smartphones are by now so widespread and so familiar to users the latter are likely to choose them first if portability and ease of use are the prime considerations.

However, the most significant results in delay variance made Raspberry Pi perform better than Smartphone in a critical metric like the delay variance and is hence a reliable and more efficient solution for real time ECG monitoring. It is especially suited for applications requiring the robustness of a network as well as timely transmission of data as are specialty healthcare emergency systems and continuous monitoring in clinical environments because its lower delays and higher processing efficiency. Nevertheless, its spread may be restricted by the prerequisite of technical expertise needed for setup and upkeep, in some areas that may not even have specialized technical support.

We further study the importance of denoising the signal to improve ECG inference accuracy and show that a deep neural network trained using the denoised dataset is successfully deployed on both device use cases. The results show that the edge device selection is very important and has to be carefully taken with respect to the function to be performed in the healthcare application, i.e. processing speed needs, wireless networking reliability, user accessibility, and trade-off between technical complexity and performance.

This research provides important insights for healthcare professionals, developers and researchers related to IoT based healthcare services. To help with the wider adoption of IoT technologies, the study makes it possible for trade-offs between the efficiency and reliability of a Raspberry Pi, but enhanced user friendliness and ubiquity of a smartphone to be identified. This work lays a foundation for improving real-time monitoring systems and optimizing edge device deployment to improve healthcare delivery.

5.2 Future Research Avenues

Based on the insights from this study, potential future research avenues to take edge-centric IoT-based healthcare systems a step further towards feasibility and improved performance are outlined below:

1. **Evaluating Sensor Communication Channels:** Future studies should assess the communication channels between health sensors and edge devices, especially via Bluetooth. Evaluating these channels will provide insights into their impact on latency, reliability, and data integrity, particularly in real-time monitoring applications.

2. **Implementation of Federated Learning:** Federated learning can be implemented. It enables edge devices to collaboratively local model training with preserving privacy of data. This method would enable a federated secure and distributed training of models leveraging data from different data sources to enhance model performance.

3. **Integration of Multiple Health Sensors:** Testing the system's ability to handle higher data loads originating from multiple health sensors is essential. This includes simulating real-world use cases such as multi-parameter monitoring (e.g., ECG, SpO_2, and temperature) to evaluate the system's feasibility and scalability in more demanding scenarios.

4. **Enhanced Signal Processing:** Investigating advanced signal denoising techniques and developing more robust denoising algorithms can substantially improve the overall system's effectiveness. Implementation of DNN based signal denoising techniques could further improve the signal quality by only removing the noise component in signal without attenuating or removing useful information from ECG signal.

5. **Scalability and Multi-User Data Handling:** Observing the system's scalability to process and deliver data from multiple users simultaneously is crucial.

6. **Real-World Field Trials:** Conducting extensive field trials in diverse healthcare environments will help identify practical challenges, validate system reliability, and assess performance under varying conditions.

Addressing these aspects in future research will lead to a better understanding of smart healthcare system deployment and associated edge device optimization.

Bibliography

[1] R. K. Pathinarupothi, P. Durga, and E. Rangan, "Iot-based smart edge for global health: Remote monitoring with severity detection and alerts transmission," *Institute of Electrical and Electronics Engineers*, vol. 6, no. 2, pp. 2449–2462, Sep. 2018. [Online]. Available: https://doi.org/10.1109/jiot.2018.2870068

[2] R. Dave, N. Seliya, and N. Siddiqui, "The benefits of edge computing in healthcare, smart cities, and iot," *Journal of Computer Science and Applications*, vol. 9, no. 1, pp. 23–34, Oct. 2021. [Online]. Available: https://doi.org/10.12691/jcsa-9-1-3

[3] K. Jaiswal, S. Sobhanayak, A. K. Turuk, S. L. Bibhudatta, B. K. Mohanta, and D. Jena, "An iot-cloud based smart healthcare monitoring system using container based virtual environment in edge device," *IEEE International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, pp. 1–7, Jul. 2018. [Online]. Available: https://doi.org/10.1109/icetietr.2018.8529141

[4] A. Cazañas-Gordón and E. Parra-Mora, "The internet of things in healthcare. an overview," *European Organization for Nuclear Research*, 07 2020. [Online]. Available: https://zenodo.org/record/5730386

[5] HioTron, "How edge computing helps secure iot," 2025. [Online]. Available: https://www.hiotron.com/how-edge-computing-help-secure-iot/

[6] M. Satyanarayanan, "Edge computing," 01 2025. [Online]. Available: https://ieeexplore.ieee.org/document/8057308/

[7] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge computing security: State of the art and challenges," 09 2023. [Online]. Available: https://ieeexplore.ieee.org/document/8741060/

[8] B. Dezfouli and Y. Liu, "Editorial: Special issue "edge and fog computing for internet of things systems"," pp. 4387–4387, 06 2022. [Online]. Available: https://doi.org/10.3390/s22124387

[9] X. Du, X. Chen, Z. Lu, Q. Duan, Y. Wang, J. Wu, and P. C. K. Hung, "A blockchain-assisted intelligent edge cooperation system for iot environments with multi-infrastructure providers," *Institute of Electrical and Electronics Engineers*, vol. 10, no. 24, pp. 21 227–21 239, 06 2023. [Online]. Available: https://doi.org/10.1109/jiot.2023.3282954

[10] R. Dave, N. Seliya, and N. Siddiqui, "The benefits of edge computing in healthcare, smart cities, and iot," vol. 9, no. 1, pp. 23–34, 10 2021. [Online]. Available: https://doi.org/10.12691/jcsa-9-1-3

[11] R. Latif, M. U. Ahmed, S. Tahir, S. Latif, W. Iqbal, and A. Ahmad, "A novel trust management model for edge computing," *Springer Science+Business Media*, vol. 8, no. 5, pp. 3747–3763, 09 2021. [Online]. Available: https://doi.org/10.1007/s40747-021-00518-3

[12] S. Hadjixenophontos, A. M. Mandalari, Y. Zhao, and H. Haddadi, "Prism: Privacy preserving healthcare internet of things security management," pp. 1–5, 07 2023. [Online]. Available: https://doi.org/10.1109/iscc58397.2023.10218268

[13] Z. Xu, D. He, P. Vijayakumar, B. B. Gupta, and J. Shen, "Certificateless public auditing scheme with data privacy and dynamics in group user model of cloud-assisted medical wsns," *Institute of Electrical and Electronics Engineers*, vol. 27, no. 5, pp. 2334–2344, 11 2021. [Online]. Available: https://doi.org/10.1109/jbhi.2021.3128775

[14] F. McNamee, S. Dustadar, P. Kilpatrick, W. Shi, I. Spence, and B. Varghese, "A case for adaptive deep neural networks in edge computing," 01 2020. [Online]. Available: https://arxiv.org/abs/2008.01814

[15] S. Shaikh and V. Chitre, "Healthcare monitoring system using iot," 05 2017. [Online]. Available: https://doi.org/10.1109/icoei.2017.8300952

[16] M. J. Baucas, P. Spachos, and S. Gregori, "Internet-of-things devices and assistive technologies for health care: Applications, challenges, and opportunities," *Institute of Electrical and Electronics Engineers*, vol. 38, no. 4, pp. 65–77, 06 2021. [Online]. Available: https://doi.org/10.1109/msp.2021.3075929

[17] M. Boyer, L. J. Bouyer, J. Roy, and A. Campeau-Lecours, "Reducing noise, artifacts and interference in single-channel emg signals: A review," pp. 2927–2927, Mar. 2023. [Online]. Available: https://doi.org/10.3390/s23062927

[18] R. Badiger and M. Prabhakar, "ASCNet-ECG: Deep Autoencoder based Attention aware Skip Connection network for ECG filtering," *International Journal of Engineering Trends and Technology*, vol. 71, no. 2, pp. 382–398, Feb. 2023. [Online]. Available: https://doi.org/10.14445/22315381/ijett-v71i2p240

[19] Z. F. M. Apandi, R. Ikeura, S. Hayakawa, and S. Tsutsumi, "An analysis of the effects of noisy electrocardiogram signal on heartbeat detection performance," *Bioengineering*, vol. 7, no. 2, pp. 53–53, Jun. 2020. [Online]. Available: https://doi.org/10.3390/bioengineering7020053

[20] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun, "Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review," 01 2020. [Online]. Available: https://arxiv.org/abs/2001.01550

[21] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Elsevier BV*, vol. 12, pp. 100 273–100 273, 08 2020. [Online]. Available: https://doi.org/10.1016/j.iot.2020.100273

[22] M. Arnold, J. Boston, M. Desmond, E. Duesterwald, B. Elder, A. Murthi, J. Navrátil, and D. Reimer, "Towards automating the ai operations lifecycle," 01 2020. [Online]. Available: https://arxiv.org/abs/2003.12808

[23] M. Wazid, P. R. Dsouza, A. K. Das, V. B. K, N. Kumar, and J. J. P. C. Rodrigues, "Rad-ei: A routing attack detection scheme for edge-based internet of things environment," *Wiley*, vol. 32, no. 15, 08 2019. [Online]. Available: https://doi.org/10.1002/dac.4024

[24] T. Luciano Musumeci, Politecnico di Torino, "A comparative analysis of adaptive notch filtering and wavelet mitigation against jammers interference." [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/navi.167

[25] X. Huang, H. Dong, Q. Tao, M. Yu, Y. Li, L. Rong, H. Krause, A. Offenhäusser, and X. Xie, "Sensor configuration and algorithms for power-line interference suppression in low field nuclear magnetic resonance," *Multidisciplinary Digital Publishing Institute*, vol. 19, no. 16, pp. 3566–3566, 08 2019. [Online]. Available: https://doi.org/10.3390/s19163566

[26] J. González–Ramos, I. Angulo, I. Fernández, A. Arrinda, and D. Vega, "Characterization of the potential effects of emc filters for power converters on narrowband power line communications," *Multidisciplinary Digital Publishing Institute*, vol. 10, no. 2, pp. 152–152, 01 2021. [Online]. Available: https://doi.org/10.3390/electronics10020152

[27] K. S. Erer, "Adaptive usage of the butterworth digital filter," *Elsevier BV*, vol. 40, no. 13, pp. 2934–2943, 01 2007. [Online]. Available: https://doi.org/10.1016/j.jbiomech.2007.02.019

[28] D. G. Tiglea, R. Candido, and M. T. M. Silva, "A low-cost algorithm for adaptive sampling and censoring in diffusion networks," 01 2020. [Online]. Available: https://arxiv.org/abs/2008.02624

[29] Y. Yu and P. Loskot, "Polynomial distributions and transformations," 01 2022. [Online]. Available: https://arxiv.org/abs/2212.04865

[30] O. Giustolisi and D. Savić, "A symbolic data-driven technique based on evolutionary polynomial regression," *IWA Publishing*, vol. 8, no. 3, pp. 207–222, 07 2006. [Online]. Available: https://doi.org/10.2166/hydro.2006.020b

[31] D. W. Hogg and S. Villar, "Fitting very flexible models: Linear regression with large numbers of parameters," *Institute of Physics*, vol. 133, no. 1027, pp. 093 001–093 001, 09 2021. [Online]. Available: https://doi.org/10.1088/1538-3873/ac20ac

[32] S. Scholl, "Fourier, gabor, morlet or wigner: Comparison of time-frequency transforms," 01 2021. [Online]. Available: https://arxiv.org/abs/2101.06707

[33] L. P. Arts and E. L. van den Broek, "The fast continuous wavelet transformation (fcwt) for real-time, high-quality, noise-resistant time–frequency analysis," *Nature Portfolio*, vol. 2, no. 1, pp. 47–58, 01 2022. [Online]. Available: https://doi.org/10.1038/s43588-021-00183-z

[34] F. Romero, D. C. Piñol, and C. R. V. Seisdedos, "Deepfilter: an ecg baseline wander removal filter using deep learning techniques," 01 2021. [Online]. Available: https://arxiv.org/abs/2101.03423

[35] ——, "Deepfilter: An ecg baseline wander removal filter using deep learning techniques," *Elsevier BV*, vol. 70, pp. 102 992–102 992, 07 2021. [Online]. Available: https://doi.org/10.1016/j.bspc.2021.102992

[36] J. N. John, C. Galloway, and A. Valys, "Deep convolutional neural networks for noise detection in ecgs," 01 2018. [Online]. Available: https://arxiv.org/abs/1810.04122

[37] P. M. Tripathi, A. Kumar, R. Komaragiri, and M. Kumar, "A review on computational methods for denoising and detecting ecg signals to detect cardiovascular diseases," *Archives of Computational Methods in Engineering*, vol. 29, no. 3, pp. 1875–1914, May 2022. [Online]. Available: https://doi.org/10.1007/s11831-021-09642-2

[38] J. Lee, J. Oh, D. Kwon, M. Kim, K. Kim, and Y. Park, "Blockchain-enabled key aggregate searchable encryption scheme for personal health record sharing with multidelegation," *Institute of Electrical and Electronics Engineers*, vol. 11, no. 10, pp. 17 482–17 494, 01 2024. [Online]. Available: https://doi.org/10.1109/jiot.2024.3357802

[39] B. Al-Shargabi and S. Abuarqoub, "Iot-enabled healthcare: Benefits, issues and challenges," in *Proceedings of the [Conference Name]*, 11 2020, pp. 1–5. [Online]. Available: https://doi.org/10.1145/3440749.3442596

[40] T. Muazu, Y. Mao, A. U. Muhammad, M. bin Ibrahim, O. Samuel, and P. Tiwari, "Iomt: A medical resource management system using edge empowered blockchain federated learning," *Institute of Electrical and Electronics Engineers*, vol. 21, no. 1, pp. 517–534, 08 2023. [Online]. Available: https://doi.org/10.1109/tnsm.2023.3308331

[41] S. Hong, Y. Zhou, J. Shang, C. Xiao, and J. Sun, "Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review," *Elsevier BV*, vol. 122, pp. 103 801–103 801, 06 2020. [Online]. Available: https://doi.org/10.1016/j.compbiomed.2020.103801

[42] G. B. Moody and R. G. Mark, "The MIT-BIH arrhythmia database," PhysioNet, 1992. [Online]. Available: https://www.physionet.org/content/mitdb/1.0.0/

[43] M. Qi, H. Shao, N. Shi, G. Wang, and Y. Lv, "Arrhythmia classification detection based on multiple electrocardiograms databases," *PloS One*, vol. 18, no. 9, p. e0290995, 2023. [Online]. Available: https://doi.org/10.1371/journal.pone.0290995