



DigiHealth-Asia

D2.4 – Pilot use case: cardiovascular patient monitoring, implementation and training

Work package	WP2
Task	Task D2.4
Due date	9 th September 2024
Submission date	3rd December 2024
Deliverable lead	NUST and UoH
Version	0.1
Authors	NUST CUST and UoH
Reviewers	Timo De Waele (UGent)

Table of Contents

Contents

1. Introduction	7
1.1. Key Similarities in CUST and NUST Cardiovascular Pilot Projects	7
2. Pilot case implementation at CUST	9
2.1. Introduction	9
3. System Architecture	10
3.1. Stage 1: Sensing Network	11
3.2. Stage 2: Edge Computation	13
3.3. Stage 3: Cloud Computing	14
3.4. Stage 4: GUIs (Web interface)	15
3.5. Communication between system components.....	16
4. Data Collection.....	17
4.1. Patient Demographics and Number of Patients.....	17
4.2. Informed Consent	17
4.3. Duration of ECG	18
4.4. Data Collection Equipment.....	19
4.5. Data Labelling	21
5. Model Training.....	23
5.1. Extracted and Pre-processed Data:	24
5.2. Rebalanced Classes.....	25
5.3. Train-Test Split:.....	26
5.4. Model Training using CNN	27
5.5. Model Parameters	27
5.6. Loss and Accuracy curve	28
5.7. Classification Report and Confusion Matrix	29
5.8. Training SVM Model:	30
5.9. Testing of Model on Self Collected ECG Dataset.....	32
6. Future Advancements on the Pilot Case	32
6.1. Thorough Investigation of Other Sensors for Improved ECG Monitoring.....	32
6.2. Movement of Cloud Services Towards End Devices.....	32

6.3.	Long-Term ECG data Analysis	33
6.4.	Privacy Preservation Using Machine Learning	33
7.	Pilot case implementation at NUST	34
8.	System Architecture-NUST	35
8.1.	Hardware Enclosure	36
8.2.	Embedded System Integration	36
	Raspberry Pi 4	37
	Protocentral Max 86150 ECG Sensor	39
	7-inch HDMI Touchscreen:	40
8.3.	GUI Software Development.....	41
	Web Dashboard Portal	43
8.4	ECG Data Processing	45
	Visualization and Noise Removal.....	46
	Peak Detection.....	49
	Feature Window Formation	51
	Class Imbalance	53
8.5.	Convolution Neural Network:.....	55
	First Convolutional Layer:	56
	Second Convolutional layer:	56
	Activation Function:.....	57
	Evaluation Metric.....	57
9.	Results and Discussion	58
	Training and Validation Accuracy:	59
10.	Conclusion and Future Work	62
10.1.	Conclusion.....	62
10.2.	Future Work.....	62
11.	Executive Summary	6
12.	References	64

Table of Figures

Figure 1: System Architecture Diagram.....	10
Figure 2: The 4 stages of Architecture.....	11
Figure 3: AD8232 sensor.....	11
Figure 4: MAX30102 sensor.....	12
Figure 5: MLX90614 sensor	12
Figure 6: ESP32 Microcontroller	13
Figure 7: Raspberry PI (Edge Device).....	14
Figure 8: Graphical User Interfaces	15
Figure 9: Hospitals visited for data collection	17
Figure 10: Consent form, Filled and signed	18
Figure 11: Consent form for patients	18
Figure 12: Shimmer Device for data collection	19
Figure 13: AS7038R vital sign sensor	20
Figure 14: Data collection through AS7038R.....	20
Figure 15: Actual vs De-Noised signal.....	21
Figure 16: Noise removal from ECG	22
Figure 17: Admin panel of annotation tool	22
Figure 18: Doctor Panel	23
Figure 20: Complete Methodology Diagram	35
Figure 21: Flow Diagram of the Methodology Followed	35
Figure 22: Prototype Version 1.....	36
Figure 23: Final design of ECG Device (Version 2)	37
Figure 24 Version 3 of ECG Device	37
Figure 25: Real Time ECG Visualization GUI	38
Figure 26: Flow Diagram of Sensor Integration with ESP 32	39
Figure 27: Protocentral Max 86150 ECG Sensor	40
Figure 28: Lead II Configuration for RA, LA, RL.....	40
Figure 29: 7 inch touchscreen	41
Figure 30: ECG Visualization and Annotation Tool.....	41
Figure 31: ECG Visualization Tool	43
Figure 32: Web Dashboard Portal Homepage.....	43
Figure 33: Web Dashboard Portal	45
Figure 34 ECG Signal Visualization.....	46
Figure 35 : ECG Signals Visualization RAW Signals Unfiltered.....	47
Figure 36: ECG Signals Visualization RAW and Filtered ECG Signals	47
Figure 37 ECG Signal Visualization.....	48
Figure 38: ECG Signals detected peaks.....	49
Figure 39 : Peak detection using local maxima	50
Figure 40: Detected Peaks of ECG Signal for 10k to 20k ms interval	51
Figure 41: Unfiltered Windows ECG Signals	52
Figure 42: Filtered Windows ECG Signals	52
Figure 43: SMOTE Algorithm Diagram.....	53

Figure 44: ECG Data Class Distribution without SMOTE.....	54
Figure 45 Class distribution after SMOTE	55
Figure 46: Training Curve generated for Accuracy with CNN for Classification	58
Figure 47: Training Curve generated for Loss with CNN for Classification.....	59
Figure 48: Confusion Matrix Plotted for the results	60

1. Executive Summary

The deliverable D4.2 is a deliverable of work package 4 “Dissemination and exploitation” and is related to the task T4.2 “Communication through traditional media”. The main objective of this deliverable is to show that the project logo and project website has been set up. The website will be used as the primary point of online dissemination of the project activities and the results that will be generated during the project.

2. Introduction

This report presents a comprehensive analysis of the implementation and training processes for an advanced cardiovascular patient monitoring system, designed to enhance patient care through real-time data tracking and early intervention. Two pilot test cases were conducted at prestigious institutions: Capital University of Science and Technology (CUST) and National University of Sciences and Technology (NUST). These pilots aimed to rigorously assess the system's performance, reliability, and impact on patient health outcomes in clinical settings. At both universities, the monitoring system was integrated into existing healthcare workflows, where it was evaluated for its accuracy in detecting cardiovascular anomalies, ease of use, and the effectiveness of training programs for medical staff. By closely monitoring patient data, these trials provided valuable insights into the system's scalability, user adaptability, and its potential to revolutionize cardiovascular care. The findings from CUST and NUST are crucial for shaping future implementations in hospitals and healthcare institutions on a larger scale.

2.1. Key Similarities in CUST and NUST Cardiovascular Pilot Projects

The CUST and NUST pilot cases share several key commonalities despite using different datasets and environments for cardiovascular monitoring. Both projects aim to improve real-time cardiac health monitoring through the integration of IoT, machine learning, and cloud computing. The summary of key findings are hereunder:

1.1.1. Objective of Cardiovascular Monitoring

Both pilots focus on real-time monitoring and early detection of cardiovascular abnormalities using non-invasive techniques to enhance healthcare outcomes.

CUST: Focuses on real-time monitoring and early detection of cardiovascular abnormalities using machine learning and IoT.

NUST: Centers around a smart ECG device aimed at improving cardiac monitoring through real-time data visualization and advanced signal analysis.

1.1.2. System Architecture and IoT Use

Each project uses a multi-stage architecture that integrates sensors (like ECG), microcontrollers (ESP32), edge computing (Raspberry Pi), and cloud services for data collection and processing.

CUST: Uses sensors like AD8232 (ECG), MAX30102 (heart rate and SpO2), and MLX90614 (temperature), interfaced with an ESP32 microcontroller. Data is processed at the edge (Raspberry Pi) before being sent to the cloud.

NUST: Integrates the ESP32 microcontroller with the Protocentral ECG sensor and Raspberry Pi, with a focus on real-time ECG data acquisition and processing.

1.1.3. Machine Learning for Data Analysis

Both pilots employ machine learning models, such as convolutional neural networks (CNNs), to analyze collected ECG data and detect abnormalities in heart function.

CUST: Utilizes convolutional neural networks (CNNs) and support vector machines (SVMs) for classifying normal and abnormal heartbeats.

NUST: Leverages machine learning algorithms, including convolutional neural networks, to detect cardiac anomalies in the ECG data

1.1.4. Edge and Cloud Computing Integration

Data is pre-processed at the edge (Raspberry Pi) and sent to the cloud for long-term storage and more intensive processing tasks, including model training.

CUST: Data pre-processed at the edge device (Raspberry Pi) is transmitted to the cloud for more computationally intensive tasks like model training.

NUST: Real-time ECG data is processed by the Raspberry Pi for immediate feedback and sent to the cloud for storage and further analysis.

1.1.5. Graphical User Interface (GUI)

Both systems provide user-friendly GUIs for healthcare professionals and patients to visualize real-time data and monitor patient health remotely.

CUST: Offers a web-based interface where users (doctors, patients, and admins) can access patient data, visualized in real-time.

NUST: Developed a real-time visualization tool and web dashboard for displaying ECG signals and results.

1.1.6. Data Collection from Hospitals:

Data collection for both pilots was done in collaboration with hospitals, though each project worked with different institutions.

CUST: Data was collected from 75 patients at AFIC (Armed Forces Institute of Cardiology, Pakistan)

NUST: Focused on indigenous data collection using their developed smart ECG device

1.1.7. Focus on Class Imbalance and Data Annotation:

Both projects address the class imbalances in their datasets using resampling techniques like Synthetic Minority Over-sampling (SMOTE) and have robust annotation processes for accurate data labeling and training.

CUST: Used SMOTE and other techniques to rebalance classes, ensuring a balanced dataset for effective machine learning.

NUST: Also applied the SMOTE algorithm to address class imbalance in ECG data and developed an annotation tool for labeling the data.

2. Pilot case implementation at CUST

The pilot case “Cardiovascular patient monitoring system” implemented at Capital University of Science and Technology (CUST) is responsible for the development of a prototype system in Pakistan for monitoring of cardiovascular patients.

2.1. Introduction

Cardiovascular diseases (CVDs) pose a significant health challenge in Pakistan, contributing significantly to the country's mortality rate. The limitations of traditional healthcare systems, including limited access to facilities, high costs, and a shortage of skilled professionals, worsen this problem. An autonomous monitoring framework, powered by machine learning, can track vital signs in real-time enabling continuous monitoring of patient health. The system can also facilitate early intervention allowing for timely medical intervention when necessary. This will improve patient outcomes, reducing the risk of complications and improving overall health.

Traditional monitoring devices can be expensive, making continuous monitoring financially challenging for many patients. A cost-effective alternative can make this essential service accessible to a larger segment of the population. By providing real-time, data-driven insights, this system can also enhance decision-making by assisting healthcare providers in making informed treatment decisions.

The following sections outline the system architecture including end to end system, sensors and equipment used in developing the system for heart health monitoring. Through strategic planning and collaboration with medical experts, the system seeks to utilize cutting-edge technology and data-driven insights to improve patient outcomes, particularly in underserved populations. From system architecture to data collection methods, every element of this pilot case has been carefully designed to ensure precision and reliability in achieving the research goals.

3. System Architecture

This section outlines the architecture of the cardiovascular patient monitoring system, designed to provide real-time monitoring and early detection of health anomalies in patients with cardiovascular conditions. The system leverages advanced technologies such as IoT, edge computing, and machine learning to ensure efficient data collection, processing, and analysis. [1]

The diagram below presents the architecture of the system. The system architecture provides the complete flow from the patient to the end system and consists of multiple components where each component has a specific role assigned to it. Overall, the system architecture divides the system's workflow into 4 different stages as defined by the following diagram.

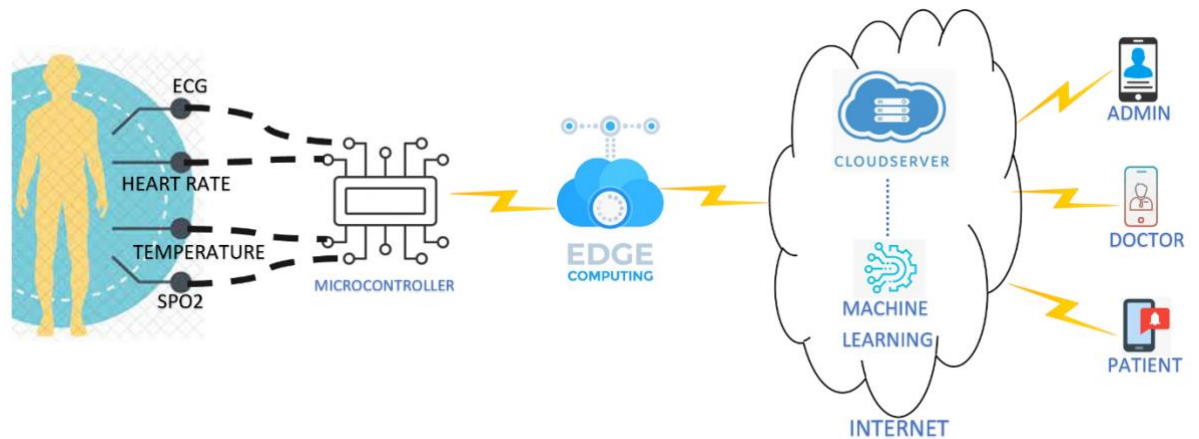


Figure 1: System Architecture Diagram

The four major stages of the system architecture are the sensing network, edge computation, cloud computing and the GUIs. The 4 stages of architecture are defined by the diagram below.

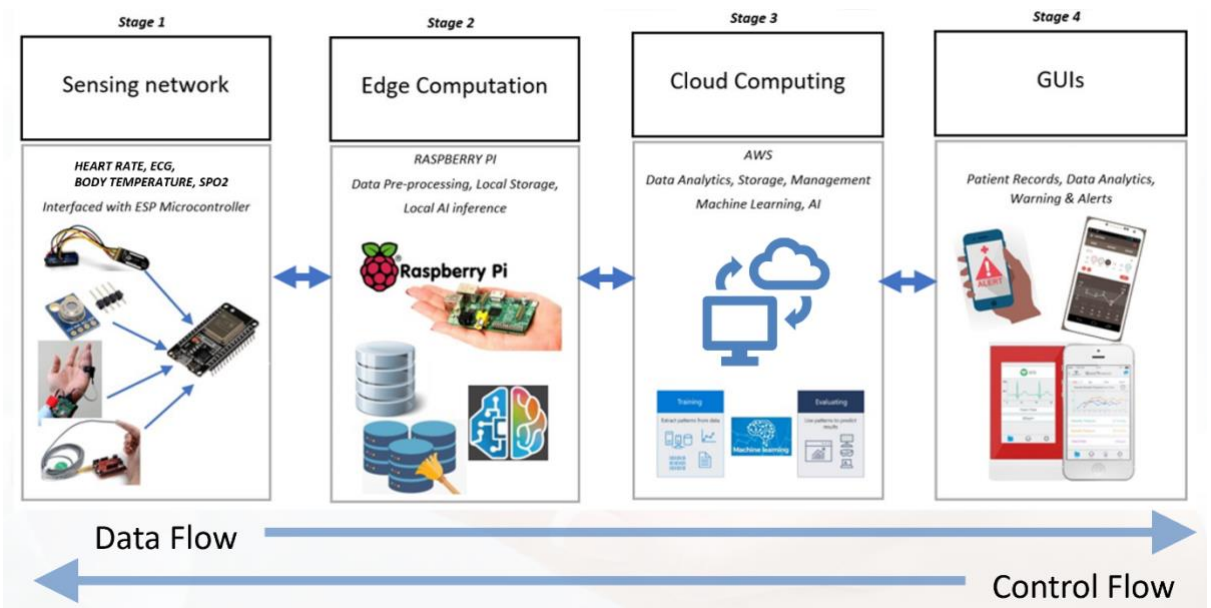


Figure 2: The 4 stages of Architecture

3.1. Stage 1: Sensing Network

In stage 1 i.e., the sensing Network, wearable sensors have been interfaced with a microcontroller. The main role of the sensing network is to collect the vitals of the patient and transfer the data to the edge node. The sensors used are as follows.

3.1.1. AD8232 sensor (ECG sensor)



Figure 3: AD8232 sensor

The AD8232 is a highly efficient, single-lead ECG monitor specifically designed to measure and analyze single-lead ECG waveforms. This sensor is commonly used in health monitoring systems due to its compact size, low power consumption, and ability to provide accurate and continuous heart activity data. It offers a non-invasive method for capturing real-time cardiac activity, making it ideal for wearable health devices. By monitoring electrical signals from the heart, the AD8232 enables the detection of arrhythmias, abnormal heart rhythms, and other

cardiovascular issues. Additionally, its ease of integration with microcontrollers like ESP32 ensures that it can be implemented in portable, cost-effective healthcare solutions, extending access to critical health monitoring even in remote or underserved areas. [2], [5], [8]

3.1.2. MAX30102 sensor

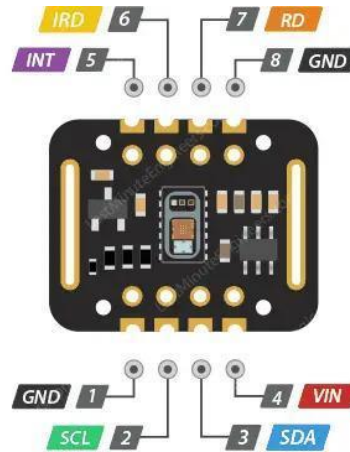


Figure 4: MAX30102 sensor

The MAX30102 is a non-invasive sensor used to monitor heart rate and blood oxygen saturation in health monitoring systems. It utilizes both a red and infrared light emitter to measure blood oxygen saturation (SpO2) and heart rate by detecting the changes in light absorption as blood pulses through the body's tissues. This sensor is ideal for wearable health devices due to its compact size, low power consumption, and ease of integration with microcontrollers like the ESP32. MAX30102 captures vital data related to heart rate and oxygen saturation, making it useful for detecting conditions like irregular heartbeats and monitoring overall cardiovascular health. [3], [5]

3.1.3. MLX90614 sensor



Figure 5: MLX90614 sensor

The MLX90614 is a precision, non-contact infrared (IR) temperature sensor designed to measure the temperature of objects and human skin without direct contact. It uses infrared technology to detect thermal radiation emitted by the target to determine the surface

temperature from a distance. This feature makes it ideal for applications such as non-invasive health diagnostics. The MLX90614 is easy to integrate with microcontrollers like the ESP32, enabling easy data acquisition and processing. Its low power consumption, compact size, and ability to interface with I2C communication make it an excellent choice for portable and battery-operated devices. [4], [5]

The above-mentioned sensors have been interfaced to an ESP32 microcontroller to collect the patient’s vitals and then send the collected data to the Edge device. [5]

3.1.4. ESP32 Microcontroller

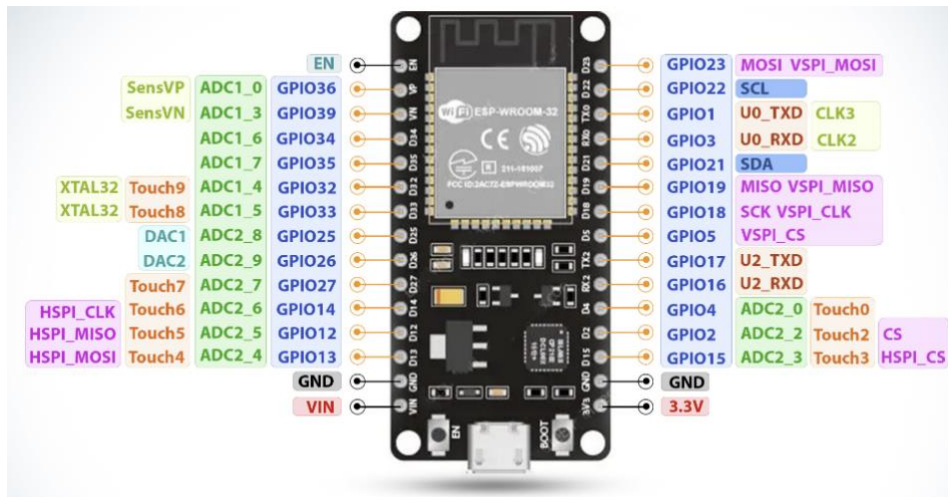


Figure 6: ESP32 Microcontroller

The ESP32 is a powerful, multipurpose microcontroller with integrated Wi-Fi and Bluetooth capabilities, making it an ideal choice for IoT applications, wearable devices, and real-time data processing systems. ESP32 offers a dual-core processor and low power consumption, which are essential for portable, battery-operated devices. The wireless connectivity of ESP32 allows seamless communication with edge devices, cloud servers, or other devices, making it an ideal choice for remote monitoring. It provides support for a wide range of peripherals, including digital and analog sensors, actuators, and displays. The ESP32 can interface with devices such as ECG sensors (like the AD8232), pulse oximeters (like the MAX30102), and infrared temperature sensors (like the MLX90614), allowing it to collect and process real-time data for applications such as cardiovascular monitoring or remote health diagnostics. [2], [3]

3.2. Stage 2: Edge Computation

Edge computing brings computational power closer to the sensing network rather than relying on the cloud server and hence improves the overall performance of the system. In this architecture, data collected from sensors is processed locally at the edge device, such as a Raspberry Pi, this removes the sole reliance on the server for data processing. This approach significantly reduces bandwidth usage by minimizing the amount of raw data sent to the cloud, as only relevant, pre-processed, or aggregated data is transmitted. By handling computation locally, response times are improved, making the system more responsive, this

is especially important for real-time applications like cardiovascular monitoring, where timely intervention is critical. [6]

The edge device receives data from the sensing network and performs several tasks, including data pre-processing, such as filtering, normalizing, and noise removal. Moreover, the edge device deploys a local inference of the trained machine learning model to perform tasks such as anomaly detection or classification. This allows for faster decision-making, as the system can generate real-time alerts without waiting for a cloud-based model to respond. Additionally, the edge device acts as an internet gateway, transmitting pre-processed data to the cloud for long-term data storage and further analysis. [9]

3.2.1. Raspberry Pi

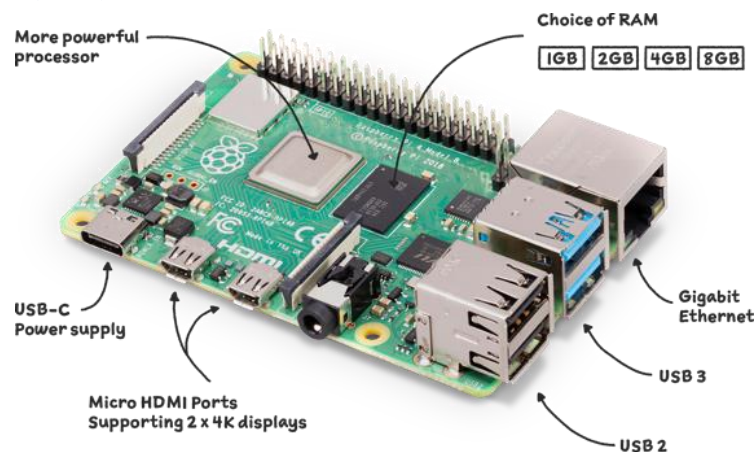


Figure 7: Raspberry Pi (Edge Device)

The Raspberry Pi is a low cost, general-purpose computing board that provides endless possibilities in areas such as IoT and product development. Technology has advanced with the advent of the Raspberry Pi as it has a compact size that is cheap and at the same time possesses a high edge computing capability, making it usable for healthcare monitoring systems. The kit is equipped with a quad-core ARM processor CPU, several USB ports, GPIO pins and Wi-Fi/Bluetooth interconnectivity. Alongside its general features as a computing device, the Raspberry Pi 4 is regarded as an edge deployment device for IoT systems. It supports edge computing by enabling the processing, storage and decision-making actions to be performed locally with low latency. This reduces the need to utilize cloud resources for help in time critical applications as the analysis of the data is done quicker. Tasks such as data preprocessing and performing AI inference are made possible through the use of the Raspberry Pi and this consequently increases the effective use of bandwidth, which affects the performance positively. Its functionality as an internet gateway improves its use in an application as the internet of things as it connects the sensors to the cloud for more data storage and the training of machine learning models. [1], [6]

3.3. Stage 3: Cloud Computing

Cloud computing, as stage 3 is referred to, plays an important role by receiving the data from the edge devices and performing more computationally extensive tasks. When the data has

been pre-processed by the edge device and is ready for long term storage, the cloud comes in to handle long-term storage of the data. After gathering data from the sensing network, that information is sent to the cloud and is made available to be accessible by the end users (doctors/patients) through a web interface. In this way, patient-related information or any other metrics can be monitored and analyzed in real time over long periods with great accuracy for discovering certain patterns and abnormalities.

Another key task performed at the cloud is the machine learning model training. A cloud server is ideal for this task as model training requires a significant amount of computational power. The system benefits scalable processing power to handle large datasets of the ECG data and training more complex models. This enables the building of better and stronger AI models which can then be sent back to the edge device for use and making decisions in real time. [2], [8]

Once the model is trained, an inference optimized version of the model is also deployed on the edge device to provide timely responses to the user.

3.4. Stage 4: GUIs (Web interface)

The system integrates a user-friendly graphical user interface (GUI) specifically designed for the end users i.e., patients and the healthcare providers, The web interface offers a comprehensive display of processed patient data sourced from the cloud. This interface allows for real-time monitoring of vital signs, providing end users with up-to-date information on a patient's health status. The GUI visualizes the vitals data from the patients and also the results of the trained ML model ensuring that the healthcare providers can respond promptly to potential emergencies. Three major roles of users have been defined in the system, Doctor, Patient and the admin.



Figure 8: Graphical User Interfaces

3.5. Communication between system components

The MLX90614 and MAX30102 sensors that are used to monitor Temperature, Heart rate and SPO2, are interfaced with an ESP32 microcontroller using an I2C communication interface. This involves connecting the data pins and voltage/ground pins of the sensors to the corresponding pins on the ESP32 microcontroller. The ESP32 microcontroller is configured to communicate with the sensors using I2C. This involves initializing the I2C peripheral on the ESP32 microcontroller, setting the appropriate I2C communication speed, and configuring the ESP32 microcontroller's I2C pins. [3], [4]

The ADS8232 sensor (ECG sensor) is interfaced with an ESP32 microcontroller using an analog communication interface. The output pin of the ADS8232 sensor is connected to an analog input pin on the ESP32 microcontroller. The ESP32 microcontroller is then configured to read the analog signal from the ADS8232 sensor.

The communication between an ESP32 microcontroller and a Raspberry Pi (Edge) is established using Wi-Fi. Both the ESP32 and Raspberry Pi have built-in Wi-Fi capabilities, hence this is a straightforward option for wireless communication. A Wi-Fi connection can be established between the two devices and data can be transferred over the established connection. At the application layer, ESP32 and Raspberry Pi use MQTT protocol for communication over the wireless network. Message Queuing Telemetry Transport (MQTT) is a lightweight, publish-subscribe protocol that is commonly used for IoT applications.

Communication between the edge device (Raspberry Pi) and the cloud server is established through an internet connection. The edge device pre-processes the data received from the sensor network and then it is transferred to the cloud server for storage and Machine Learning Model training.

On the edge device (Raspberry Pi), the patient data is anonymized before being transmitted to the cloud for storage and Machine Learning model training. Anonymization ensures that the data cannot be traced back to individual patients by removing the personal identifiable information of the patients.

In the cloud, access control mechanisms are enforced to limit who can access patient data. Role-based access controls (RBAC) are implemented to ensure that only authorized healthcare professionals can view sensitive information. Data is also being encrypted before being stored on the cloud, providing an additional layer of protection for patient privacy.

By combining encryption, anonymization, and secure access control, the system ensures that both the data in transit and data at rest are protected, safeguarding patient privacy while maintaining the integrity and confidentiality of the transmitted health data.

4. Data Collection

Data collection is a critical step in this cardiovascular patient monitoring system. The data collection process is structured to ensure comprehensive and representative ECG recordings from cardiovascular disease patients. Multiple hospitals have been visited to discuss the project and sign an MoU for the collection of patient's data.



Figure 9: Hospitals visited for data collection

An MoU was signed with AFIC (Armed Forces Institute of Cardiology, Rawalpindi, Pakistan) for the collection of patients' data.

4.1. Patient Demographics and Number of Patients

Data was collected from a total of 75 patients at the Armed Forces Institute of Cardiology (AFIC), with an equal representation of male and female participants (50% each). The age range for male participants was between 32 and 89 years, while female participants ranged from 18 to 89 years. This demographic balance was intentionally maintained to ensure the system could handle a wide variety of cases, thereby improving the reliability and generalizability of the results across different age groups and genders.

4.2. Informed Consent

Obtaining informed consent was a critical step in the process. Each patient signed a consent form, ensuring they understood the nature of the study and agreed to participate. This process upheld medical ethics by respecting patient autonomy and confidentiality. The consent form clearly explained the purpose, risks, and benefits of the study.

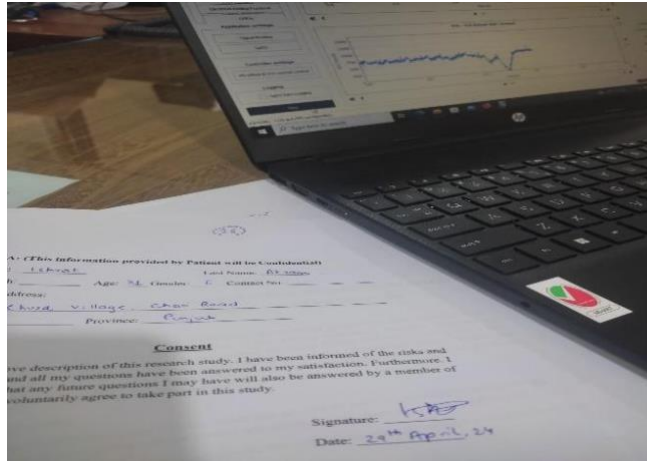



Figure 10: Consent form, Filled and signed



Capital University of Science & Technology
Islamabad
Expressway, Kahuta Road, Zone-V, Islamabad
Phone: 92-51-111-555-444 Fax: 92-51-4484705
Email: info@cusat.edu.pk Website: http://www.cusat.edu.pk

**CONSENT FORM FOR THE PATIENT REGARDING RESEARCH
PROJECT: ERASMUS PLUS DIGIHEALTH**

Introduction:
The project "Erasmus Plus DigiHealth" is an international project that aims to build capacity in Higher Education Institutes (HEIs) of the partner countries (Pakistan, Thailand and Mongolia) to develop project-based courses in digital health and that would begin a new transformation in the operation of Asian healthcare systems. Capital University of Science and Technology is a part of this international project and is working on the development of remote patient monitoring devices for cardiovascular diseases.

Instructions:

- This consent form may contain words that are new to you. If you read any words that are not clear to you, please ask the person who gave you this form to explain them to you.
- Your records will be kept confidential and will not be released without your consent except as required by law.
- Your identity will be kept private.
- If the results of this study are written in a scientific journal or presented at a scientific meeting, your name will not be used.
- Your decision to take part in this research study is entirely voluntary.
- You may refuse to take part in or you may withdraw from the study at any time without penalty or loss of benefits to which you are normally entitled.
- You may be asked to leave the study for any of the following reasons:
 - Failure to follow the Project Director's instructions.
 - The Project Director thinks it is in the best interest of your health and welfare.
 - The study is terminated.
- You may wish to discuss this with others before you agree to take part in this study.
- If you have any questions about the research now or during the study, please contact: _____

BIODATA: (This information provided by Patient will be Confidential)

First Name: _____ Last Name: _____

Date of Birth: _____ Age: _____ Gender: _____ Contact No: _____

Permanent Address: _____

Address: _____

City: _____ Province: _____

Consent

I have read the above description of this research study. I have been informed of the risks and benefits involved, and all my questions have been answered to my satisfaction. Furthermore, I have been assured that any future questions I may have will also be answered by a member of the research team. I voluntarily agree to take part in this study.

Signature: _____
Date: _____

Figure 11: Consent form for patients

The consent form is designed in such a way that it contains an introduction to the Digi-Health Project and why the data is being collected. It provides instructions related to the data collection and finally signatures are taken as consent from the patient. Consent forms have been filled in, signed and collected from all 75 patients.

4.3. Duration of ECG

For each patient, an ECG recording lasting 5 minutes was obtained. This duration was believed sufficient to capture key heart rate and rhythm patterns essential for the diagnosis and the training of machine learning models. [8]

4.4. Data Collection Equipment

Multiple pieces of equipment have been tried and tested for data collection from the patients. The first device tested to collect the patient's data was a Shimmer device. The Shimmer device is a versatile, wearable sensor platform used to collect ECG data in real-time. It is designed for non-invasive health monitoring, allowing for continuous tracking of heart activity through wireless communication. The Shimmer device collects ECG data by using electrodes placed on the body, typically on the chest, arms, or legs, to measure the electrical signals generated by the heart. These electrodes detect the small electrical changes that occur with each heartbeat, and the data is transmitted wirelessly for real-time analysis.

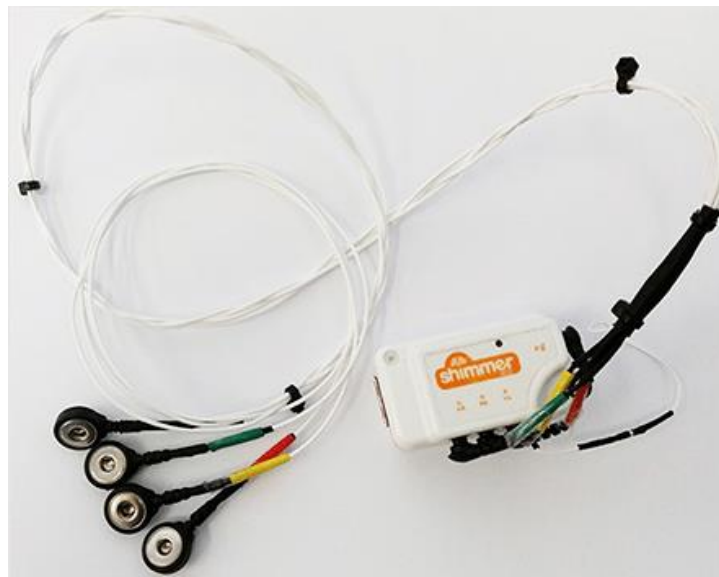


Figure 12: Shimmer Device for data collection

The feature of collecting data by placement of electrodes on the body of the patients provides us with high quality and precisely collected data but the placement of electrodes on the body of the patients makes it infeasible for continuous data collection in the provided data collection environment. In hospitals in Pakistan, especially in public healthcare settings, space is often limited, and patients are treated in overcrowded wards without designated rooms or beds for specialized procedures. This lack of space made it difficult to use traditional electrode-based ECG systems, as these require patients to lie still in a controlled environment to ensure accurate placement and continuous monitoring. The absence of proper beds and dedicated spaces for ECG data collection further complicated this process, as patients were frequently mobile or seated, making it challenging to maintain electrode placement for long-term monitoring. Given these practical constraints, alternative non-invasive solutions became necessary.

The built prototype was also used for collection of data from the patients but it also poses same problems i.e., placement of electrodes on the body of the patients. Finally, AS7038R Vital sign sensor was used to collect the data from the patients. In the congested hospital environment, as mentioned earlier, there was no adequate space to properly place electrodes for continuous data collection. Patients were often in shared spaces, without designated

beds, which made it impractical to use this method reliably. To overcome these issues, we switched to the AS7038R Vital Sign sensor, which provides non-invasive data collection through finger placement, offering a more feasible solution for the hospital environment.

Regarding the concern of using a different device for training data: The AS7038R sensor was selected after considering the feasibility in real-world hospital conditions, and its data collection method has been evaluated to ensure that it provides compatible data for training our machine learning models. Efforts were made to remove any discrepancies between the prototype and AS7038R data, ensuring continuity and accuracy in the training process.

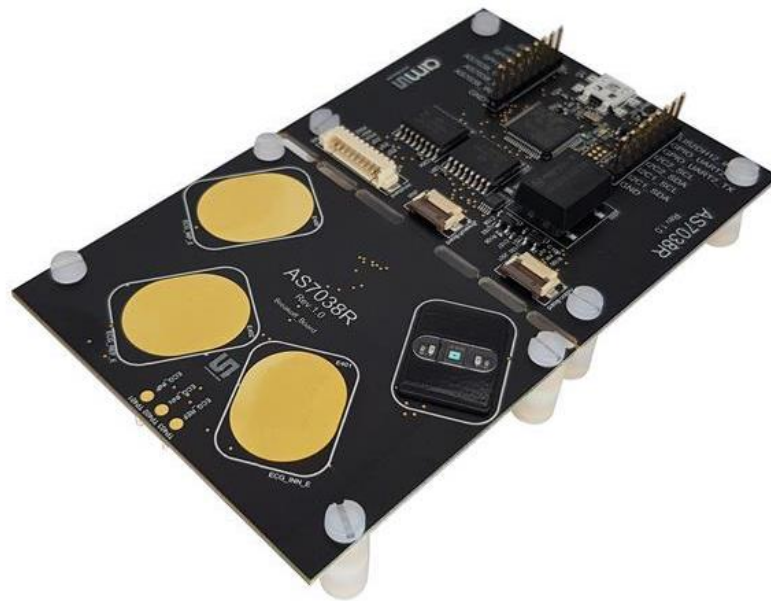


Figure 13: AS7038R vital sign sensor

The AS7038R vital sign sensor collects ECG data through finger placement on sensor pads equipped with built-in electrodes. These electrodes detect the electrical signals generated by the heart, capturing the heart's activity non-invasively. This method of data collection is particularly convenient for wearable health devices, providing real-time monitoring of vital signs such as heart rate and ECG, without the need for placing electrodes on other parts of the body.

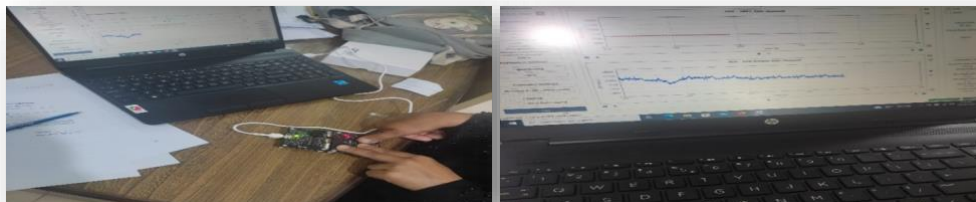


Figure 14: Data collection through AS7038R

It offers a compact and user-friendly solution for continuous cardiovascular monitoring. Once a patient places his fingers on the electrodes of the sensor, the sensor collects the data and stores it as a CSV file. The AS7038R collects data at a sampling frequency of 200 Hz.

4.5. Data Labelling

Accurate data labeling is crucial for training well performing machine learning models. This pilot case focused on classifying each heartbeat as one of two classes i.e., normal heartbeats and abnormal heartbeats. The process of data labelling involved several steps:

4.5.1. Annotation Tool

An online annotation tool was developed specifically for labeling the collected ECG data. The tool divides the ECG signal of every patient into multiple 10-second segments, making it easier for doctors to analyze.

4.5.2. Denoising

ECG signals are prone to various noise sources, such as power line interference, baseline wandering, electrode motion artifacts, and electromyographic (EMG) contamination. These noises were removed using denoising algorithms to produce a cleaner signal that is essential for accurate labeling and model training. [2], [9]

The difference between the raw and denoised signals was validated using the SimEMG dataset as ground truth.

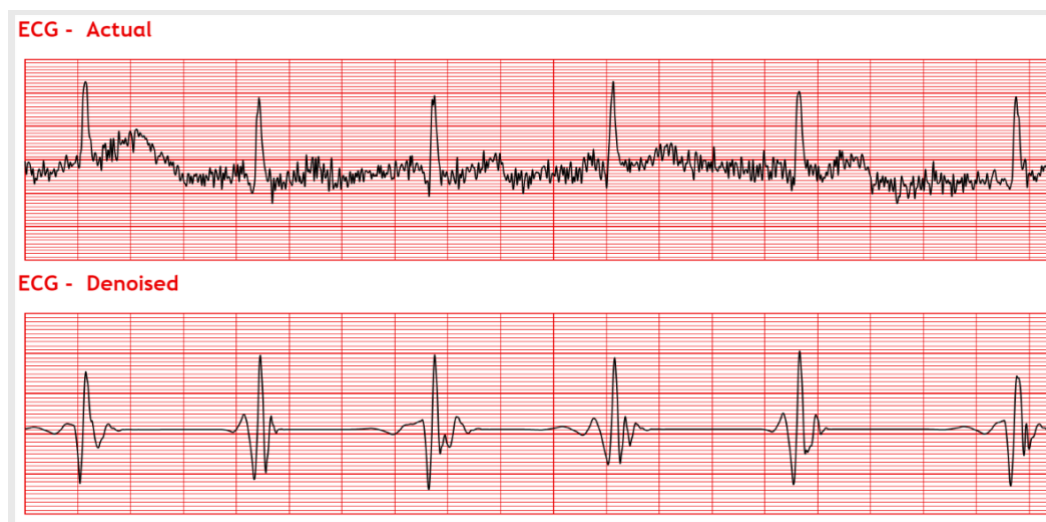


Figure 15: Actual vs De-Noised signal

Following are the types of noise removed from the ECG signals:

- Power Line Interference (PLI): Noise from electrical equipment.
- Baseline Wandering: Slow fluctuations in the ECG signal.
- Electrode Motion Artifacts: Caused by patient movement.
- EMG Contamination: Electrical activity from muscles impacting ECG signals.

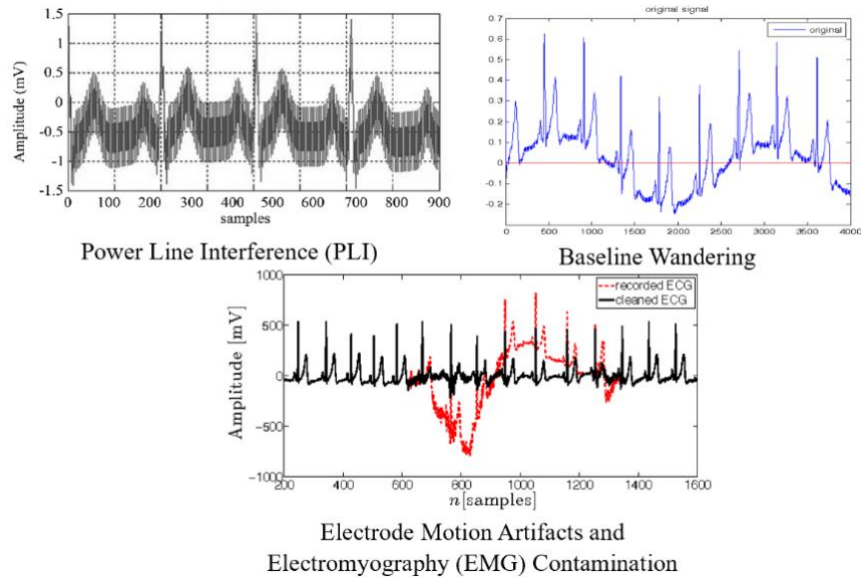


Figure 16: Noise removal from ECG

The annotation tool was designed in such a way that it contained two separate interfaces. One interface was used by the admin, who can control the values of the different filters that are being applied. The other interface was designed for the doctor that was performing the labeling. Once the collected ECG data is uploaded to a server, the doctor can view all the ECG data on the interface. The data shown contained the patient ID (p1 to p75), and patient record ID. Each patient’s ECG was split into 10 second segments and the graph of each segment was displayed to the doctor for use in annotation.



Figure 17: Admin panel of annotation tool

4.5.3. Interaction with Doctors:

Doctors played a key role in labeling the ECG data. They were tasked with reviewing the 10-second ECG fragments and assigning labels to each segment as being normal, abnormal, or unknown. If the ECG signal was unclear or distorted, the doctor could mark it as "Unknown."

This allowed for a flexible approach, ensuring that only high-quality data was used for training the machine learning models.



Figure 18: Doctor Panel

4.5.4. Classes and Annotations:

The dataset was categorized into two main classes: Normal and Abnormal. These classes were essential for creating a labeled dataset that could be used to train supervised machine learning models. The use of doctors in the labeling process ensured that each ECG segment was carefully reviewed by a medical professional before being assigned a label. The annotation process involved displaying the 10-second ECG fragments to the doctor, who then labeled them based on the observed patterns. After labeling one fragment, the doctor could move on to the next segment from the same patient or repeat the process from the beginning if necessary.

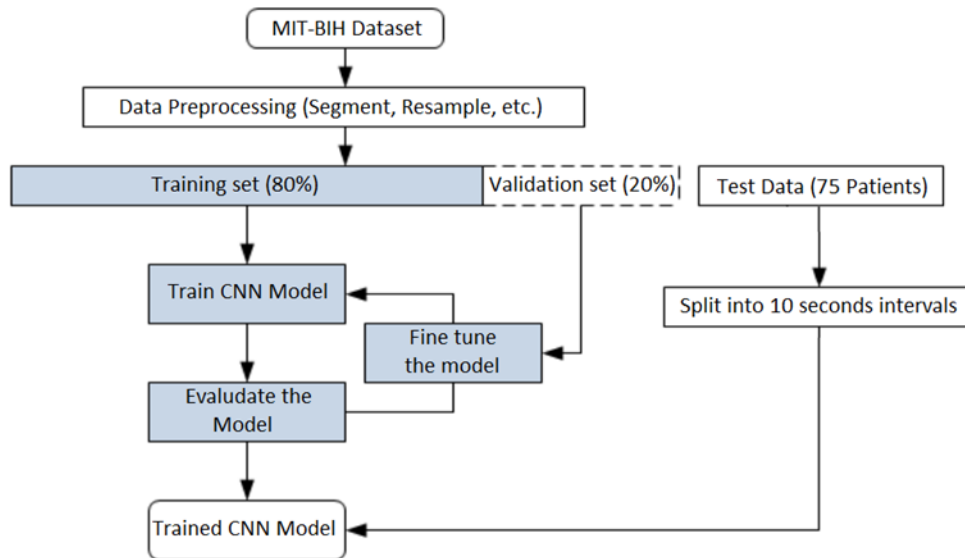
5. Model Training

In this section, we describe the training pipeline used for our ECG classification model. Our goal was to create a model capable of accurately distinguishing between normal and abnormal heartbeats from ECG signals. We employed a Convolutional Neural Network (CNN) designed to classify ECG signals into abnormal or normal categories. This CNN model was trained on the MIT-BIH Arrhythmia Database, which contains data sampled at a frequency of 360 Hz. The architecture of the CNN model includes several convolutional and pooling layers to extract features from the time-series ECG data, followed by fully connected layers to perform classification.

Given that the model was trained on data sampled at 360 Hz, it expects the input ECG signals to be at the same frequency for consistent performance. This necessitates a preprocessing step where the input signals from different sources, which may have been recorded at various sampling frequencies, are resampled to **360 Hz** to align with the model's requirements.

We used two types of data for model development: **training data** and **evaluation data**. The **training data** consists of the MIT-BIH Arrhythmia dataset, a well-established dataset for ECG signal classification. We also used **patient-specific ECG data** collected from 75 individuals, each with around 5 minutes of recorded ECG data, split into 10-second intervals. For

evaluation, the test data consisted of similar 10-second ECG intervals to measure the model's real-world performance. These decisions were made based on the availability of labeled data, the need for patient-specific training, and to ensure generalizability across abnormal cases. Following flowchart represents the training and testing pipeline of our trained model.



5.1. Extracted and Pre-processed Data:

Firstly, we loaded the MIT-BIH dataset, this dataset includes numerous records of ECG signals captured from various patients, along with corresponding annotation files that mark the presence of different types of arrhythmias and other cardiac events.

Once the dataset was loaded, we proceeded to extract the signal records and annotation files. The signal records provide the raw ECG data, while the annotation files offer crucial information on the occurrences of specific events within the signals, such as beats or anomalies.

Subsequently, we applied several pre-processing techniques to the extracted ECG signals to prepare them for model training and evaluation. This pre-processing involved several key steps:

1. **Noise Reduction:** We implemented filtering techniques to reduce the noise present in the ECG signals, enhancing the signal quality and making it easier for the model to learn relevant features.
2. **Z-Score Normalization:** The ECG signals were normalized to ensure that the amplitude values are within a consistent range. This step helps in stabilizing the learning process and improving the model's performance.
3. **Segmentation:** To make the signals compatible with the model's input requirements, we segmented the continuous ECG recordings into fixed-length windows of 10 seconds length without centering the R Peaks.. Each segment represents a specific time duration, ensuring that the input data is uniform and manageable.

4. **Resampling:** The ECG signals were resampled to adjust their frequency to match the model's requirements. For instance, if the model was trained on data sampled at 360 Hz, we resampled the signals from their original frequency to 360 Hz. This step is crucial to ensure that the input data aligns with the model's expected input format.
5. **Label Encoding:** The annotations were converted into a format suitable for model training. This involved encoding the labels into categorical formats that the model can interpret, such as binary labels indicating the presence or absence of specific arrhythmias.

By applying these pre-processing techniques, we prepared the ECG signals to be fed into the model, ensuring that they are in the optimal format for training and evaluation. This pre-processing not only enhances the quality of the data but also improves the model's ability to accurately detect and classify various cardiac conditions.

5.2. Rebalanced Classes

To address the issue of class imbalance present in the dataset, two effective methods were employed to rebalance the classes that are SKLEARN's resample method and SMOTE with NearMiss. These techniques ensured that the model received sufficient data from all categories, including the underrepresented ones, which is crucial for improving model performance.

Importantly, the rebalancing was performed only on the training set after the train-test split, ensuring that the test set remained untouched and unbalanced, reflecting real-world class distributions. This approach allows for an unbiased evaluation of the model's performance on naturally occurring imbalances, thus providing a more accurate assessment of its generalization capabilities.

5.2.1. SKLEARN Method - Resample:

The first method utilized was the resample function from the popular machine learning library, Scikit-learn (SKLEARN). This method helps in rebalancing the dataset by either oversampling the minority class or undersampling the majority class. Oversampling involves duplicating instances from the underrepresented class to increase their presence in the training data, while undersampling reduces the number of samples from the overrepresented class. The resample function in SKLEARN allows easy implementation of both strategies. This helps in ensuring that the model doesn't become biased towards the majority class, leading to more balanced predictions across all classes.

5.2.2. SMOTE with NearMiss:

The second approach used for rebalancing the classes was a combination of SMOTE (Synthetic Minority Over-sampling Technique) and NearMiss. SMOTE is an advanced technique that generates synthetic samples for the minority class rather than simply duplicating the existing ones. It creates new instances by interpolating between existing data points, ensuring that the new samples are realistic and follow the pattern of the original data. This helps in

increasing the diversity of the minority class and mitigates overfitting issues that might arise from duplicating the same data points.

On the other hand, NearMiss is a data undersampling technique applied to the majority class. It selects samples from the majority class that are closest to the minority class in terms of distance. This allows the model to focus on more challenging examples from the majority class that are harder to distinguish from the minority class, improving its ability to classify both classes effectively.

By combining SMOTE with NearMiss, we ensured a more balanced and representative dataset. The oversampling through SMOTE enriched the minority class with new, synthetic samples, while the NearMiss undersampling technique refined the majority class, helping the model learn the subtle differences between the two classes. This dual approach significantly improved the model's performance by providing a more balanced and diverse set of training data.

5.3. Train-Test Split:

To evaluate the model's performance effectively and avoid overfitting, the dataset was split into training and testing sets using an 80-20 ratio. This means that 80% of the data was used for training the model, while the remaining 20% was reserved for testing and evaluation.

The training set contains the majority of the data and is used to help the model learn patterns, features, and relationships within the ECG signals. By training the model on a large portion of the data, it becomes capable of recognizing complex features and detecting patterns such as normal and abnormal heart rhythms.

The testing set, on the other hand, is a separate portion of the data that the model has never seen before. After the model is trained, the testing set is used to assess its generalization ability, i.e., how well it performs on unseen data. This ensures that the model isn't just memorizing the training data but can accurately predict outcomes on new, real-world data.

In addition to this, the split was performed in a stratified manner to maintain the class distribution of the data. This ensures that both the training and testing sets contain a representative proportion of each class (e.g., normal and abnormal ECG patterns), which is especially important when dealing with imbalanced datasets like the MIT-BIH ECG data. This approach prevents the test set from being skewed toward a particular class, thus allowing for a fair and accurate assessment of the model's performance across all categories.

Crucially, this split was performed at the patient level, meaning that data from patients included in the training set is not present in the testing set, and vice versa. This approach ensures that the model is tested on entirely new patients, allowing for a more realistic assessment of how well it generalizes to unseen individuals. It prevents any potential overlap that could bias the evaluation and lead to overestimation of the model's performance.

5.4. Model Training using CNN

The input is passed through sequential layers of convolutions with the ReLU activation function, AvgPooling, and dropout, the convolutions are then followed by dense layers.

5.5. Model Parameters

The model was trained using the following configuration:

5.5.1. Loss Function: categorical_crossentropy

The categorical cross-entropy loss function is ideal for multi-class classification problems like ECG signal classification. It measures the performance of a classification model whose output is a probability value between 0 and 1 for each class. By minimizing the cross-entropy between the true labels and the predicted probabilities, the model learns to output probabilities that are closer to the true distribution of the data.

5.5.2. Optimizer: Adam

The Adam optimizer (Adaptive Moment Estimation) was used to optimize the model's learning process. Adam is one of the most efficient and widely used optimizers due to its adaptive learning rate and momentum. It computes individual learning rates for different parameters, making it faster and more stable for training. It combines the advantages of two other extensions of gradient descent: AdaGrad and RMSProp.

5.5.3. Batch Size: 36

The model was trained with a batch size of 36, meaning that 36 samples from the dataset were processed at a time before updating the model weights. Batch size affects the speed of training and the generalization ability of the model. A batch size of 36 strikes a balance between utilizing system resources effectively and stabilizing the gradient updates, which can help improve training efficiency.

The batch size of 36 was selected after conducting multiple trials with different batch sizes to find the most efficient balance between training time and model performance. During this experimentation phase, batch sizes such as 28, 32, and 40 were also considered, but 36 was found to provide the best compromise in terms of memory usage and training stability on the available hardware. Moreover, this specific batch size helped prevent the model from overfitting during training by maintaining a good trade-off between training speed and generalization. Therefore, the value of 36 was not chosen arbitrarily, but rather based on performance metrics observed during model training.

5.5.4. Epochs: 50

The training was carried out for 50 epochs. An epoch refers to one complete pass through the entire training dataset. Over the course of 50 epochs, the model continually updated its parameters by going through the data multiple times, allowing it to learn increasingly complex patterns in the ECG signals. After 50 epochs, the model likely reached a good level of convergence, where further improvements in performance became minimal.

While the model was trained for 50 epochs, extensive monitoring of the training and validation loss was performed throughout the training process. The decision to stop at 50 epochs was based on the observation that the model's performance had plateaued, with minimal improvements in accuracy and loss beyond this point.

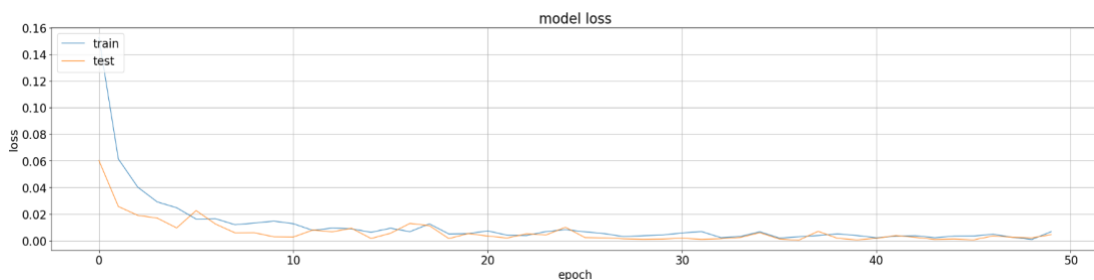
This training configuration was chosen to maximize the model's ability to learn important features from the ECG data, ensuring good classification performance while maintaining generalization on unseen data.

5.6. Loss and Accuracy curve

The **Binary Classification: Loss and Accuracy Curve** provides insights into the model's performance during training. The curve typically illustrates two key metrics over multiple epochs: **loss** and **accuracy**.

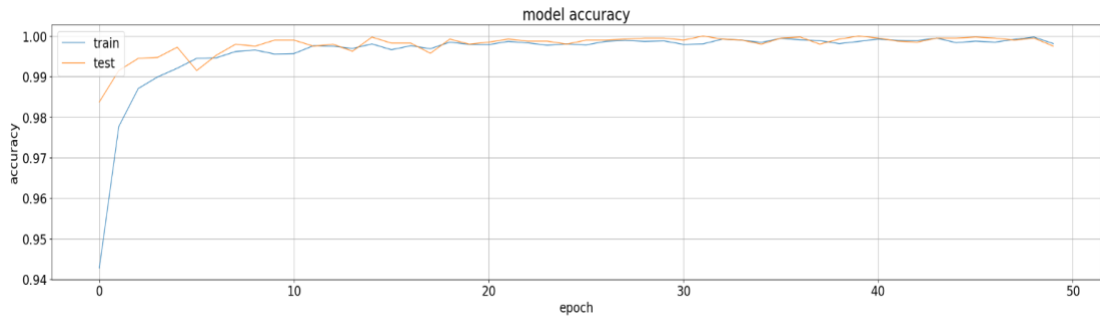
5.6.1. Loss:

The **loss curve** shows how the model's error decreases over time as it learns from the training data. In binary classification, the model minimizes a loss function (like binary cross-entropy) to improve its predictions. A steadily decreasing loss suggests that the model is learning, while a plateau or increasing loss could indicate overfitting or other issues. Following diagram show the Loss Curve on test and train datasets:



5.6.2. Accuracy:

The **accuracy curve** measures how well the model is performing in classifying the binary labels (e.g., 0 and 1). It shows the percentage of correct predictions. A rising accuracy curve on the test data indicates that the model is improving its ability to generalize from the data. Following diagram show the Accuracy Curve on test and train datasets:



This visualization helps in determining how well the model is converging and whether adjustments (e.g., in model complexity, regularization, or learning rate) are needed to improve performance further.

5.7. Classification Report and Confusion Matrix

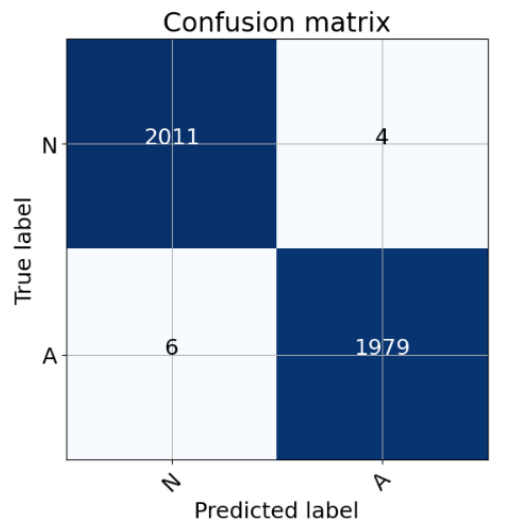
5.7.1. Classification Report:

The **classification report** is a performance evaluation metric in machine learning that provides detailed information about the classification results, specifically in binary classification tasks. Following diagram shows the classification report on test data using CNN:

Class	Precision	Recall	F1-Score	Support
Normal	0.9970	0.9980	0.9975	2015
Abnormal	0.9980	0.9970	0.9975	1985
Accuracy				4000
Average	0.9975	0.9975	0.9975	4000

5.7.2. Confusion Matrix:

The confusion matrix is a summary of the prediction results for a binary classification model. It is a table that contrasts the actual true values with the predicted values. Following diagram shows the confusion matrix using CNN:



5.8. Training SVM Model:

All the preprocessing, class balancing and train- test split is the same as for the CNN.

5.8.1. Model Parameters

For training the Support Vector Machine (SVM) model, **four different kernels** were initially employed. These kernels are crucial components in SVM as they define the decision boundary in higher-dimensional spaces and are responsible for transforming data into a format where it can be linearly separable. The kernels used for this task include:

- **Linear Kernel:** This kernel is used when the data is linearly separable, meaning that a straight line can be drawn to classify the data into different categories. It is one of the simplest kernels, often preferred when the data is simple or when interpretability is essential.
- **Radial Basis Function (RBF) Kernel:** RBF is a highly popular kernel in SVM that can handle non-linear data. It maps data points into higher dimensions, allowing the SVM to create a non-linear decision boundary. The RBF kernel is commonly used due to its ability to handle complex datasets effectively.
- **Polynomial (Poly) Kernel:** This kernel adds polynomial terms to the data, enabling it to fit non-linear models. The degree of the polynomial is a hyperparameter that can be tuned to control the complexity of the decision boundary.
- **Sigmoid Kernel:** Often associated with neural networks, this kernel behaves similarly to the activation function in neural networks and can be useful for certain types of non-linear problems.

After training these four kernels, an analysis of their performance was conducted. Based on the results, **three kernels (Linear, RBF, and Poly)** were chosen for further **hyperparameter tuning**. The hyperparameter tuning process optimizes the model's performance by adjusting parameters such as the regularization term (C), the degree of the polynomial in the Poly kernel, and the gamma value in the RBF kernel. By selecting the best-performing kernels and fine-tuning their parameters, the model can achieve higher accuracy and generalization on unseen data.

5.8.2. Hyperparameter Tuning

For hyperparameter tuning, the **Grid Search Cross-Validation (Grid Search CV)** technique was employed. This method is a systematic way of finding the optimal hyperparameters for a machine learning model by testing different combinations of predefined parameters. The updated parameter grid is as follows:

```
parameters = {'C': [1, 10],
              'gamma': [0.001, 0.01],
              'kernel': ['linear', 'rbf', 'poly']}
}
```

The degree of the polynomial kernel was not optimized in this particular grid search because we focused on optimizing the most influential hyperparameters for the model's performance, such as C, gamma, and kernel. While the degree of the polynomial kernel can indeed impact performance, it often requires a more extensive and computationally expensive search, especially when used in combination with other parameters. In our initial experimentation, a default degree value provided satisfactory results.

Meaning and purpose of each parameter:

1. C:

This is the regularization parameter in the SVM model. It controls the trade-off between achieving a low error on training data and minimizing the model's complexity.

- **C = 1:** A smaller value of C leads to a simpler decision boundary but may increase the error on the training set.
- **C = 10:** A larger C value makes the model focus more on minimizing training errors, even if that leads to a more complex decision boundary, which could risk overfitting.

2. Gamma:

This parameter is the kernel coefficient when using the RBG or polynomial kernels. It defines the influence of each training example.

- **Gamma = 0.001:** A smaller gamma value results in a more generalized model because each data point influences a larger region.
- **Gamma = 0.01:** A higher gamma value focuses more on individual data points, leading to a more complex decision boundary, which could capture intricate patterns in the data but also risks overfitting.

After testing all combinations of hyperparameters, the best combination is selected based on the chosen evaluation metric (such as accuracy, F1-score, or precision). The final model, trained with these optimal hyperparameters, is then used to predict the test set and evaluate its overall performance.

5.8.3. Accuracy on test data:

```
▼ SVC
SVC(kernel='poly', verbose=2)
```

```
y_pred = classifier.predict(X_test)
print("Accuracy:", metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9940008998650203
```

5.9. Testing of Model on Self Collected ECG Dataset

We have used our own dataset for testing of model which was collected from 75 patients, with each patient's recording lasting approximately 5 minutes. To make the data more manageable and useful for analysis, we divided these recordings into 10-second segments, resulting in a total of 1894 samples. These segmented ECG samples were then fed into a pre-trained machine learning model based on Convolutional Neural Networks (CNNs).

Only abnormal ECG beats were used for this evaluation as the primary focus of this specific analysis was on detecting abnormalities in the ECG signals. While this approach helps assess the model's sensitivity to abnormal cases, we understand that a robust ECG classification model should be able to identify both normal and abnormal heartbeats accurately. This was a preliminary evaluation, and future evaluations will involve a balanced dataset containing both normal and abnormal beats to ensure the model's overall classification performance, including its ability to correctly identify normal heart rhythms, which is essential for real-world applicability.

Data Up-sampling:

Our data was collected on 200Hz frequency and model is trained on 360Hz to we up-sampled our test data to 360Hz and then passed this data to our trained model.

The model produced an accuracy of 74 percent, indicating the effectiveness of the CNN in recognizing patterns in abnormal ECG signals.

6. Future Advancements on the Pilot Case

6.1. Thorough Investigation of Other Sensors for Improved ECG Monitoring

Future work aims to explore additional sensor technologies to enhance the quality of ECG readings. While the AD8232 sensor has been effective in monitoring heart activity, newer sensors like the AS7038R may offer more non-invasive solutions, capturing data more conveniently through finger placement. These sensors can improve both the accuracy as well as the ease of use, making the system more effective for continuous monitoring. Future research can be targeted to investigate advanced multi-lead ECG sensors that can capture richer datasets for machine learning models, improving the detection of cardiovascular abnormalities.

6.2. Movement of Cloud Services Towards End Devices

Currently, the pilot case involves a combination of cloud-based and edge computing for data processing. The edge device (Raspberry Pi) handles initial data processing tasks locally. This includes pre-processing data received from sensors, such as filtering, normalization, and cleaning to remove noise. Additionally, the edge device performs real-time analysis using pre-

trained machine learning models for anomaly detection. This localized processing allows for quick responses without relying on a continuous cloud connection.

Once the data is pre-processed at the edge, only relevant, processed information is transmitted to the cloud servers. The cloud plays a crucial role in long-term storage, advanced analytics, and machine learning model training. The cloud servers are responsible for handling computationally intensive tasks, such as retraining models with large datasets and conducting in-depth analysis of long-term trends across patient data. This allows for higher-level insights and adjustments to the models that are then pushed back to the edge devices for real-time decision-making.

An improvement to this system would involve shifting more of these tasks from the cloud and edge devices to **mobile devices** (e.g., smartphones or tablets). Mobile edge computing can take over functions like real-time data processing and local anomaly detection, significantly reducing dependency on cloud servers. For instance, smartphones can process ECG data directly, enabling faster response times and minimizing the need to send large volumes of data to the cloud for real-time decisions.

This shift would particularly enhance system performance in areas with limited internet connectivity, as it reduces the volume of data that needs to be transmitted to the cloud. Processing on mobile devices also ensures more immediate feedback, improving the user experience for both patients and healthcare providers by delivering actionable insights in real time, even in offline environments.

6.3. Long-Term ECG data Analysis

As more patient data is gathered, long-term trends in cardiovascular health can be analyzed to detect changes that could indicate serious conditions. This analysis will help identify risk factors for heart diseases before they manifest into severe health issues. Machine learning models will be adapted to track not just real-time anomalies but also slow-moving trends across months or even years. Such advancements will improve preventive healthcare strategies and allow for more proactive medical interventions.

6.4. Privacy Preservation Using Machine Learning

One of the primary concerns with continuous health monitoring systems is patient privacy. To address this, advanced machine learning techniques will be integrated to ensure that sensitive ECG data is protected. Federated learning can be explored as a method to decentralize model training, keeping data on local devices while sharing only model updates with the cloud. Encryption techniques can also be enhanced to ensure end-to-end security, protecting patient data from unauthorized access while maintaining compliance with healthcare regulations.

7. Pilot case implementation at NUST

Cardiovascular diseases (CVDs) are a leading cause of mortality in Pakistan and they pose a grave challenge to the healthcare sector. The burden is exacerbated by the limitations of the traditional healthcare infrastructure, where access to timely and adequate care is curbed by high costs, limited medical facilities, and a shortage of skilled healthcare professionals. These constraints highlight the need for innovative solutions. One such solution is the implementation of a health monitoring framework that also integrates machine learning, which can track the cardiovascular health. By enabling continuous monitoring, this system allows for early detection of potential health issues, facilitating timely medical intervention and significantly reducing the risk of severe complications.

Moreover, the high cost of traditional monitoring devices makes continuous healthcare inaccessible for many patients. A cost-effective, AI-driven alternative offers the potential to democratize healthcare, making it more affordable and accessible to a larger portion of the population. In addition to improving patient outcomes, real-time data collection and analysis also support healthcare providers in making more informed and accurate treatment decisions, ultimately enhancing the quality of care. This approach not only improves patient health outcomes but also contributes to a more efficient and responsive healthcare system.

The pilot case is focused on developing a comprehensive ECG [10] data acquisition and visualization system, integrating both hardware and software components. The pilot case at NUST's part involved the creation of an ECG data acquisition device built using open-source hardware and software, designed to capture high-fidelity ECG signals. The data acquired from the device is also accessible over a cloud-based web dashboard, which was also developed as a part of this project, allowing seamless access and visualization for doctors, patients, and hospitals. This platform not only enables real-time monitoring but also supports historical data analysis. Additionally, a Convolutional Neural Network (CNN) model was trained using the data acquired from this device to enhance predictive analysis and support diagnostic decisions, marking a significant advancement in telemedicine and personalized healthcare. We developed three versions of pilot case, the first one was intended to help us understand the challenges faced during ECG data collection, and making the integration of hardware and software seamlessly. The second version was followed by it, with some major improvements and a completely assembled version that can be used in both professional and commercial environments. The third version was then developed, aimed at reducing the scale of the device and making it as portable as possible.

This case study report explains the methodology used including hardware and software development, data acquisition, and training deep learning model. The results obtained from this case study are then discussed in report. By using the data acquired from this pilot case device, we trained Convolution Neural Network to identify different classes of arrhythmias. The report also explains the challenges observed from the development to the implementation of the case study.

8. System Architecture-NUST

This work presents the design and development of an AI-powered ECG device, integrating custom hardware with dedicated software to achieve the development of an AI powered ECG data acquisition and diagnosis device. The hardware platform employs a robust enclosure housing an embedded system composed of various sensors, a Raspberry Pi microcontroller, a touchscreen display, and a power bank for portable operation. Software development focused on constructing efficient algorithms and user interfaces tailored to optimize data acquisition, processing, and visualization on the embedded system.

In the continuously evolving field of biomedical engineering, the integration of diverse technologies can pave the way for groundbreaking advancements. This pilot case's work centers on the innovative development of an Electrocardiogram (ECG) device, leveraging the computational capabilities of the Raspberry Pi in tandem with the ESP32 microcontroller. The crux of the signal acquisition is facilitated by the Protocentral MAX86150 sensor, utilizing a chest lead configuration. Once acquired, the ECG signals are channeled to a custom-built Graphical User Interface (GUI) on the Raspberry Pi, providing real-time visualization on a 7-inch display. Recognizing the inherent challenges of signal noise and interference, an integrated tool has been developed for meticulous signal processing. This ensures not only the removal of extraneous noise but also the precise annotation of salient waveform segments. Following the acquisition and refinement stages, the processed signals are archived, constituting a comprehensive dataset. This repository then serves as a foundation for the training of advanced Artificial Intelligence (AI) models, aiming to provide enhanced diagnostic insights into cardiac anomalies. The work is a nexus of different hardware and software modules that integrate to form the ECG device. The following are the most important modules:

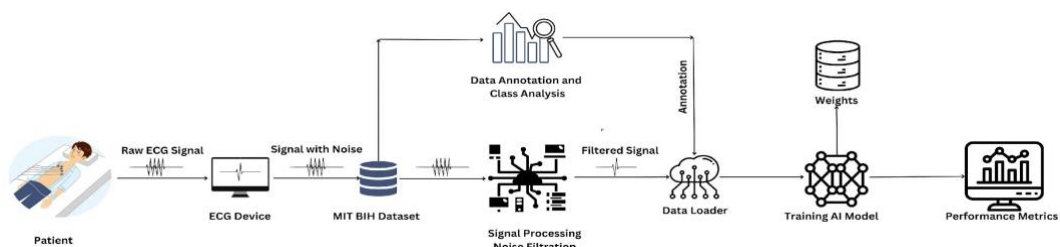


Figure 19: Complete Methodology Diagram

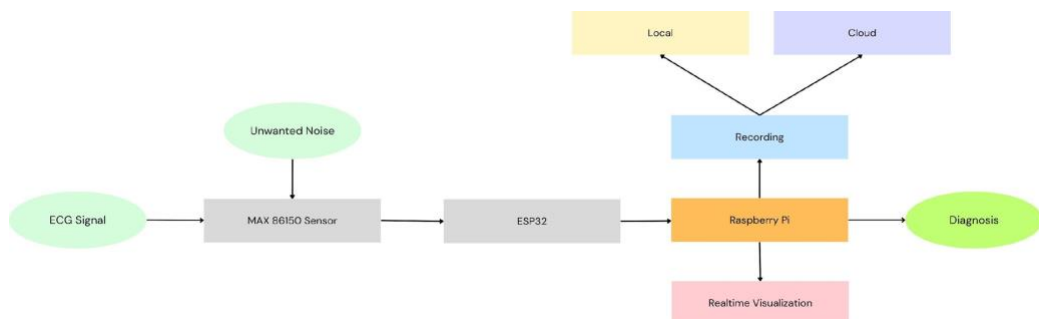


Figure 20: Flow Diagram of the Methodology Followed

8.1. Hardware Enclosure

Keeping the usage of the device in view, a hardware enclosure that offers the user robustness and portability had to be taken into account. For this, starting from a prototype version 1 (Figure 26) was initially developed. Based upon a shell enclosure, the design allowed us to modify or increase the power capacity by swapping the installed power bank. Secondly, it was robust enough to survive harsh environments such as water spills and extreme temperatures. This prototype was used to collect the data of the patients from the hospital. The design visualizes the real time ECG signals along with the option of recording and saving them. Although this version offered robustness and a modular approach, this came with a trade-off in terms of portability and ease of use. Another potential challenge in this design was the exposure of the ECG sensor to the environment without any enclosure. This introduced noise in the signals. To combat this issue, a different design was opted for. This new and improved enclosure became the final version since it offered I/O ports, a separate compartment for every electronic component, and most importantly, a shielded portion for the placement of the sensor. With these required upgrades, this design is portable and easy to hold.



Figure 21: Prototype Version 1

8.2. Embedded System Integration

The working of the device depends upon the embedded systems installed inside it. The embedded systems including the micro controller, sensor, and computing device are:



Figure 22: Final design of ECG Device (Version 2)



Figure 23 Version 3 of ECG Device

Raspberry Pi 4

The Raspberry Pi 4 Model B is a credit-card sized computer developed by the Raspberry Pi Foundation with the intention of promoting the teaching of basic computer science in schools. It is a powerful and versatile minicomputer that can be used for a wide variety of purposes, including research. The Raspberry Pi 4 features a 1.5GHz quad-core 64-bit ARM Cortex-A72 CPU, up to 8GB of LPDDR4-3200 SDRAM, dual micro-HDMI ports that support up to 4Kp60 resolution, two USB 3.0 ports and two USB 2.0 ports, Gigabit Ethernet, 802.11ac Wi-Fi, and Bluetooth 5.0. This makes it significantly more powerful than previous Raspberry Pi models, and it is now capable of running demanding applications such as video editing, 3D printing, and even light gaming.



Figure 24: Real Time ECG Visualization GUI

One of the benefits of using the Raspberry Pi 4 for research is its affordability. The base model with 1GB of RAM costs only \$35, making it a much more cost-effective option than traditional desktop computers or laptops. This makes it ideal for researchers who are working on a budget or who need to deploy a large number of devices. Another advantage of the Raspberry Pi 4 is its small size and low power consumption. This makes it ideal for portable applications or for situations where space is limited. For example, it can be used to collect data in the field or to deploy in remote locations. The Raspberry Pi 4 is also very versatile. It can run a variety of operating systems, including Raspbian (a Debian-based Linux distribution), Ubuntu, and Windows 10 IoT Core. This makes it suitable for a wide range of research applications. In addition, the Raspberry Pi has a large and active community of users and developers. This means that there is a wealth of resources available to help you get started with your research project. There are also many tutorials and forums where you can find help and advice. Its small size, low power consumption, and wide range of capabilities make it ideal for edge computing in portable devices where edge computing is required. In this work, the heart of the processing lies with the Raspberry Pi. The ECG data is read from the sensor using an embedded micro-controller, which is then transmitted to Raspberry Pi using wired serial communication. The onboard Raspberry Pi performs the following tasks: 1) Real-time Visualization 2) Data Recording 3) Data Annotation. The first two tasks are to be performed at a high sampling rate, in this case, 200 Hz. These tasks are programmed using Python in Raspberry Pi's native operating system Raspbian. The following libraries are used for ECG

signal visualization and recording: Pandas, 2) PyQt5 3) numpy 4) scikit-learn

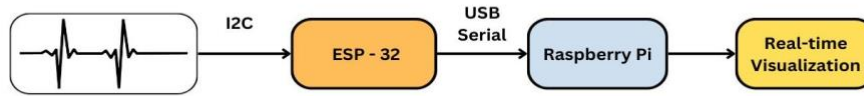


Figure 25: Flow Diagram of Sensor Integration with ESP 32

Protocentral Max 86150 ECG Sensor

The Protocentral MAX 86150 ECG Sensor (Figure 30) allows measuring ECG signals with reliability. As a flagship product from Protocentral, the MAX 86150 can seamlessly be integrated into various medical devices and wearable technologies. This integration facilitates essential data collection, crucial for diagnosing and monitoring a wide range of heart conditions, thereby enhancing patient care and enabling proactive health management.

From a technical standpoint, the MAX 86150 ECG sensor operates with a high sampling rate of 2000 Hz per channel, ensuring the capture of high-resolution data necessary for accurate cardiac assessments. The built-in analog front-end (AFE) provides low-noise amplification and precise signal conditioning, further enhancing the quality of the recorded signals. Designed with energy efficiency in mind, the sensor offers optimized power consumption, making it ideal for use in portable and wearable devices that require extended operational periods without frequent recharging. Its compact and lightweight design, measuring 50mm by 30mm by 15mm and weighing just 50 grams, ensures comfortable wearability and easy integration into various form factors, whether incorporated into sophisticated medical equipment or everyday wearable health devices.

Durability and compliance are also key features of the Protocentral MAX 86150 ECG Sensor. Constructed with robust materials, the sensor is built to withstand daily wear and environmental factors, ensuring long-term reliability even in demanding conditions. It meets stringent international medical device standards, including ISO 13485 and FDA regulations, which guarantees its safety and efficacy for clinical use. Furthermore, the sensor's compatibility with major medical software platforms and customizable APIs allows for tailored application development, providing flexibility to healthcare professionals and developers alike.



Figure 26: Protocentral Max 86150 ECG Sensor

In this work, we utilized a 3-lead configuration corresponding to Lead II. This setup involves electrodes placed on the **Right Arm (RA)**, **Left Arm (LA)**, and **Right Leg (RL)**. These leads form Einthoven's triangle for the Lead II configuration, enabling accurate measurement of voltage levels and heartbeats. This arrangement allows for the visualization of basic cardiac cycles and the detection of arrhythmias.

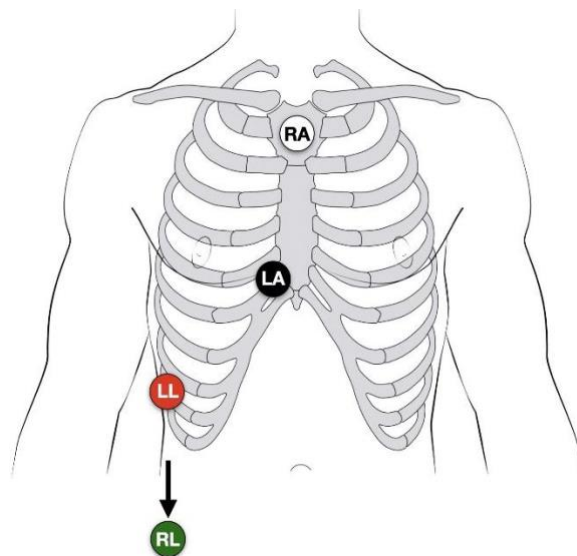


Figure 27: Lead II Configuration for RA, LA, RL

7-inch HDMI Touchscreen:

The 7" Capacitive Touch Screen LCD described herein serves to enhance the user interface (UI) experience. This touchscreen display operates seamlessly with a built-in HDMI interface, allowing it to function as a computer monitor akin to any standard HDMI screen.

This LCD display is equipped with a preloaded Raspbian driver and Ubuntu image, facilitating ease of integration into various environments. The key features of the Raspberry Pi HDMI LCD 7 Inch Touchscreen Display Module encompass an 800×480 high resolution, and capacitive

touch control. The display module includes essential accessories such as an HDMI cable and a USB type A plug to micro B plug cable.



Figure 28: 7 inch touchscreen

8.3. GUI Software Development

The work of making a smart ECG device involves various parts and an intricate integration of hardware as well as software components. For the device to be completely robust and functional, the hardware part needs to be well-defined, robust against external noise, and be operating at stable voltages. To make the device portable it should also be ensured that the device is as compact as possible. For the software part, the GUI (Figure 34) needs to visualize the signals in realtime so that they can easily be interpreted. Apart from just visualization, the signal needs to be recorded locally as well to be used for the annotation process and the subsequent training of the AI algorithm. The annotation tool is another core component of the software. This tool would allow us to load, visualize, and annotate the ECG segments, and then save them in a machine-readable format. This tool has to be user-friendly and yet powerful to allow multiple different operations to be performed over the signals.



Figure 29: ECG Visualization and Annotation Tool

Many open-source signal visualization tools lack temporal mapping capabilities. Since the accurate interpretation of ECG signals fundamentally depends on their visualization over time, it is essential to have tools that facilitate this aspect of analysis. The need for electrocardiogram (ECG) signal visualization in a standardized graphical format arises from the necessity for medical professionals to precisely diagnose cardiac conditions. Effective time-based visualization ensures that healthcare providers can reliably assess heart activity, leading to more accurate and timely diagnoses.

The graph's depiction of the heart's electrical activity is instrumental in identifying life-threatening conditions such as myocardial infarction, arrhythmias, and other cardiac abnormalities. Without good visual representation, these conditions might remain undetected and untreated, potentially leading to severe health consequences. The significance of a correct ECG visualization cannot be overstated. It enables the detection of heart rate, rhythm, and conduction abnormalities; the identification of electrolyte imbalances; and the observation of the effects of cardiac drugs. It also aids in the prognosis of patients with cardiovascular diseases and in pre-operative and post-operative monitoring. However, if the ECG is not visualized following the established graph format, scaling errors could arise, in which the annotator misinterprets the voltages and/or time scale. This tool makes it possible to annotate the data in the required format for training AI model. The input to this tool is the raw ECG signal recorded by the device in the csv format which is then visualized in the standard format. Once visualized, the tool is designed to allow the following operations: **ECG Traversal, Heartbeat Annotation, Wave Segment Annotation, R-Peak Annotation, and Signal Filtration**

For performing annotation of R-peaks, the user can select the option of R-peak from the checklist and then by clicking on the R-peaks, the annotations are visualized and saved. With these different options, the user has the liberty to perform three types of annotations for the ECG signal: Heartbeat, Wave Segment, and R-peak. For each annotation type, a separate column is saved in the csv file along with the ECG signal. Under the heartbeat annotation, the complete heartbeat is selected from P to T point and then annotated. Under the wave segment, the particular segment of the wave is selected, e.g. P, Q, R, S, or T segment and

annotated for the respective label. R-peak annotation allows for labelling the R-peaks.

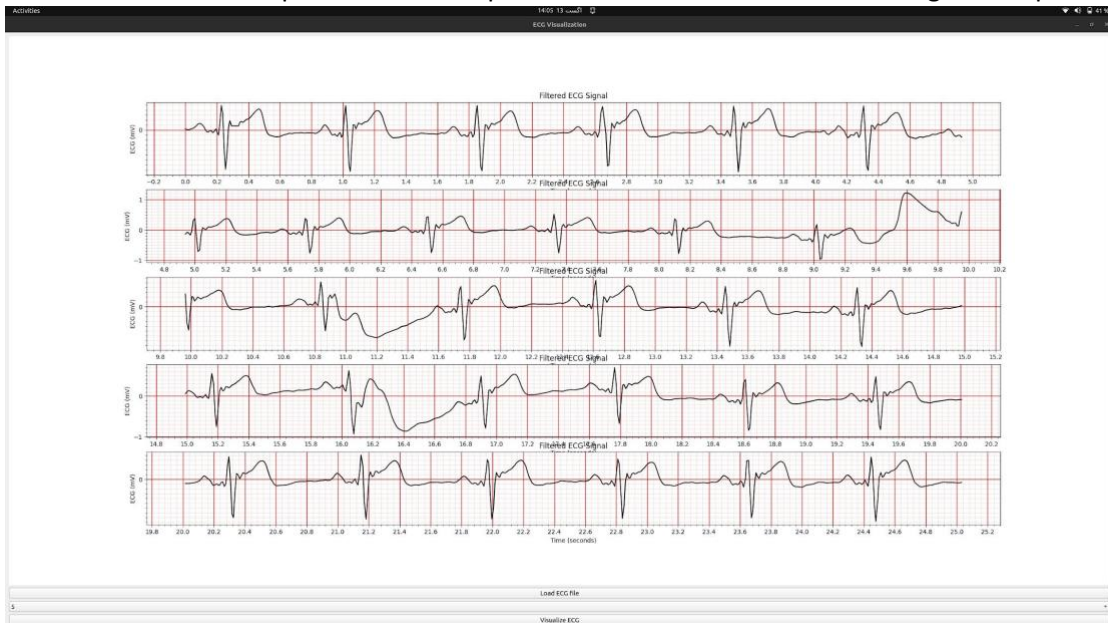


Figure 30: ECG Visualization Tool

Web Dashboard Portal

The web dashboard portal of the pilot case study is an important component, designed for the visualization and management of ECG data and patient information. It serves as both an on-device and off-device platform, offering a user-friendly interface for doctors, medical staff, and patients. Below is an in-depth exploration of the web portal, emphasizing its functionality, user experience, and technical aspects.

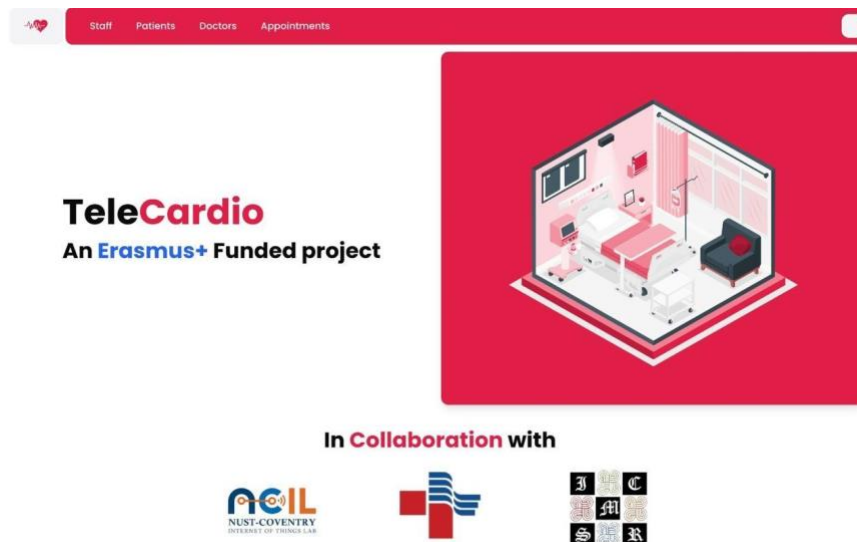


Figure 31: Web Dashboard Portal Homepage

The visualization system in the pilot case study includes both on-device and off-device components. On-device visualization allows the system to process and display data directly

on medical hardware, such as a tablet or embedded system attached to the ECG device, providing immediate feedback on ECG readings by displaying waveforms and vital statistics in real time. Off-device visualization is achieved through a web dashboard accessible remotely on computers, laptops, or mobile devices, enabling healthcare providers to monitor patients' ECG data, view diagnostic trends, and manage patient profiles from anywhere with internet access.

Web Dashboard Interface: The dashboard portal is designed to be intuitive and comprehensive, as depicted in the provided images. The dashboard registers the profile for the end user or the doctor. After registration, this web portal offers different features that assist them in detecting arrhythmias and performing diagnosis.

User-Friendly Design: The dashboard has a clean and professional look, with a clear layout that shows essential data points such as the number of patients, ECG scans, and the latest activities. It is divided into sections for easier navigation, including patient profiles, ECG history, and medical reports.

Activity Log: The activity panel on the dashboard displays recent activities such as the completion of ECG scans for each patient, providing a time-stamped summary of medical events.

ECG Data Visualization: One of the most critical aspects of the portal is ECG data visualization. The dashboard allows healthcare professionals to observe ECG waveforms, with annotations highlighting irregularities like tachycardia. This feature helps in quickly identifying issues such as arrhythmias, enabling timely medical interventions.

Patient Management: The portal provides robust functionality for managing patient data. Each patient has a unique profile, which includes:

Demographic Information: Such as age, gender, and appointment details.

Patient status: The patient's medical status, including whether their condition is normal or requires further examination.

Appointment Scheduling: Healthcare providers can schedule and view upcoming appointments within the dashboard. The integration of this data ensures that doctors and healthcare staff have immediate access to a patient's medical history and status.

Customization and Analytics: The dashboard allows for customization based on the needs of the healthcare provider. Medical staff can filter ECG data by date, patient, or specific conditions like tachycardia. Moreover, the dashboard can provide analytics on the collected data, such as trends in patient heart rates over time, offering a more comprehensive understanding of a patient's health.

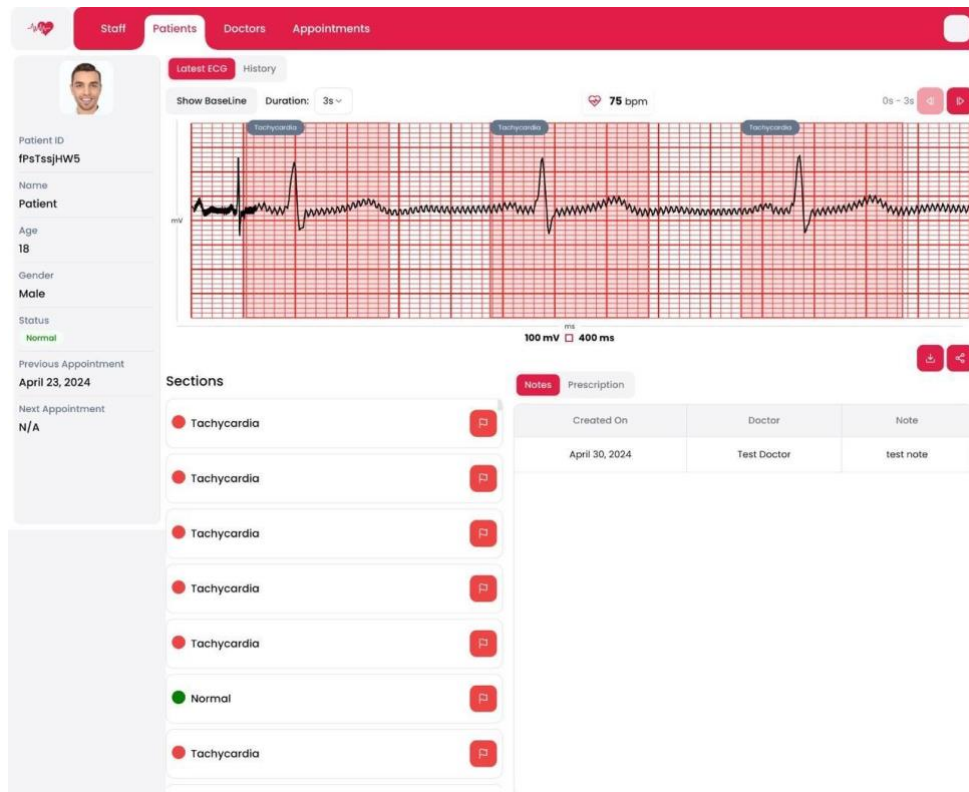


Figure 32: Web Dashboard Portal

8.4 ECG Data Processing

The process of the ECG Data Processing is a challenging step since the edifice of deep learning network's training relies on this step. The data processing involves reading the data in the right format, looking for irrelevant signals and omitting them, filtering noise from the signal, detecting heartbeats, segmenting the data for use in neural networks.. Firstly, the CSV file of the ECG signal is read using the pandas library. Each CSV file as the following columns: **time_ms, ecg_signal, heartbeat, r_peak**

The first column "time_ms" holds the values of time in milliseconds, used for plotting the samples and also calculating the sampling rate of the data. The second column stores the most important data, the ECG signal values in mV. After this column, there is a "heartbeat" column in which the time series segmentation of the labeled heartbeats is stored. By time-series segmentation, the part of the heartbeat starting from P to T segment of each heartbeat is annotated. Finally, the R_peak column is used to store the particular value of a signal that represents the R peak of the signal. This is stored as a Boolean value (1), in data processing, the other Nan or empty cells are populated with 0. The data using this device was acquired in collaboration with the Polyclinic Federal Government Hospital using different patients admitted in the cardiac ward. The data of around 10 patients for 20 minutes each was acquired for 20 minutes and saved in the described CSV format. In order to make the data ready for the training of AI model, we first need to visualize it, as it allows us to look for artifacts and noise in the data.

Visualization and Noise Removal

To visualize the newly acquired data, we used Matplotlib in Python. Upon looking at the visualized ECG signals (Figure 35), it was obvious that the raw signals contained a significant amount of noise, which was due to the 50Hz electrical signal coming from the main powerlines.

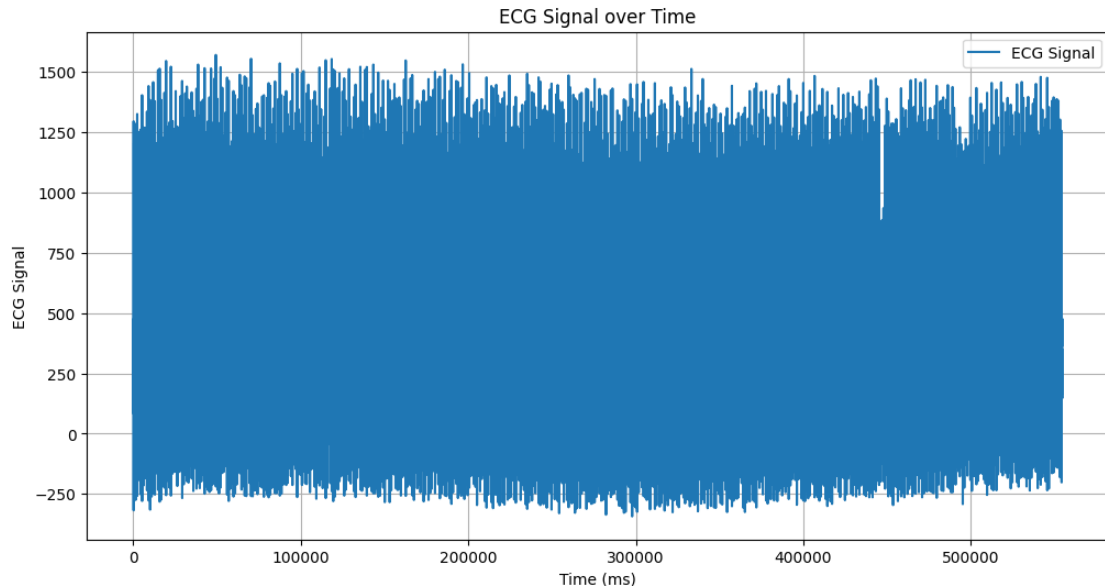


Figure 33 ECG Signal Visualization

To remove this noise, different noise filtration algorithms can be used. In our case, we used a "Notch Filter", which is a band stop filter that filters out a specific frequency from the signals. The used filter the parameter values of 50 Hz for the notch frequency, 200 Hz for the original signal's sampling rate, and 30 as the quality factor. This filtered signal is then passed on to the next step of detecting where the heartbeats are located in the signal through R-peak detection.. Once the R peaks from each heartbeat are detected, we map those peaks or label them concerning the segment of already annotated heartbeats. This step concludes the data preprocessing. In this step the detected R peak is relabeled with respect to the particular heartbeat it falls into, and then stored under a different column.

Superimposed on the ECG signal is background noise, attributed to various electrical influences within and around the body. Power line interference at 50/60 Hz, muscle movement artifacts, and intrinsic biological noise contribute to this interference, posing challenges to signal accuracy. This visual analysis provides an overview of the raw ECG signal's components and challenges. Figure 35 visually represents this amalgamation, laying the groundwork for a detailed examination of signal processing techniques aimed at improving the precision of ECG interpretation.

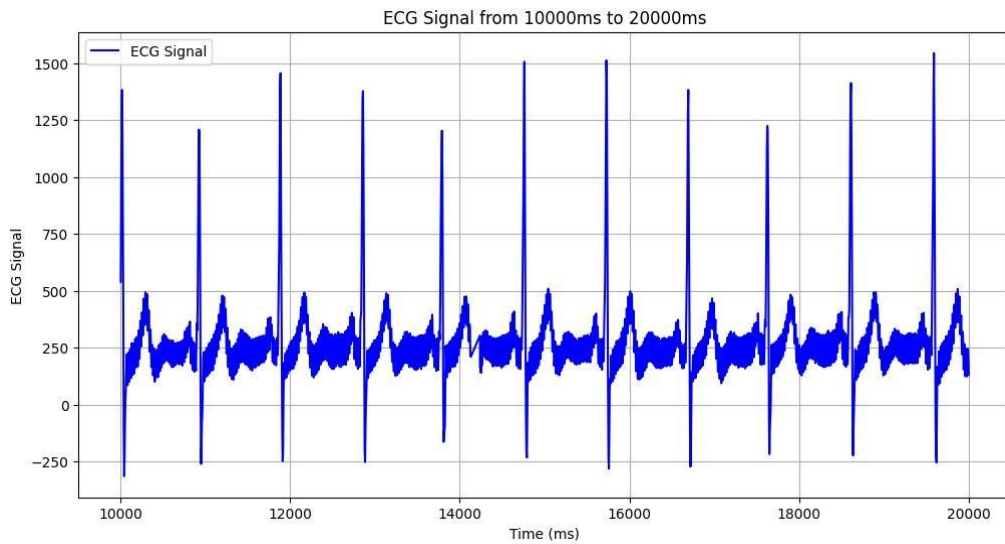


Figure 34 : ECG Signals Visualization RAW Signals Unfiltered

The presence of noise (Figure 35) necessitates careful consideration. While it may obscure certain diagnostic features, it also offers a glimpse into the complex interplay between the heart and its environment. The contrasting difference between unfiltered and filtered signals is depicted in Figure 36. The noisy signal is represented with blue color while the filtered ECG signal after applying notch filter is plotted in red color. By understanding the nature and sources of noise, we can develop filtering techniques and signal processing algorithms to extract the true electrocardiographic signal with greater clarity. By keenly examining the unfiltered ECG signal visualization, we gain invaluable insights into the heart's intrinsic electrical activity. This initial exploration lays the foundation for further analysis, paving the way for filtering, feature extraction, and ultimately, the classification of ECG patterns for diagnostic purposes.

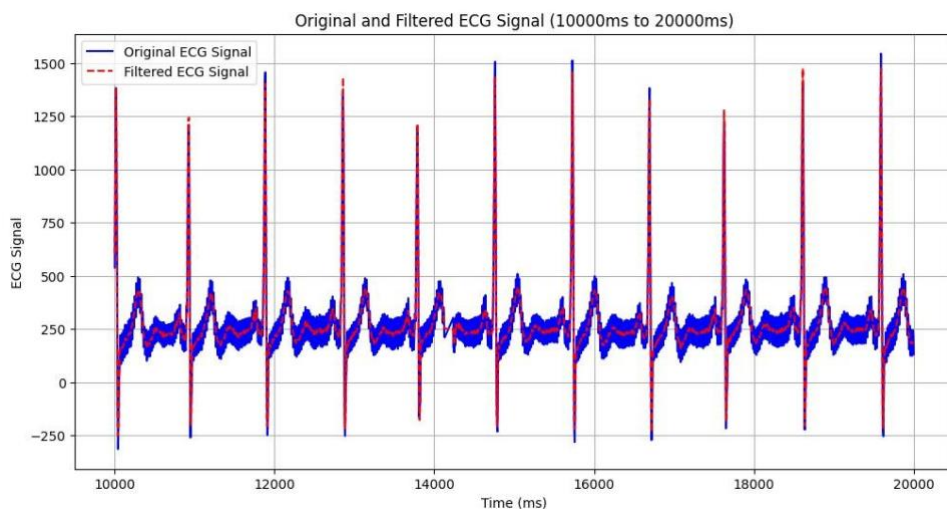


Figure 35: ECG Signals Visualization RAW and Filtered ECG Signals

The figure 36 shows a scaled portion of an ECG signal with several labeled peaks, including the R peaks. This magnified view of the ECG signal reveals a closer look at the individual components that make up each heartbeat. Prominently featured are the P waves, small deflections preceding the QRS complex and representing atrial depolarization. The QRS complex, the most prominent feature of the ECG, marks the depolarization of the ventricles, the heart's main pumping chambers. It consists of a sharp upward deflection (Q wave), a peak (R wave), and a downward deflection (S wave). Finally, the T wave, a rounded deflection following the QRS complex, reflects ventricular repolarization. These peaks and valleys within the ECG signal provide valuable information about the heart's electrical activity. By analyzing their shapes, amplitudes, and intervals, healthcare professionals can gain insights into a variety of cardiac conditions, including arrhythmias, conduction abnormalities, and myocardial ischemia. This magnified view of the ECG signal not only unveils the individual components of each heartbeat but also plays a crucial role in crafting the input features for our analysis. The detected peaks, particularly those of the QRS complex, serve as essential landmarks for segmenting the continuous ECG signal into windows, each containing a single heartbeat. To achieve this, we strategically center each window around a detected QRS peak of interval 200ms, ensuring that the window encompasses the most informative segment of the heartbeat. This approach allows us to isolate and focus on the unique characteristics of each heartbeat, providing a comprehensive representation of the heart's electrical activity. By carefully framing the ECG signal in this manner, we can create a structured dataset of heartbeat windows, ready to be fed into machine learning algorithms for automated classification and analysis. These windows are the input key features to the CNN. Each window is a feature vector of 200 samples. The input vector now becomes the heartbeat. In this work, the main concept of the input features to the deep learning network revolves around the selection of a window of the ECG signal. To perform this, we first detect R peaks from the ECG signal using the peak detection algorithm included in scipy library to detect R peaks in each ecg heartbeat. Since the frequency of the ECG signals is 200 hertz, which means that for 200 milliseconds, there should be one heartbeat. We integrate this approach in the formation of the windows for ECG signals. First, we detect the R peak, and then for each detected R peak, we form a window around it in such a way that the detected R peak is in the center of the window. Since the heartbeats are already labeled using the annotation tools, they act as the labels for the class of the input

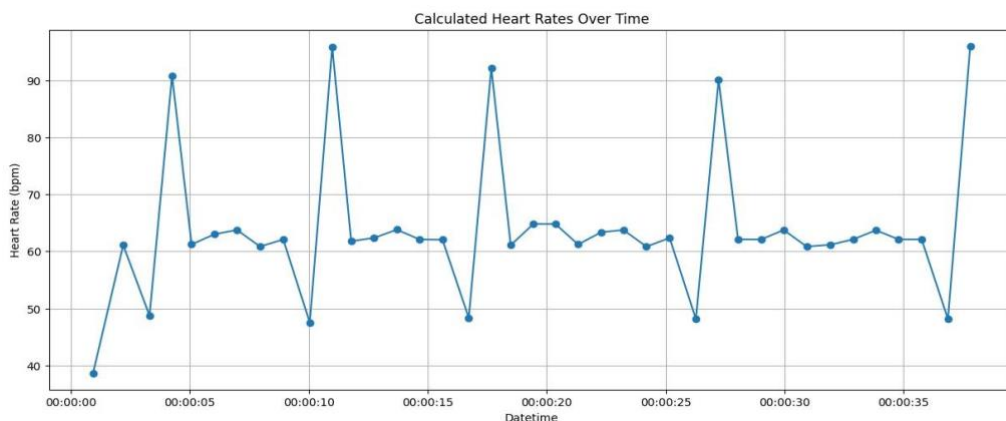


Figure 36 ECG Signal Visualization

Figure 36 provides a detailed representation of the human cardiac cycle, as observed through an Electrocardiogram (ECG) graph derived from our proprietary dataset. The application of the SciPy peak detector has enabled us to discern key events, notably the QRS complex, offering insights into the orchestrated rhythm of the heartbeat.

These identified peaks extend beyond mere graphical representations; they serve as indicators of cardiac health. The amplitude and intervals of these peaks, akin to strokes on a canvas, convey information regarding heart rate, rhythm, and potential irregularities. An in-depth exploration of these intricacies holds the promise of unlocking valuable information, ranging from recognizing healthy variations to detecting potential abnormalities that might otherwise elude observation. The significance of this peak detection process within the confines of our indigenous dataset is noteworthy. Beyond its technical achievements, it underscores the adaptability and inclusivity of our chosen algorithm. This achievement paves the way for the development of culturally sensitive ECG analysis tools tailored to diverse populations, ensuring comprehensive cardiac assessments irrespective of individual variations.

However, it is essential to acknowledge the challenges inherent in this process. The ECG landscape is inherently dynamic, characterized by individual variations, real-world noise, and the ever-evolving nature of the cardiac system.

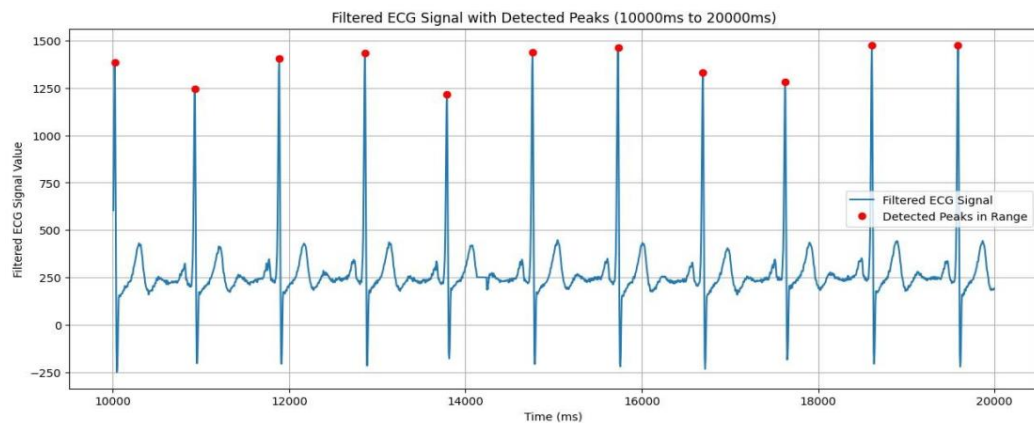


Figure 37: ECG Signals detected peaks

Peak Detection

Once the filtration process is complete, the next step is to extract the feature vector for training the deep learning network. In order to do so, an effective approach that involves the use of peaks in the signal to form a feature vector is used.

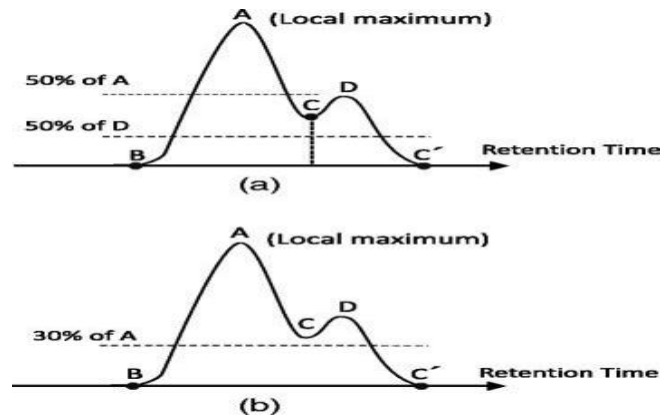


Figure 38 : Peak detection using local maxima

Peak detection with tailored parameters has been undertaken on the entirety of the filtered ECG dataset. This involves:

- a. **Height threshold adjustment:** The threshold for peak identification has been set to a value exceeding the mean of the filtered signal by one standard deviation. This aims to capture peaks of significant prominence while mitigating false positives.
- b. **Minimum peak distance enforcement:** A minimum distance constraint has been imposed, equivalent to 40% of the sampling rate. This circumvents the detection of adjacent peaks that likely represent the same underlying physiological event.

Subsequently, a temporal range of 10,000 to 20,000 milliseconds has been selected for focused analysis. This entails:

- a. **Data segmentation:** The ECG data has been compartmentalized to encompass exclusively samples within the specified temporal range, fostering a concentrated examination of this segment.
- b. **Peak filtering:** The complete set of detected peaks has undergone a filtering process to isolate those residing within the designated range. This yields a refined collection of peaks pertinent to the segment under investigation.

To find_peaks function from scipy is used with the following parameters:

x: The input 1-D array (signal) in which peaks need to be identified.

height: Specifies the required height of peaks. It can be a number, None, an array matching x, or a 2-element sequence. The first element is the minimal required height, and the second (if supplied) is the maximal required height.

threshold: Specifies the required threshold of peaks, which is the vertical distance to neighboring samples. Similar to height, it can be a number, None, an array matching x, or a 2-element sequence.

distance: Specifies the required minimal horizontal distance in samples between neighboring peaks.

prominence: Specifies the required prominence of peaks. Similar to height and threshold, it can be a number, None, an array matching x, or a 2-element sequence.

width: Specifies the required width of peaks in samples. Similar to other parameters, it can be a number, None, an array matching x, or a 2-element sequence.

wlen: Used for the calculation of peak prominences. It is only used if prominence or width is given.

rel_height: Used for the calculation of peak width. It is only used if width is given.

plateau_size: Specifies the required size of the flat top of peaks in samples. It can be a number, None, an array matching x, or a 2-element sequence.

The resulting filtered peaks 42, alongside their corresponding indices within the range-specific data, constitute valuable assets for subsequent analyses and visualizations.

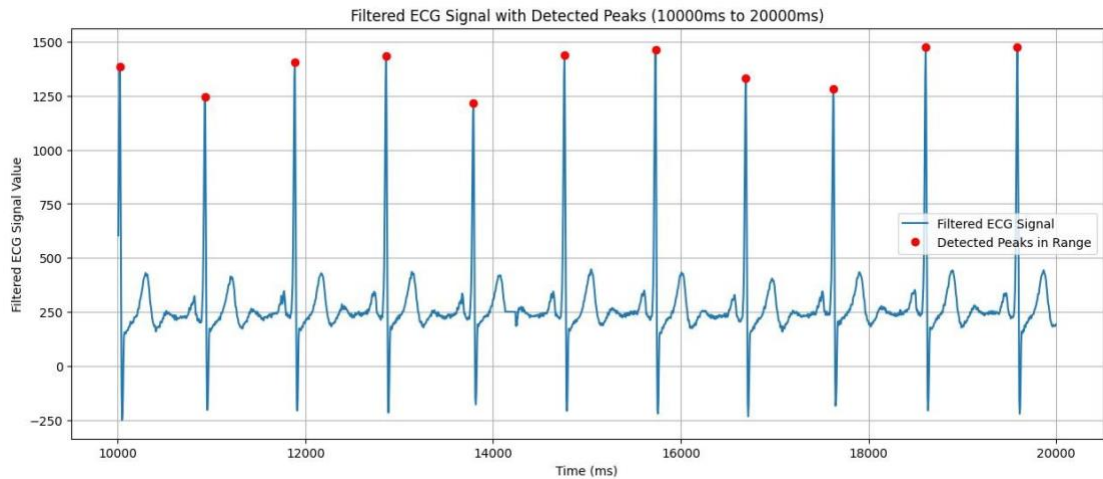


Figure 39: Detected Peaks of ECG Signal for 10k to 20k ms interval

Feature Window Formation

After detecting the peaks, R-peaks in this case, the ECG signal is confined for an interval of 200 samples. This makes the duration of the signal for 1 second. In this duration, we get all the segments from P to T. As the ECG signals are already annotated in the form of segments for their respective heartbeats, it becomes easier to assign a label for each detected peak. A peak is always a part of the heartbeat, which means that it is a part of the segmented section. The whole concept is to detect the peak and form a window in such a way that the peak lies in the middle of the heartbeat. For a 200-sample window, a peak would usually be at the 100th sample; before the peak, there are P and Q segments, while S and T appear after the R peak. This makes it easier to form a feature vector of 200 samples from the value of the signal based on this window generated from the detected peak.

So, for each signal, there is a window of a fixed interval around the R peak in such a way the peak is at the middle of the window. This makes it possible to include the important segments of the ECG signal in the input feature vector. This solves once problem of the input feature, while the enigma of forming output labels remains unsolved. This is achieved by extracting the labeled class in which the segment of the peak lies.

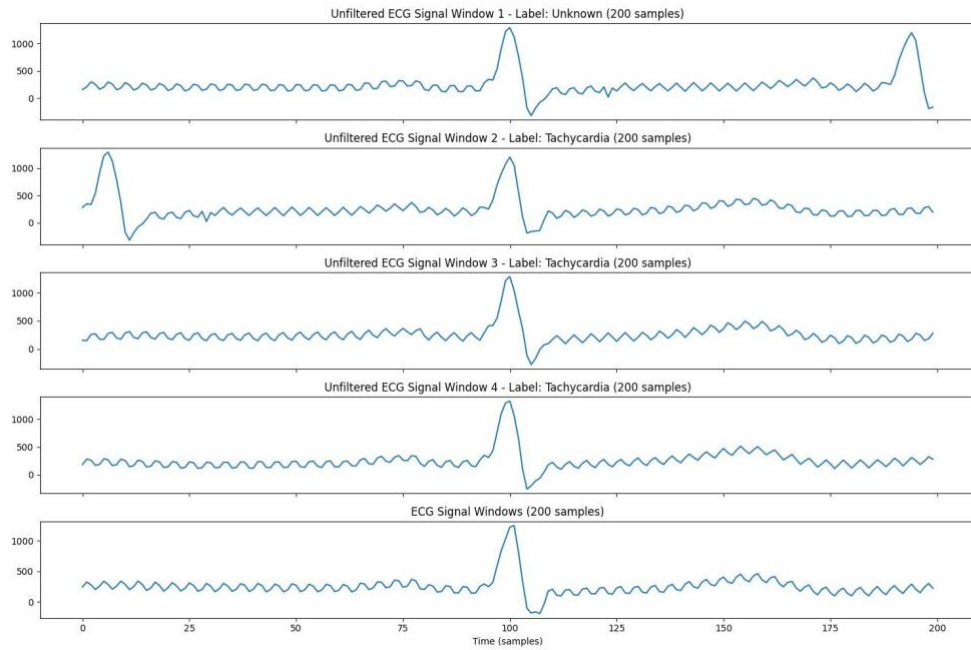


Figure 40: Unfiltered Windows ECG Signals

To visualize the windows crafted using the detected peaks and their corresponding labels, graph of each detected window was plotted in 41 (unfiltered) and 42 (filtered) with their corresponding classes for each detected peak and the window formed around it. These windows act as input vector of size 200 for the input of deep learning network.

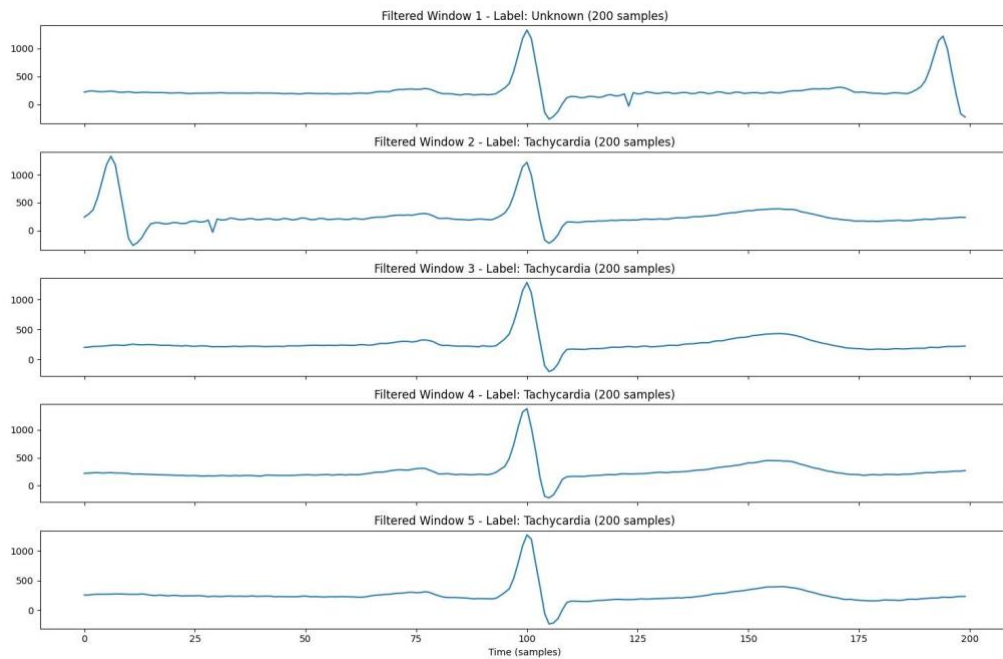


Figure 41: Filtered Windows ECG Signals

Class Imbalance

A pivotal aspect of our results discussion revolves around the incorporation of SMOTE analysis. Recognizing the potential impact of class imbalance on the performance of the network and the training process, we conduct a detailed examination of the role played by Synthetic Minority Over-sampling Technique (SMOTE) in addressing this challenge. The class imbalance, often inherent in biomedical datasets, can lead to biases in the model's learning process, particularly affecting its ability to accurately classify minority classes. Through the integration of SMOTE, we aim to rectify this imbalance by generating synthetic samples of the minority class, thereby enhancing the model's ability to learn from both majority and minority classes more effectively.

In the initial phase of Synthetic Minority Over-sampling Technique (SMOTE), the determination of the total number of oversampling observations, denoted as N , is established. Conventionally, this value is chosen to achieve a balanced binary class distribution, often set at a ratio of 1:1. However, this parameter can be adjusted based on specific requirements. The subsequent iterative process commences by randomly selecting a positive class instance. Following this selection, the K -nearest neighbors (KNN), typically set to a default value of 5, are identified for the chosen instance.

Subsequently, N instances are chosen from these K -nearest neighbors to facilitate the interpolation of new synthetic instances. The generation of synthetic instances involves calculating the difference in distance between the feature vector of the selected instance and its neighbors, utilizing a specified distance metric. This difference is then multiplied by a random value within the range $(0,1]$, and the resultant product is added to the original feature vector. This iterative process ensures the creation of diverse synthetic instances that contribute to mitigating class imbalance, as illustrated in the diagram 43.

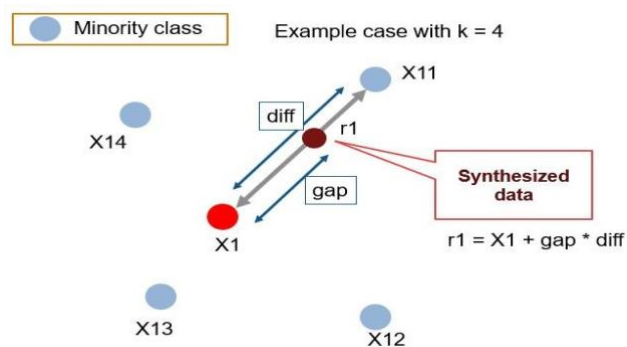


Figure 42: SMOTE Algorithm Diagram

The dataset exhibits a noticeable class imbalance, with the distribution of the following classes for cardiac rhythm:

1. Normal
2. Wolf-Parkinson-Syndrome (WPS)
3. Tachycardia
4. Other

- 5. Ventricular Fibrillation
- 6. Premature Ventricular Contractions
- 7. Myocardial Infarction

The class distribution for the used dataset is shown in Figure 44. Here we can clearly observe that normal beats are the majority class, while the other classes contain significantly less samples, resulting in a highly imbalanced dataset. This class imbalance introduces a potential bias during the training of AI models, affecting their ability to accurately distinguish between the different cardiac conditions.

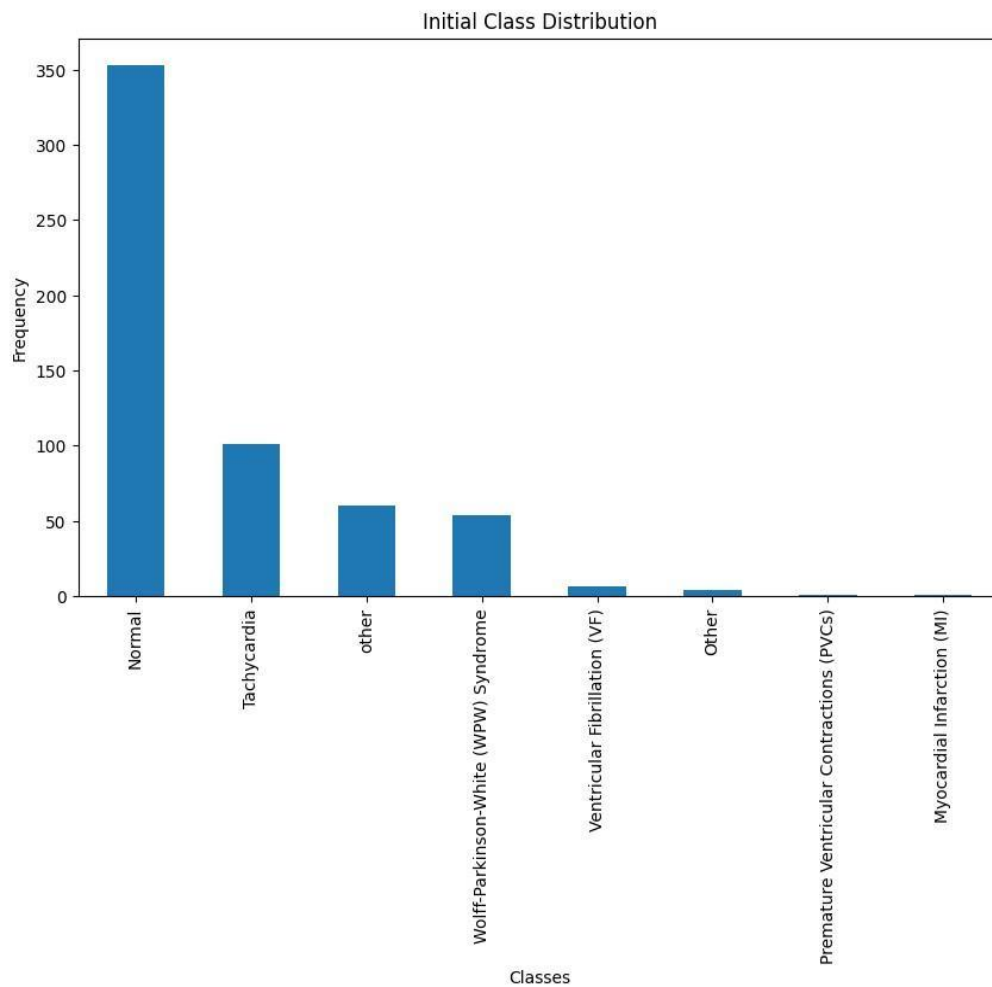


Figure 43: ECG Data Class Distribution without SMOTE

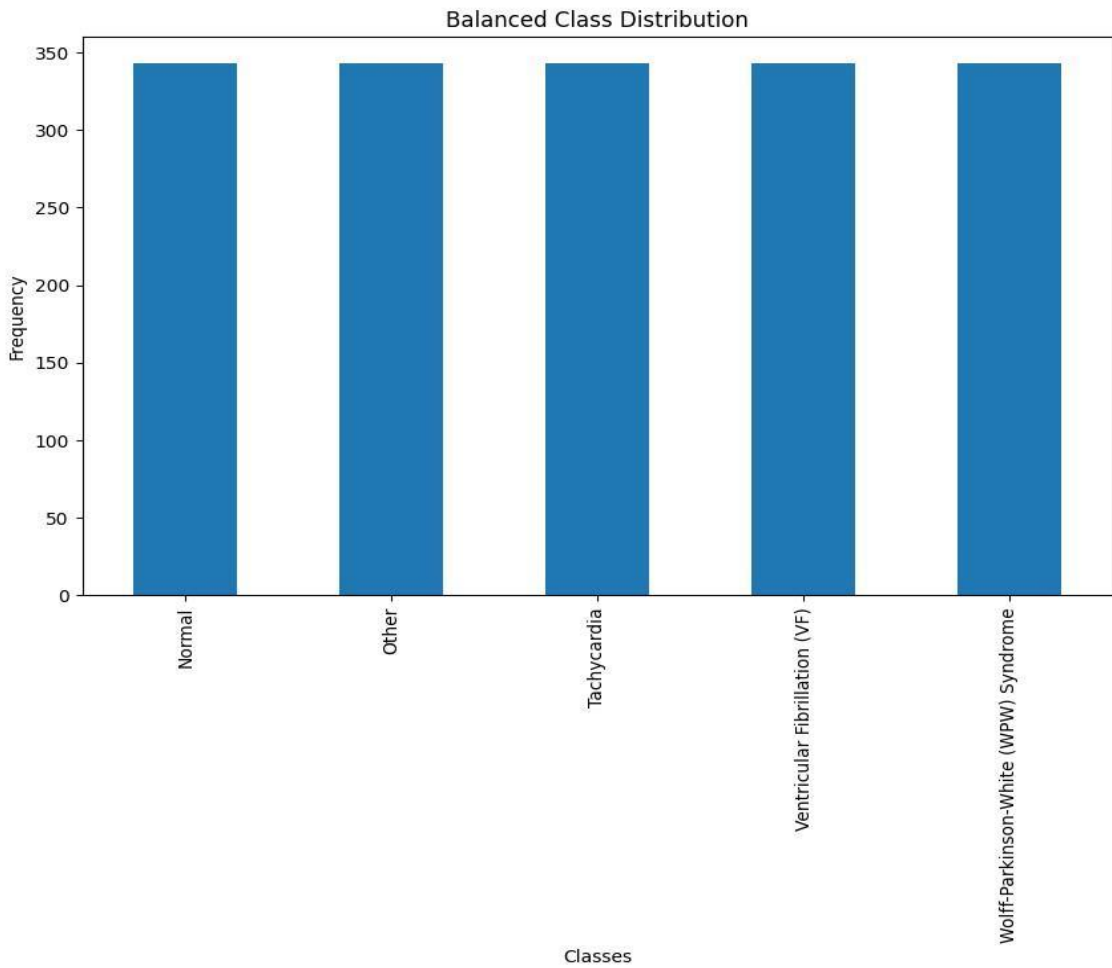


Figure 44 Class distribution after SMOTE

To combat this issue, we rebalanced the dataset by applying the SMOTE algorithm. By synthesizing new instances of the minority classes, SMOTE effectively balances the dataset, ensuring that each class contributes proportionally to the training of the CNN models. The graph of the distribution of the classes after applying SMOTE is illustrated in figure 45. The application of SMOTE thus enhances the model's capacity to discern between Normal, Tachycardia, Other, Ventricular Fibrillation and Wolf-Parkinson-White Syndrome with improved accuracy.

8.5. Convolution Neural Network:

A Convolutional Neural Network (CNN) is a deep learning architecture commonly used for image and time-series data processing, such as ECG signals. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input data, which makes them particularly powerful for tasks like pattern recognition in ECG signals. In this project, the CNN is employed to classify ECG heartbeats into different classes (e.g., Normal, Abnormal, or Other). Below is an in-depth explanation of how CNNs function and how the specific architecture used in this project works.

The input to the CNN is a 1D time-series signal from the ECG, where each heartbeat segment contains 200 samples. The input shape is (200, 1), which corresponds to the number of time

steps (samples) in each heartbeat window, representing 1 second of data (since the sampling frequency is 200 Hz).

Convolutional layers are the core building blocks of CNNs. They apply a set of filters (also called kernels) over the input data to extract important features. These features may include local patterns such as spikes, waves, and slopes that are characteristic of ECG signals. The CNN in this model has two convolutional layers.

First Convolutional Layer:

Uses 32 filters, which means the model will learn 32 different features or patterns from the ECG signal. These features might capture various waveform characteristics, such as the presence of QRS complexes, T-waves, and noise. The kernel size is set to 3, which means each filter looks at 3 consecutive samples at a time while scanning over the input data. A smaller kernel is used to capture fine-grained details in the ECG signal. Strides: The stride of the convolution, which determines how much the filter moves with each step, is implicitly set to 1. This ensures that the filters slide over each sample of the ECG signal, detecting patterns across the entire input. Activation Function: ReLU (Rectified Linear Unit) is applied to the output of the convolution. ReLU introduces non-linearity, which allows the model to learn complex relationships in the data by activating neurons only for positive values and setting negative values to zero. Output of Layer: After this convolutional layer, the output will be a 2D array where each of the 32 filters has learned different local features from the ECG input. Max pooling reduces the dimensionality of the data while preserving the most important features. It works by taking the maximum value from each 2-sample window in the output from the convolutional layer. Pooling Size: A pooling size of 2 means that the dimensionality of the data will be reduced by half, which helps in reducing the computational complexity and preventing overfitting. Why Use Pooling?: Pooling retains the strongest features while discarding less important details, ensuring the model focuses on the most relevant parts of the ECG signal. For example, it might retain the sharp peaks associated with R-peaks while reducing noise.

Second Convolutional layer:

Uses 64 filters, which allows the model to learn more complex and abstract features by building on the patterns detected by the first convolutional layer. Each filter now looks at the max-pooled output from the first layer. Kernel Size: The kernel size is kept at 3, focusing on learning fine-grained features. ReLU is again used to introduce nonlinearity, allowing for more complex feature extraction. The increase in the number of filters (from 32 to 64) allows the model to detect a wider variety of features. In early layers, CNNs learn simple patterns (such as sharp edges or transitions in an ECG signal), while deeper layers learn more abstract representations. Purpose: Like the first pooling layer, this second max pooling layer reduces the dimensionality of the data by half again, allowing the model to focus on the most important features detected by the second convolutional layer. After the convolutional and pooling layers, the output is still in a multi-dimensional array (a 1D feature map). This output needs to be "flattened" into a single long vector so it can be fed into the fully connected layers. The flattening layer takes the multi-dimensional feature maps and converts them into a 1D array, preparing it for further processing by the dense layers. Fully connected layers take all

the neurons from the previous layer and connect them to every neuron in the current layer. These layers are used to combine the learned features and make predictions. The dense layer has 128 neurons. Each neuron receives input from all the neurons of the previous layer (the flattened feature vector), and it outputs a value based on the activation function.

Activation Function:

ReLU is used to introduce non-linearity, enabling the model to learn complex representations of the ECG signal. Why Fully Connected?: The fully connected layer is where the high-level decision making happens. It takes the learned features from the convolutional layers and combines them to make meaningful predictions (in this case, classifying heartbeats). Softmax is applied to the output layer to produce probabilities for each class. Softmax ensures that the sum of all outputs equals 1, making it easier to interpret the output as a probability distribution. For example, if the output is [0.7, 0.3], the model predicts a 70% probability that the heartbeat is "Normal" and a 30% probability it is "Abnormal." The Adam optimizer is used to adjust the learning rate during training. Adam is an adaptive learning rate optimization algorithm that combines the advantages of two other popular optimizers: AdaGrad and RMSProp. The sparse categorical cross-entropy loss function is used since this is a multi-class classification problem. Cross-entropy measures the difference between the true label and the predicted probability.

Evaluation Metric

Accuracy is used as the primary evaluation metric. It measures the proportion of correctly classified heartbeats during training and testing. The CNN is trained over 200 epochs with a batch size of 32, meaning that 32 samples are processed in parallel during each iteration. The training process also involves the validation dataset, allowing the model to monitor its performance on unseen data during training. After training, the model is evaluated on the test set. The test set contains heartbeats that were not used during training, so evaluating the model on this set provides an unbiased estimate of how well the model generalizes to new, unseen data. The model's performance is measured in terms of Test Loss: The error between the predicted labels and the true labels on the test set and test Accuracy: The percentage of heartbeats that were correctly classified by the model on the test set.



9. Results and Discussion

This section presents the outcomes of the work in making an Electrocardiogram (ECG) data acquisition device for diagnostic purposes, enhanced by the integration of artificial intelligence (AI) models. The primary goal was to design a device capable of capturing accurate ECG signals and performing analysis of the data using deep learning techniques.

Analysis of the implications of the achieved results is performed, considering their significance in the context of cardiac healthcare. We thoroughly examine the device's performance metrics, and the AI models' accuracy in identifying cardiac anomalies, and discuss how these advancements may benefit medical practitioners and patients.

By connecting our results with existing literature, we critically evaluate the unique contributions of our device to ECG data acquisition. We also explore the broader implications of integrating AI models for diagnostic training, addressing challenges encountered during implementation, and proposing directions for future improvements.

The results showcase the successful development and validation of the ECG data acquisition device. We delve into its functionality, examining its ability to capture high-quality ECG data and the effectiveness of AI models for both diagnosis and training applications.

The discussion revolves around the most significant findings, addressing the impact of filtering and class balancing on model performance. Detailed comparisons between CNN architectures highlight their respective strengths and weaknesses in handling specific ECG patterns. The suitability of each architecture for real-world applications is discussed, considering factors such as interpretability and computational efficiency. Unexpected discoveries or challenges encountered during training and evaluation are critically analyzed, contributing to a comprehensive understanding of the research journey.

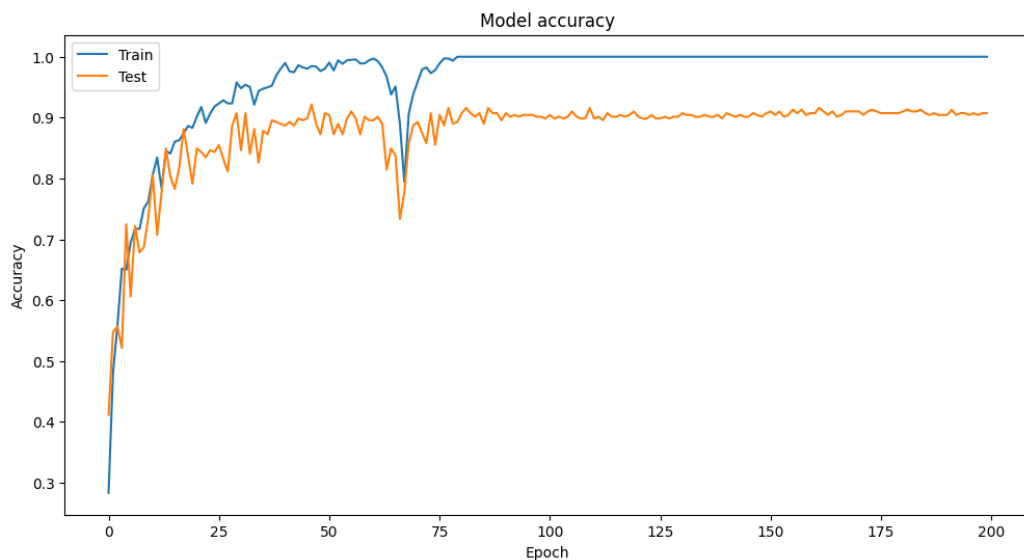


Figure 45: Training Curve generated for Accuracy with CNN for Classification



Training and Validation Accuracy:

The convolutional neural network (CNN) model developed for ECG heartbeat classification was trained over the data acquired using the pilot case and by using different metrics, its performance was accessed across five cardiac conditions: Normal, Other, Tachycardia, Ventricular Fibrillation (VF), and Wolff-Parkinson-White (WPW) Syndrome. This section provides an analysis of the model's learning process, highlighting the accuracy and loss trends during training, as well as its ability to generalize to new data. Additionally, the confusion matrix is presented to detail the classification performance for each condition, shedding light on the model's strengths and areas of confusion. These results offer insights into how effectively the model distinguishes between different cardiac conditions and the challenges it faces in certain classifications.

The graphs generated during the training of your convolutional neural network (CNN) model for ECG heartbeat classification are crucial for evaluating the model's learning process. The accuracy graph shows the performance over 200 epochs, with the training accuracy (represented by the blue line) quickly reaching 100%, while the validation accuracy (orange line) stabilizes around 90%. In the initial phase, both the training and validation accuracy increase rapidly, with training accuracy improving more sharply. The gap between these two accuracies, which becomes evident in the middle and final phases of training, indicates that while the model performs exceptionally well on the training data, it struggles to generalize as effectively to unseen data. This difference highlights the challenge of balancing model performance on both training and validation sets. The final validation accuracy plateaus around 90%, despite the perfect score on the training data. This behavior implies that the model is effective in learning the training data patterns but may need further adjustments, such as regularization techniques or data augmentation, to enhance its performance on new data.

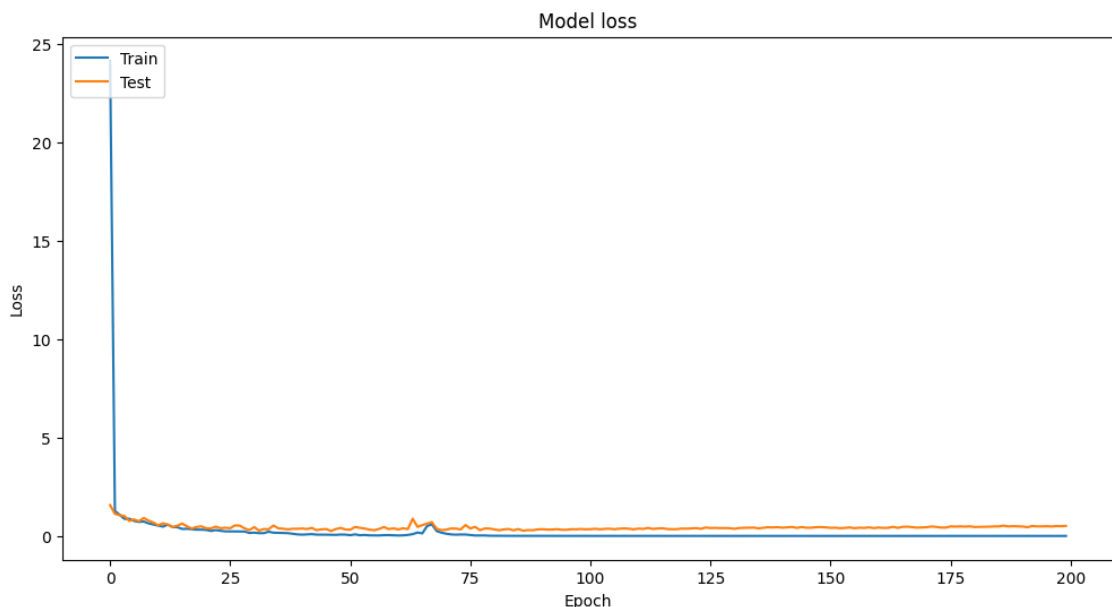


Figure 46: Training Curve generated for Loss with CNN for Classification



When looking at the model loss graph, the behavior becomes more apparent. The loss value for the training data (blue line) decreases rapidly and almost reaches zero, indicating that the model is highly confident in its predictions on the training set. The validation loss (orange line), on the other hand, decreases initially but stabilizes at a higher value, further reflecting that the model may be overfitting. The loss plateau in the validation data, combined with the widening gap between the training and validation losses, underscores the imbalance between learning and generalizing. The near-zero training loss confirms that the model fits the training data almost perfectly, while the consistent validation loss suggests a limitation in how well the model can generalize to new data points.

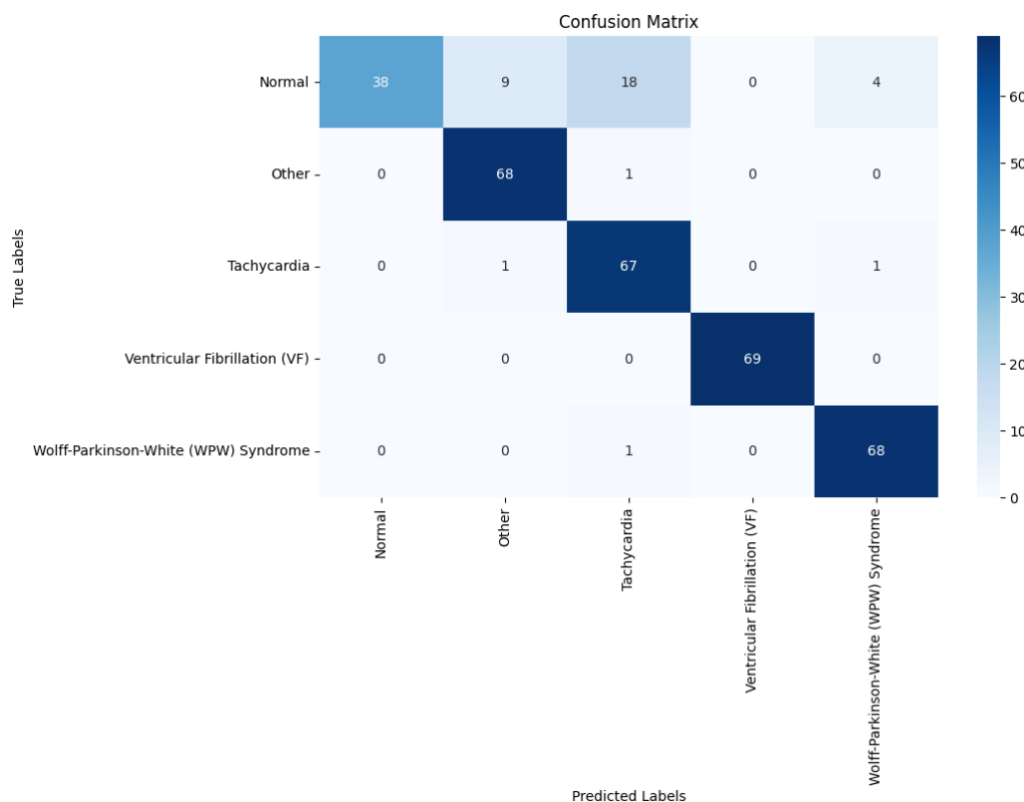


Figure 47: Confusion Matrix Plotted for the results

The confusion matrix in figure 47 highlights the performance of the classification model on ECG data, where five cardiac conditions—Normal, Other, Tachycardia, Ventricular Fibrillation (VF), and Wolff-Parkinson-White (WPW) Syndrome—are evaluated. The matrix demonstrates a relatively high accuracy in classifying "Other," "Ventricular Fibrillation," and "Wolff-Parkinson-White Syndrome," with true positive counts of 68, 69, and 68, respectively. However, there is observable confusion between the "Normal" and "Tachycardia" classes. Specifically, 38 "Normal" samples are correctly classified, but 18 samples are misclassified as "Tachycardia," and 9 are misclassified as "Other." This indicates a challenge for the model in distinguishing between "Normal" and "Tachycardia," likely due to overlapping features in their ECG signals.

The "Other" class is largely well classified, with only one instance being misclassified as "Tachycardia." Similarly, the "Tachycardia" class shows a high accuracy with 67 correctly



classified samples, but it experiences minor misclassifications, with one sample misclassified as "Other" and one as "WPW Syndrome." The "Ventricular Fibrillation" class exhibits perfect classification, with all 69 samples correctly identified.

The model demonstrates strong classification ability in certain classes but struggles in differentiating between more closely related conditions like "Normal" and "Tachycardia." The results suggest that further model refinements or additional features could improve class separability, particularly between classes with similar ECG signal characteristics.

Considering these trends, it's clear that the model's architecture or training process might benefit from adjustments. The perfect score on the training set and the relatively high performance on the validation set (90% accuracy) suggest that the model is learning important features.



10. Conclusion and Future Work

10.1. Conclusion

In conclusion, the analysis of the training and validation graphs reveals that while the model performs exceptionally well on the training data, achieving very high accuracy, there is evidence of overfitting, as the validation accuracy plateaus around 90%. This gap between training and validation performance indicates that the model is learning the training data too closely, potentially at the expense of generalizing to new, unseen data.

10.2. Future Work

Looking ahead, there are several directions in which this research can expand. One immediate area of future work involves the exploration of more advanced techniques to handle class imbalance. While SMOTE has been effective, newer algorithms and methods could offer even better performance, especially when dealing with large-scale datasets.

Another avenue for exploration is the optimization of CNN architectures for edge devices. The current research has established a baseline for efficiency and accuracy, but there is always room for improvement, particularly in terms of computational speed and power consumption. Lightweight neural network architectures, such as MobileNets or SqueezeNets, could be investigated for their suitability in ECG signal analysis.

In addition to technical advancements, future work should also focus on the clinical validation of the device. Collaborations with healthcare professionals and institutions to conduct comprehensive clinical trials will be essential to assess the practical utility and accuracy of the device in a variety of realworld settings. Such studies could help in refining the device's algorithms further, ensuring they are robust across diverse patient populations and clinical conditions.

Furthermore, advancements in explainable AI could be incorporated into future iterations of the device to enhance the trust and transparency of AI-assisted diagnoses. This is particularly important in the medical field, where understanding the rationale behind a diagnosis is as crucial as the diagnosis itself.

Interoperability with existing healthcare systems and electronic health records (EHRs) is another critical area for development. Ensuring that the device can seamlessly integrate with various EHR systems will facilitate a smoother workflow for healthcare providers and enable more holistic patient care.

In terms of hardware, exploring the use of next-generation sensors and improving the design for better signal acquisition could lead to more accurate and reliable ECG readings. The potential for miniaturization could also be explored, making the device more portable and convenient for patients who require continuous monitoring.

Finally, a significant portion of future work should be dedicated to security and privacy concerns. As with any healthcare device that handles sensitive data, ensuring the security of patient information in compliance with regulations such as HIPAA and GDPR is paramount. Research into advanced encryption methods and secure data transmission protocols will be critical to protect against data breaches and unauthorized access.



In conclusion, the successful development of the ECG data acquisition device represents a step forward in cardiac healthcare technology. The incorporation of AI models has shown promising results, and there is a clear path ahead for further improvements. With continued research and development, the full potential of this technology can be realized, leading to better patient outcomes and a new standard in cardiac care and diagnostics.



11. References

- [1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, 'Internet of Things (IoT): A vision, architectural elements, and future directions,' *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.
- [2] A. Shukla, A. Verma, and P. Gupta, 'IoT-based real-time ECG monitoring system,' in *Proc. IEEE Int. Conf. on Inventive Computation Technologies*, pp. 524–528, 2021.
- [3] Maxim Integrated, 'MAX30102 Pulse Oximeter and Heart-Rate Sensor for Wearable Health,' Datasheet, 2019. [Online]. Available: <https://www.maximintegrated.com/>
- [4] Melexis, 'MLX90614 Infrared Thermometer Module for Non-Contact Temperature Measurements,' Datasheet, 2020. [Online]. Available: <https://www.melexis.com/>
- [5] Espressif Systems, 'ESP32 Series Datasheet,' 2020. [Online]. Available: <https://www.espressif.com/>
- [6] Raspberry Pi Foundation, 'Raspberry Pi 4 Model B Specifications,' 2019. [Online]. Available: <https://www.raspberrypi.org/>
- [7] P. Laguna, R. G. Mark, A. Goldberg, and G. B. Moody, 'A database for evaluation of algorithms for measurement of QT and other waveform intervals in the ECG,' in *Computers in Cardiology* 1997, pp. 673–676, 1997.
- [8] MIT-BIH Arrhythmia Database, 'PhysioNet: The Research Resource for Complex Physiologic Signals,' [Online]. Available: <https://physionet.org/>
- [9] N. Vanaja and J. Ramesh, 'Survey on ECG Signal Denoising Techniques,' *Procedia Computer Science*, vol. 143, pp. 700–707, 2018.
- [10] N. Arora and B. Mishra, "Origins of ecg and evolution of automated dsp techniques: a review," *IEEE Access*, vol. 9, pp. 140853–140880, 2021.