REVOLUTIONIZING REMOTE HEALTH MONITORING: AUTONOMOUS DETECTION OF CARDIAC ABNORMALITIES WITH CUSTOMIZED DIETARY PLANNING

Pannila Vithanage Poorna Prabhathiya Senadheera

BSc (Hons) in Information Technology Specializing in Software Engineering

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

August 2024

REVOLUTIONIZING REMOTE HEALTH MONITORING: AUTONOMOUS DETECTION OF CARDIAC ABNORMALITIES WITH CUSTOMIZED DIETARY PLANNING

Final Report

Pannila Vithanage Poorna Prabhathiya Senadheera

IT21126888

BSc (Hons) in Information Technology Specializing in Software Engineering

Department of Software Engineering

August 2024

DECLARATION

I declare that this is my own work, and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as article or books).

Name	Student ID	Signature
Senadheera P.V.P.P	IT21126888	[illegible]

The above candidate is carrying out research for the undergraduate Dissertation under my supervision.

[illegible]
Signature of Supervisor
(Dr. Dilshan De Silva)

23/8/2024
Date

ABSTRACT

This research presents a novel approach to reconstructing multiple ECG leads from a single lead input, with a focus on enhancing accessibility and efficiency in cardiac monitoring. Utilizing a deep learning model based on a modified Attention U-Net framework, the study successfully reconstructs a complete set of 12-lead ECG signals from lead I. The process begins with data acquisition using a customized ambulatory ECG device, which records the lead I signal. This signal is then processed by the deep learning model, which reconstructs the remaining ECG leads with high accuracy, as demonstrated by strong Pearson correlation coefficients and R^2 values. The proposed method addresses the limitations of traditional ECG acquisition, which typically requires multiple electrodes placed at specific anatomical locations. By reducing the hardware complexity to a single lead, this solution offers a more user-friendly and cost-effective alternative without compromising diagnostic quality. This innovation has significant implications for remote health monitoring, particularly in settings where conventional 12-lead ECG setups are impractical. The results suggest that this technology could play a pivotal role in the future of portable and accessible cardiac care.

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Dilshan De Silva, for his constant guidance and unwavering support, which were instrumental in the successful completion of my research. His insightful feedback and encouragement have been invaluable throughout this journey. Additionally, I would like to extend my heartfelt thanks to Dr. P.V.W. Senadheera, whose generous assistance in connecting us with cardiology specialists and continuous support played a crucial role in the advancement of this project. I am deeply appreciative of their contributions, which have greatly enriched my research experience. I would also like to extend my sincere thanks to my team members for their dedication, collaboration, and hard work, which were essential to the success of this project. Finally, I would like to thank my family, friends, and colleagues for their encouragement and support throughout this project.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

Abbreviation	Description
WHO	World Health Organization
ECG	Electrocardiogram
DWT	Discrete Wavelet Transform
LSTM	Long Short-Term Memory
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
GAN	Generative Adversarial Network
USB	Universal Serial Bus
GPU	Graphics Processing Unit
ReLU	Rectified Linear Unit
ADAM	Adaptive Moment Estimation
GB	Gigabytes
RAM	Random Access Memory
MSE	Mean Squared Error
MAE	Mean Absolute Error
SDK	Software Development Kit
IDE	Integrated Development Environment
UI	User Interface
API	Application Programming Interface
HTTP	Hypertext Transfer Protocol

1. INTRODUCTION

1.1 Background & Literature Survey

A malady is a pathological state that is highly dangerous for a person's general health and can lead to a variety of problems, including pain, psychological discomfort, functional impairment, social difficulties, and even death. Remarkably, diseases comprise a significant fraction of the causes of death worldwide. The World Health Organization (WHO) states that the main causes of mortality rates worldwide are chronic illnesses. Ischemic heart disease alone was responsible for almost 8.89 million deaths in 2019, illustrating this trend. But when all cardiovascular diseases are considered, the annual death toll increases dramatically to over 17.9 million [1]. This emphasizes how important diseases are in determining the state of world health, especially chronic ones.

In order to lessen the effects of cardiac irregularities and improve the preservation of human life, a methodical strategy incorporating ongoing health monitoring must be put into place. This involves patients following a specified course of action that includes routinely visiting specialty cardiology clinics, getting cardiac tests as directed by physicians, and consulting cardiologists as needed. The goal of this all-encompassing approach is to provide a systematic framework for ongoing health surveillance that will guarantee the early identification, treatment, and management of cardiac problems in order to maximize patient outcomes.

In modern times, remote health monitoring systems are used all over the world and play a crucial role as information technology solutions to handle various medical difficulties. Doctor channeling, advice for physical examinations, and real-time monitoring of vital health indicators including electrocardiogram (ECG) are just a few of the functions that these systems enable. Incorporating these cutting-edge technological solutions has the potential to significantly improve accessibility, optimize healthcare delivery, and encourage proactive health management practices.

The ECG becomes a vital diagnostic method when it comes to cardiovascular illnesses. This diagnostic modality is essential for understanding the complex electrical activity patterns of the heart and for quickly identifying and diagnosing a range of cardiac disorders. ECG testing stands out for being non-invasive, painless, and quick to perform. It is a vital component in the diagnostic methods. The availability of a wide range of ECG tests meets specific needs and increases the diagnostic technique's adaptability and application in cardiovascular healthcare.

According to the literature that is currently available, there are three main forms of ECG testing that are classified: resting ECG, ambulatory ECG, and exercise stress test [2]. In order to minimize the impact of extraneous muscle electrical impulses during the process, the subject must be positioned supine throughout the execution of a resting ECG test. An ECG test performed while at rest usually takes five to ten minutes to complete. On the other hand, an ambulatory ECG uses a portable recorder that is attached for at least a day to allow for free mobility while registering sporadic symptoms that would not show up in a static resting ECG. To determine continued cardiac function, doctors may advise ambulatory ECGs for patients experiencing intermittent symptoms or those recuperating from a heart attack. In order to improve the association between individual experiences and the collected ECG data, patients are asked to record symptoms throughout this time. The third variation, the exercise stress test, examines heart function under physical activity. It is usually conducted through stationary exercise, like riding a bike or walking on a treadmill. This dynamic ECG version, which lasts for about 15 to 30 minutes, can also include pharmaceuticals to see how they affect heart function [2].

In the medical field, there are two basic methods used to acquire ECGs. Table 1 provides a detailed description of these two different approaches, including the electrode arrangement used and the resulting output ECG leads.

TABLE 1: Comparative analysis of main ECG mechanisms

ECG Mechanism	Number of Electrodes	Output ECG Leads
Standard 12-Lead ECG	10	I, II, III, aVR, aVL, aVF, V1, V2, V3, V4, V5, V6
3-Lead ECG	3	I, II, III

Four limb electrodes and six chest electrodes must be carefully positioned in order to obtain a 12-lead ECG [3], [4]. Figure 1 shows the standard 12-lead ECG placement. In the three-dimensional realm of cardiac activation, each of these 12 ECG leads represents a unique vector [5]. This arrangement allows for a more detailed analysis of the heart, defining its anterior, lateral, and inferior sections. Interestingly, changes seen in certain ECG leads for each anatomical region offer important information about the localized issues related to cardiac activity [6].

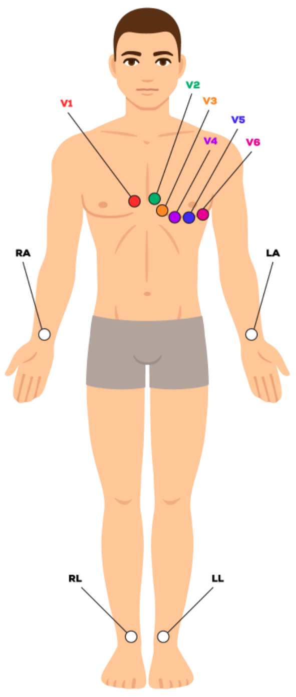


Figure 1: 12 lead ECG placement

Source: [7]

1.2 Research Gap

Five important studies in the field of ECG reconstruction are examined in the thorough literature evaluation of earlier research projects. Research indicates the potential for reconstructing accurate ECGs with fewer leads [8], [9], [10], [11]. Kapfo *et al.* [10], for instance, combined a discrete wavelet transform (DWT) with Long Short-Term Memory (LSTM) networks in a patient-tailored system to reconstruct standard 12-lead ECGs from just three input leads, demonstrating strong correlation and low error in their results. Similar success with LSTMs was reported by Jangjay *et al.* [11]. Nonetheless, practical hurdles exist with these methods since they still necessitate electrodes across the chest and limbs. The wearables industry demands solutions that further streamline data collection.

To address this challenge, Gundlapalle and Acharyya [12] introduced a novel technique utilizing Convolutional Neural Network (CNN)s, LSTMs, and Multilayer Perceptron (MLP)s to reconstruct a full 12-lead ECG from the singular input of lead II. They achieved noteworthy correlation and regression coefficients. Yoon *et al.* [13] explored the use of a generative adversarial network (GAN) for single-lead reconstruction, employing a U-net and patch discriminator architecture to attain favorable signal quality. To our knowledge, these represent the predominant studies dedicated to single-lead ECG reconstruction.

However, significant limitations exist in both works. The use of shorter segmented signals (1-2.5 seconds) potentially simplifies reconstruction but undermines practicality, as traditional ECGs necessitate 10-second recordings [14]. Seamlessly recombining these segments for real-world use poses an unresolved complexity. Additionally, the deep learning algorithms employed are both computationally intensive (particularly Yoon *et al.*'s GAN [13], [14], [15]) and can introduce instability. Furthermore, Gunlapalle and Acharyya's potential data leakage in their train/test split raises concerns about the generalizability of their findings.

Garg, Venkataramani and Priyakumar [16] discussed an approach to reconstruct multi-lead ECGs from a single lead using a modified Attention U-Net framework. It addresses the gap between reduced lead set data and standard 12-lead ECG interpretation. The model's ability to reproduce 11 leads from lead II with high Pearson correlation, low mean square error, and satisfactory R-squared values is emphasized, showcasing its potential for real-life application in disease classification tasks.

Based on the previously indicated corpus of research, three key features have surfaced as centers of interest for the investigation of novel ECG techniques. These include, notably, the effective recording of full 12-lead ECG patterns, the use of lead I ECG signal as the initial signal and reconstruct ECG signals with a duration of 10 seconds or longer. A thorough comparative study, as presented in Table 2, compares these unique characteristics over the range of the cited research works and the suggested solution, offering a nuanced comprehension of the contributions and developments in the field. The purpose of this comparison framework is to make it easier to assess the suggested solution considering important characteristics found in earlier studies.

Table 2: Comparison of former researches

Application Reference	**Use Lead I as the Initial Signal (Input)**	**Reconstruct ECG signals with a duration of 10 seconds or longer**	**Obtaining Standard 12-Lead ECG**
Reconstruction of 12-lead ECG using a DWT with LSTM network by Kapfo *et al.* [10]	X	✓	✓
Reconstruction of 12-lead ECG using LSTM network by Jangjay *et al.* [11]	X	✓	✓
Reconstruction of 12-lead ECG using CNNs, LSTMs, and MLPs by Gundlapalle and Acharyya [12]	X	X	✓
Reconstruction of 12-lead ECG using GAN, a U-net and patch discriminator architecture by Yoon *et al.* [13]	✓	X	✓

Reconstruction of 12-lead ECG using a modified Attention U-Net framework by Garg, Venkataramani and Priyakumar [16]	X	✓	✓
Proposed System (CardioFit AI)	✓	✓	✓

Given the dearth of prior studies that tackle the combination of the three critical characteristics that have been found, this study suggests developing an entirely novel smart tablet platform. This novel technology seeks to capture and generate 12-lead ECG signals from a single lead. We envision a practical and approachable way to detect cardiac abnormalities by utilizing this strategy. This innovative concept offers a holistic solution that combines state-of-the-art technology with useful clinical applications, marking a substantial advancement in the sector.

1.3 Research Problem

As was previously explained, there are two primary methods used in the present capture of ECG. The standard 12-lead ECG method requires exact electrode positioning on the body, which is thoroughly illustrated in Figure 1. Simultaneously, as Figure 2 shows, the 3-lead ECG method requires precise electrode placement on the body. Interestingly, disposable electrodes (gel conductive electrodes) are mostly used in both approaches, indicating how commonplace they are in modern ECG procedures. These standardized techniques for electrode placement and type are essential parts of the widely used ECG acquisition systems.

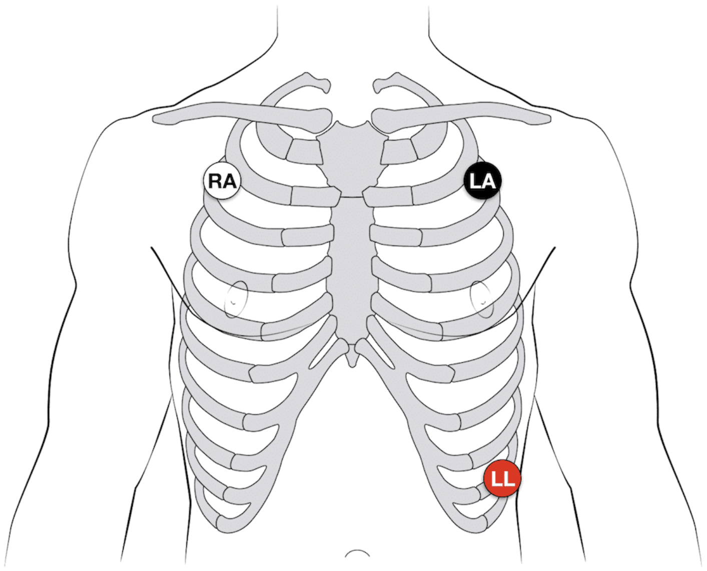


Figure 2: 3 lead ECG placement

Source: [18]

As such, current ECG acquisition technologies are not ideal for people who want to self-evaluate their heart health. Given this, I have identified the research problem to investigate the following question: "How to extract 12-lead ECG information from a single ECG lead?" By offering 2 or 3 electrodes as a workable and practical substitute, the research's primary goal of innovating and advancing ECG acquisition methodologies, particularly in the setting of self-assessment is emphasized in this problem description.

2. OBJECTIVES

2.1 Main Objectives

The primary objective of this work is to provide an advanced smart tablet platform that will make self-assessment ECG tests easier. The ultimate objective of the research is to take a whole 12-lead ECG pattern straight out of a single ECG lead. This research goal emphasizes a dedication to developing and reinventing ECG testing techniques, especially when it comes to self-evaluation. The platform that is being imagined is expected to provide a revolutionary and approachable way to gather vital cardiac data.

2.2 Specific Objectives

There are four specific objectives should be reached in order to achieve the main objective mentioned above.

- Determine the optimal ECG lead for obtaining accurate readings suitable for extracting other leads.
    - Gathering the signal of optimal ECG lead which contains more cardiac data is the objective of this phase. To acquire ECG signal, a customized ambulatory ECG device is to be used.

- Training the model to re-construct the remaining 11 ECG leads.
    - A deep learning model is to be built with the Attention U-Net Framework [16] which takes the acquired ECG signal from the previous objective as the input and re-constructs the remaining 11-leads by analyzing the cardiac data contained in the input.

- Design and implement a smart tablet-based platform for ECG viewing, facilitating user-friendly access and interpretation of cardiac data.
    - The outcome of this phase is a smart tablet-based platform which contains user-friendly interfaces and an ECG viewer that is easy to view 12-lead ECG patterns. Also, this platform gives a full guidance for the users during the ECG assessment.

3. METHODOLOGY

The proposed system demonstrates the capability to reconstruct standard ECG leads by thoroughly analyzing a single ECG lead. The complexities of this analysis procedure are explained in Figure 3, which offers a graphic depiction of the methodical and complex approach used by this specific system component. The standard ECG leads reconstruction process is demonstrated by this graphical representation, which also helps to clarify the analytical details.

3.1 Feasibility study

- Project members should have some expert knowledge in flutter mobile application development framework, knowledge in software architectures and a good understanding on server-side development such as Flask.
- The proposed sub-component must operate flawlessly, free of errors or failures. It should also be more reliable, offer higher performance, and be cost-effective. The component should be designed to minimize resource costs while meeting all necessary requirements.

3.2 Component Diagram

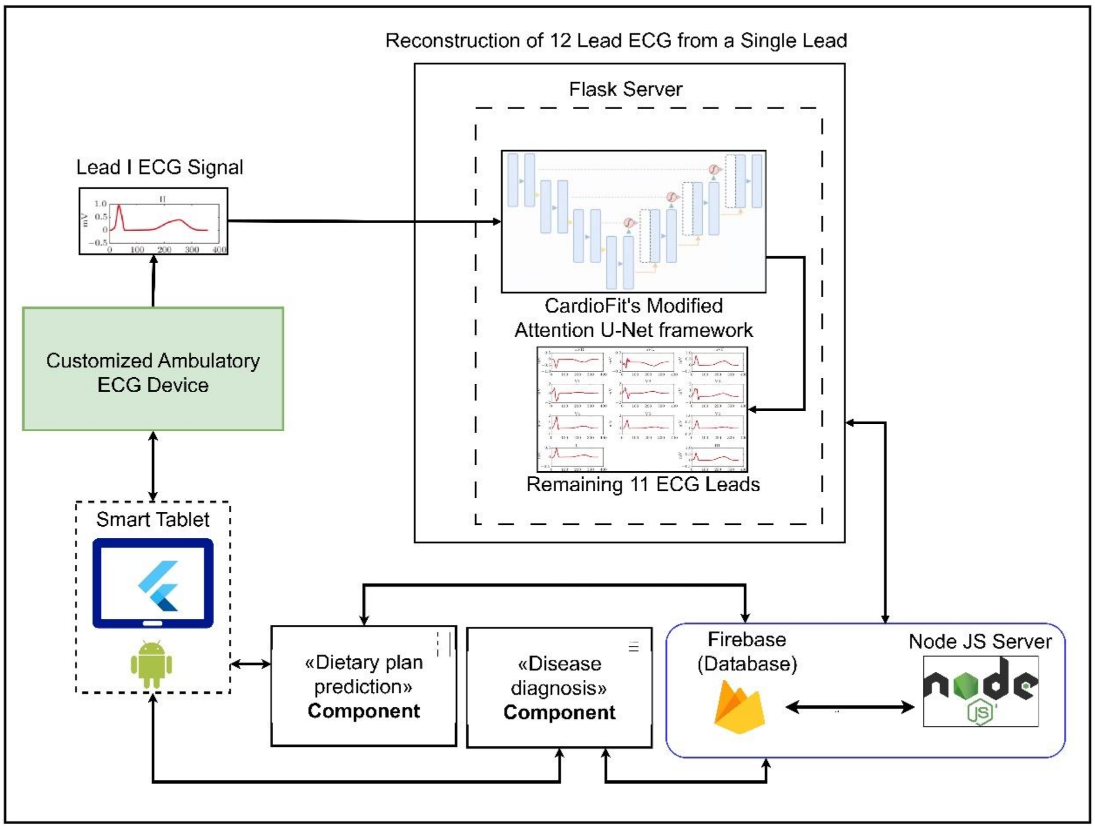

Reconstruction of 12 Lead ECG from a Single Lead
Flask Server
Lead I ECG Signal
CardioFit's Modified Attention U-Net framework
Remaining 11 ECG Leads
Customized Ambulatory ECG Device
Smart Tablet
«Dietary plan prediction» Component
«Disease diagnosis» Component
Firebase (Database)
Node JS Server

Figure 3: Reconstruction of 12-lead ECG from a single lead component diagram

As part of the suggested remedy, a customized ambulatory ECG device is identified as a component part of this research component. The device is essential to the acquisition of ECG signals since it integrates with the smart tablet in a fluid manner. An exhaustive list of information and specifications for this portable ECG machine is thoroughly displayed in Table 3.

TABLE 3: Ambulatory ECG device information

Operating Voltage	5.0V
Included modules	• AD8232 ECG Module (amplify and process the lead I ECG signal)

	• Arduino UNO Rev3 with ATmega328p microcontroller (convert the analog ECG signal to a digital signal)
Connection with Tablet Device	Universal Serial Bus (USB) Type-C
Number of electrodes connected	3

The ECG lead I signal obtained from the specified device is effortlessly transferred to the smart tablet application. That ECG signal is then subjected to a second step in the application framework, where it is processed using a deep learning model that is intended to reconstruct the remaining ECG leads. Based on the Attention U-Net Framework, which evolved from CNN principles, the deep learning model's architecture is built [16]. The computational power of an NVIDIA Tesla P100 graphics processing unit (GPU) is utilized to increase the effectiveness of the training process for this deep learning model. This significantly accelerates the computational effort and speeds up the training stage.

During the final phase of the procedure, the 7 leads (II, V1, V2, V3, V4, V5, V6) that were reconstructed using the deep learning model and other calculated 4 leads (III, aVR, aVL, aVF) using lead I & II are combined with the ECG lead I that was obtained from the ambulatory ECG device. This combined ECG data set is sent to the Firebase database. This final phase makes sure that the enriched ECG data is properly compiled and stored, which makes it easier to retrieve and analyze the data later on inside the larger telemedicine framework.

Table 4 provides a detailed overview of the technologies, architectures, and algorithms used in the palm analysis component of cardiac rhythm detection.

TABLE 4: Technologies, architectures and algorithms used.

Technologies	Flutter, Python, TensorFlow, Keras, Arduino
Architectures	Attention U-Net
Algorithms	CNN

3.3 Commercialization Aspects of the Product

- Commercialization Aspects of the Product:
    - The product under consideration, which is dubbed “Reconstructing Multiple ECG Leads from a Single Lead,” provides an appealing opportunity for commercialization, specifically in the medical diagnostics and telemedicine markets. This product aims at offering a complete 12-lead ECG reconstruction from one lead only, leading to its wide application where a fast and non-intrusive cardiac analysis is required.

- Market Demand:
    - The health care industry is increasingly embracing solutions that provide reliable diagnostic tools beyond the traditional hospital setting. In emergency rooms, outpatient clinics, and even remote healthcare facilities, having a 12-lead ECG reconstructed from one lead simplifies diagnosis and makes it very attractive. The main target market for this product includes healthcare providers in need of fast and accurate ECG tests without the bulkiness and sophistication of full scale 12 lead ECG machines.
    - With the increasing number of cardiovascular diseases globally, there is an increased demand for accessible and efficient diagnostic tools. This product provides a low-cost option for institutions that lack enough capital to purchase whole scale ECG equipment or for cases where quick response is needed like ambulances or remote health care facilities.

- Competitive Advantage:
    - The simplicity and effectiveness of this product has contributed to its competitive edge over others. In some cases, traditional 12-lead ECG systems could be limited by the many electrodes in use as well as time-

consuming setups especially during emergencies. This product however uses only one lead to reassemble a full 12-lead ECG, while maintaining result accuracy.

- Additionally, the use of complex deep learning algorithms for this reconstruction maintains high accuracy rates that pose a challenge to traditional forms of ECG systems. Reduced hardware complexity without sacrificing diagnostic quality sets this apart from other solutions currently on the market.

4. PROJECT REQUIREMENTS

4.1 Functional Requirements

1. The input of the ECG Reconstruction model should be an ECG signal with a duration of 10 seconds.
2. The remaining 11 leads should be acquired with high accuracy and displayed.
3. Reconstructed ECG data should be saved in the database for future reference.

4.2 Non-functional Requirements

1. Interfaces should be User-friendly.
2. The application should be reliable.
3. Results should be more efficient.

4.3 Data Requirements

1. PTB-XL large publicly available electrocardiography dataset [17]

4.4 System Requirements

Software requirements are meant to specify the software resources that have to be implemented on a system in order for the proposed system to operate as intended. The following are the requirements for the software specs of this proposed component.

1. Flutter to build the cross-platform mobile application.
2. Firebase as the real-time database

3. TensorFlow libraries to manipulate signal data and signal processing.
4. Keras to build the deep learning model.

Flask server to execute deep learning model and keep the connection with the mobile application.

4.5 Sequence Diagram

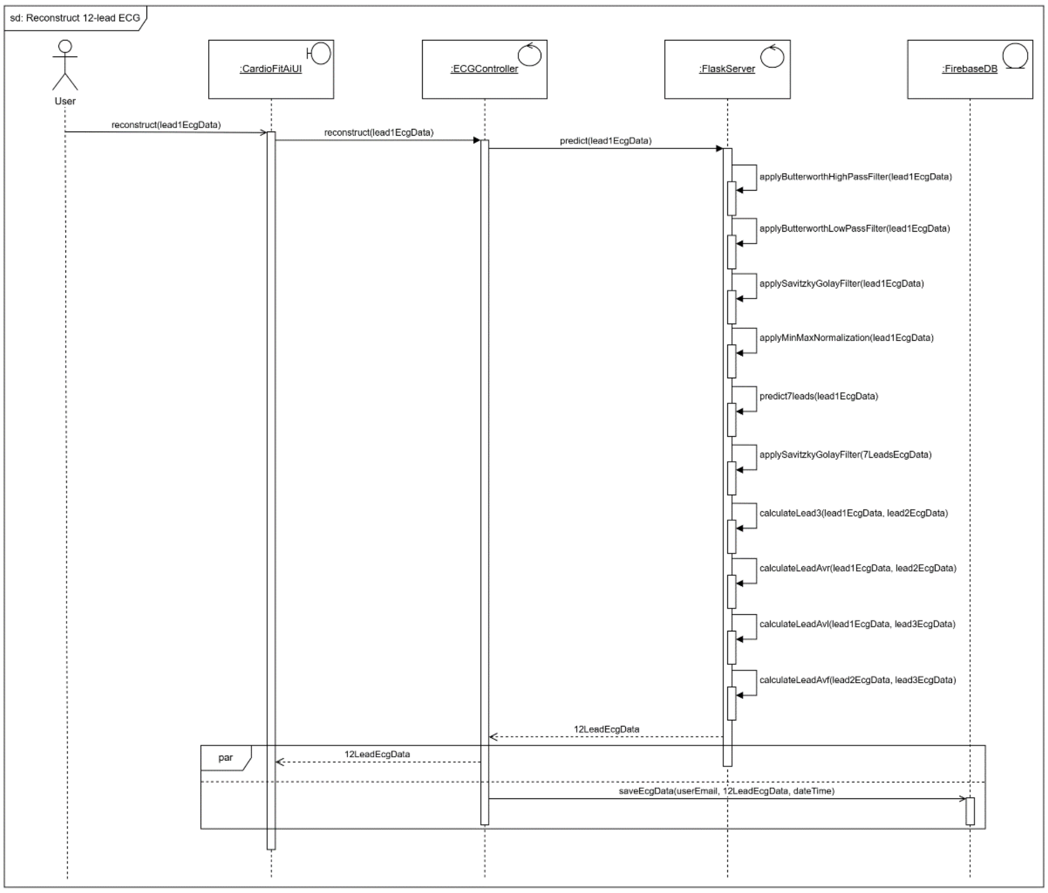
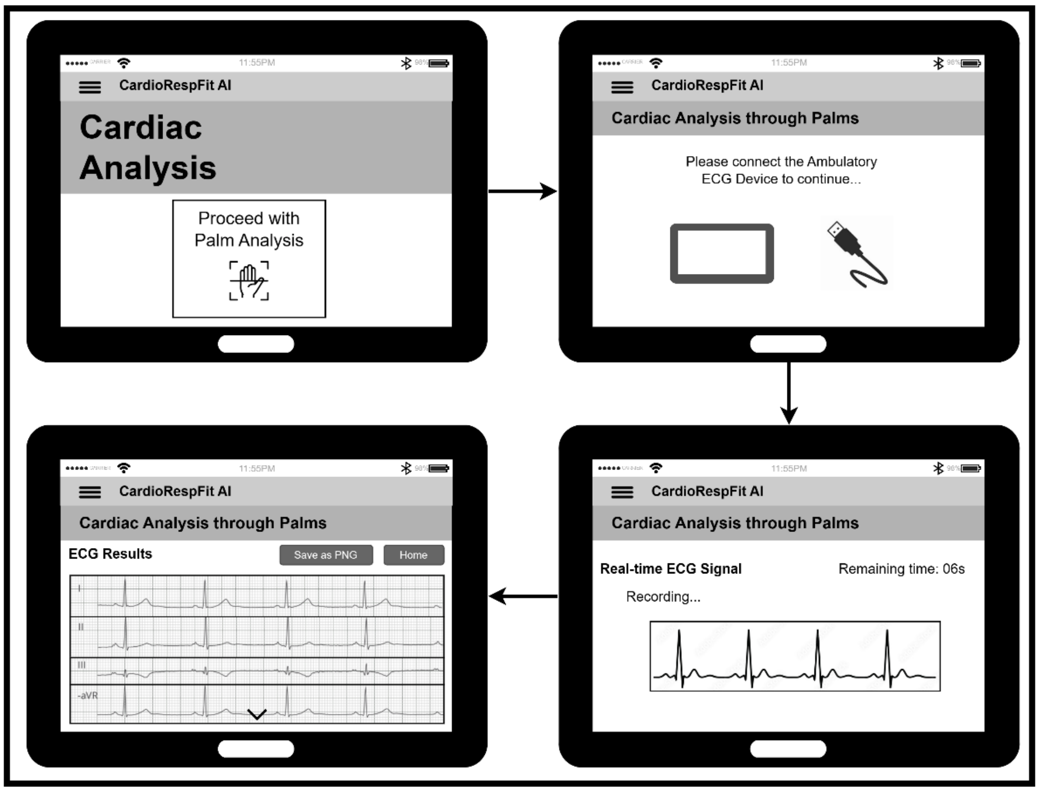

sd: Reconstruct 12-lead ECG
User
:CardioFitAiUI
:ECGController
:FlaskServer
:FirebaseDB
reconstruct(lead1EcgData)
reconstruct(lead1EcgData)
predict(lead1EcgData)
applyButterworthHighPassFilter(lead1EcgData)
applyButterworthLowPassFilter(lead1EcgData)
applySavitzkyGolayFilter(lead1EcgData)
applyMinMaxNormalization(lead1EcgData)
predict7leads(lead1EcgData)
applySavitzkyGolayFilter(7LeadsEcgData)
calculateLead3(lead1EcgData, lead2EcgData)
calculateLeadAvr(lead1EcgData, lead2EcgData)
calculateLeadAvl(lead1EcgData, lead3EcgData)
calculateLeadAvf(lead2EcgData, lead3EcgData)
12LeadEcgData
par
12LeadEcgData
saveEcgData(userEmail, 12LeadEcgData, dateTime)

Figure 4: Sequence diagram of reconstruct 12-lead ECG use case

4.6 Wireframes



CardioRespFit AI
Cardiac Analysis
Proceed with Palm Analysis
CardioRespFit AI
Cardiac Analysis through Palms
Please connect the Ambulatory ECG Device to continue...
CardioRespFit AI
Cardiac Analysis through Palms
Real-time ECG Signal
Remaining time: 06s
Recording...
CardioRespFit AI
Cardiac Analysis through Palms
ECG Results
Save as PNG
Home

Figure 5: Designed wireframes at the designing phase

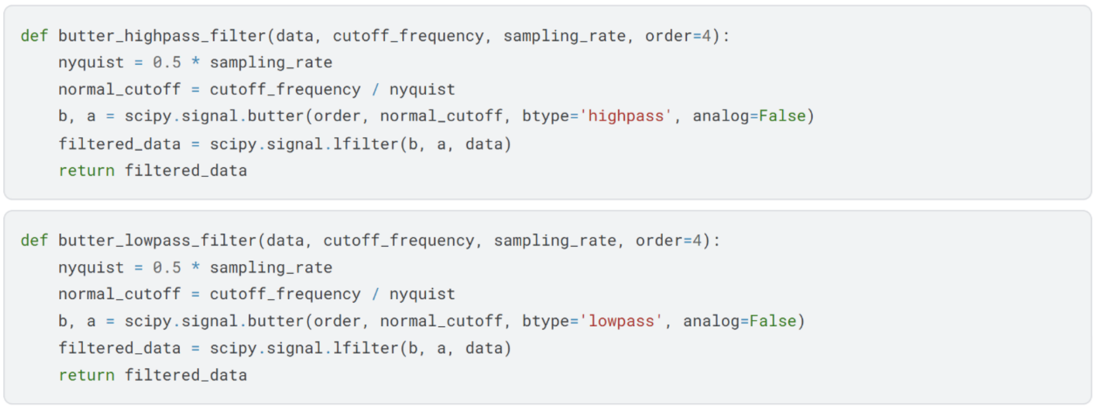
5. IMPLEMTATION AND TESTING

5.1 Data Acquisition and Data Pre-processing

For this research project to train the deep learning model, data acquisition is necessary. This is why PTB-XL dataset [17] which contains 21,799 clinical 12-lead ECGs from 18,869 unique patients were utilized. It should be noted that these data are gender balanced with males accounting for 52% and females 48%. Every ECG spanned over ten seconds, and it contained essential electrical cardiac activities. Raw signal data had a complete set of twelve leads as a basis for our analysis. The pre-processing steps such as standardization of data format, artifact correction and signal enhancement were all done to allow accurate feature extraction leading to further analysis.

Although the dataset contains ECG samples at both sampling rates of 100Hz and 500Hz, the ECG samples recorded at a frequency of 500 Hz were decided to use instead of those recorded at 100 Hz. I arrived at this decision because a higher sample rate of 500 Hz offers more detailed information on ECG signals that can lead to better analysis and interpretation.

Empty records were removed in order to improve the quality of data. Thereafter, several digital signal processing methods were used to help reduce noise and improve the clarity of signals. In particular, I employed a Butterworth [19] high-pass filter having a cut-off frequency of 0.5Hz for lowering the level of low-frequency noise components, whereas another complementary Butterworth low pass filter was also used with a cut-off frequency of 200Hz for getting rid of any high-frequency noise artifacts. Figure 6 demonstrates how Butterworth low-pass and high-pass filters are used.

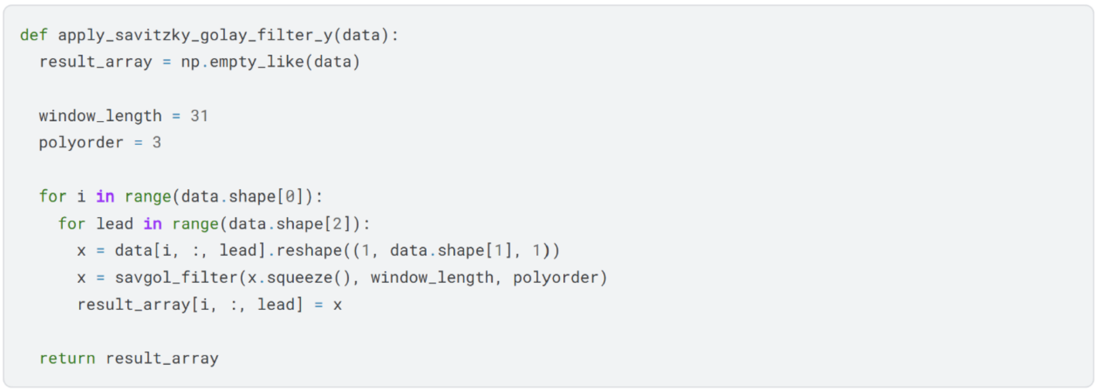


```python
def butter_highpass_filter(data, cutoff_frequency, sampling_rate, order=4):
    nyquist = 0.5 * sampling_rate
    normal_cutoff = cutoff_frequency / nyquist
    b, a = scipy.signal.butter(order, normal_cutoff, btype='highpass', analog=False)
    filtered_data = scipy.signal.lfilter(b, a, data)
    return filtered_data
```

```python
def butter_lowpass_filter(data, cutoff_frequency, sampling_rate, order=4):
    nyquist = 0.5 * sampling_rate
    normal_cutoff = cutoff_frequency / nyquist
    b, a = scipy.signal.butter(order, normal_cutoff, btype='lowpass', analog=False)
    filtered_data = scipy.signal.lfilter(b, a, data)
    return filtered_data
```

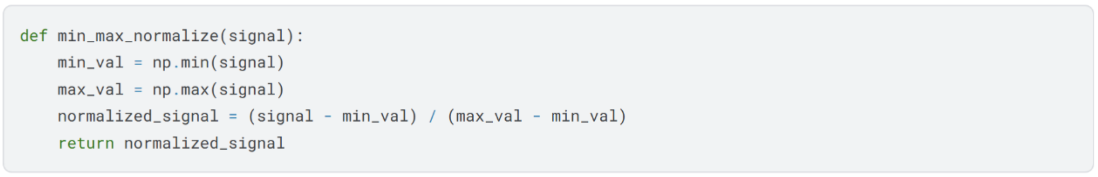
Figure 6: Application of the Butterworth filters in code level

Furthermore, to further refine the ECG signals, a Savitzky-Golay [20] filter was applied, smoothing out any remaining fluctuations or irregularities in the waveforms. Figure 7 represents the application of Savitzky-Golay filter in code level. To standardize the amplitude range across leads, Min-Max Normalization was implemented as Figure 8, scaling all ECG lead signals between 0 and 1. Subsequently, the dataset was partitioned into input (X) and output (Y) sets, with lead I designated as the input feature (X), and leads II, V1, V2, V3, V4, V5, and V6 forming the output features (Y). Finally, to facilitate model training and evaluation, the dataset was split into training and testing subsets at an 80%-20% ratio as the Figure 9, ensuring a robust assessment of model performance while preserving data integrity.



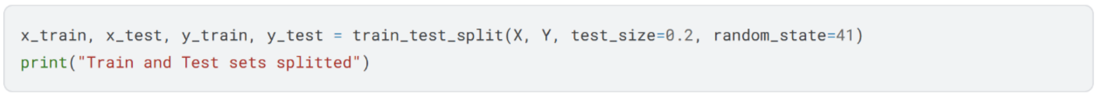
```python
def apply_savitzky_golay_filter_y(data):
  result_array = np.empty_like(data)

  window_length = 31
  polyorder = 3

  for i in range(data.shape[0]):
    for lead in range(data.shape[2]):
      x = data[i, :, lead].reshape((1, data.shape[1], 1))
      x = savgol_filter(x.squeeze(), window_length, polyorder)
      result_array[i, :, lead] = x

  return result_array
```

Figure 7: Application of Savitzky-Golay filter in code level



```python
def min_max_normalize(signal):
    min_val = np.min(signal)
    max_val = np.max(signal)
    normalized_signal = (signal - min_val) / (max_val - min_val)
    return normalized_signal
```

Figure 8: Application of Min-Max normalization in code level



```python
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_state=41)
print("Train and Test sets splitted")
```

Figure 9: Code segment of splitting the dataset

5.2 Deep Learning Model Implementation

Figure 10 illustrates the architectural blueprint of the deep learning model tailored for ECG multi-lead reconstruction. Designed to operate on individual ECG leads, each containing 5000 data points sampled at 500Hz over a duration of 10 seconds, the model aims to predict seven ECG leads, each also comprising of 5000 data points. The architecture primarily consists of convolutional layers, strategically arranged to extract hierarchical features from the input signals.

The model has the following convolutional layer's structure: firstly, 1D convolution layers which include Rectified Linear Unit (ReLU) activation functions are employed with kernel size of 3 and stride of 1 to capture local patterns in the input signals. Then, additional 1D convolution layers with ReLU activation functions and batch normalization are used, employing a kernel size of 3 and stride of 2 to down sample the feature maps so as to improve model efficiency. Lastly, I have used 1D transposed convolution with ReLU activations and batch normalizations characterized by a kernel size of 3 and a stride of 2 for signal reconstruction. In code level, Figure 11, Figure 12 and Figure 13 show how these convolution layers have been applied.

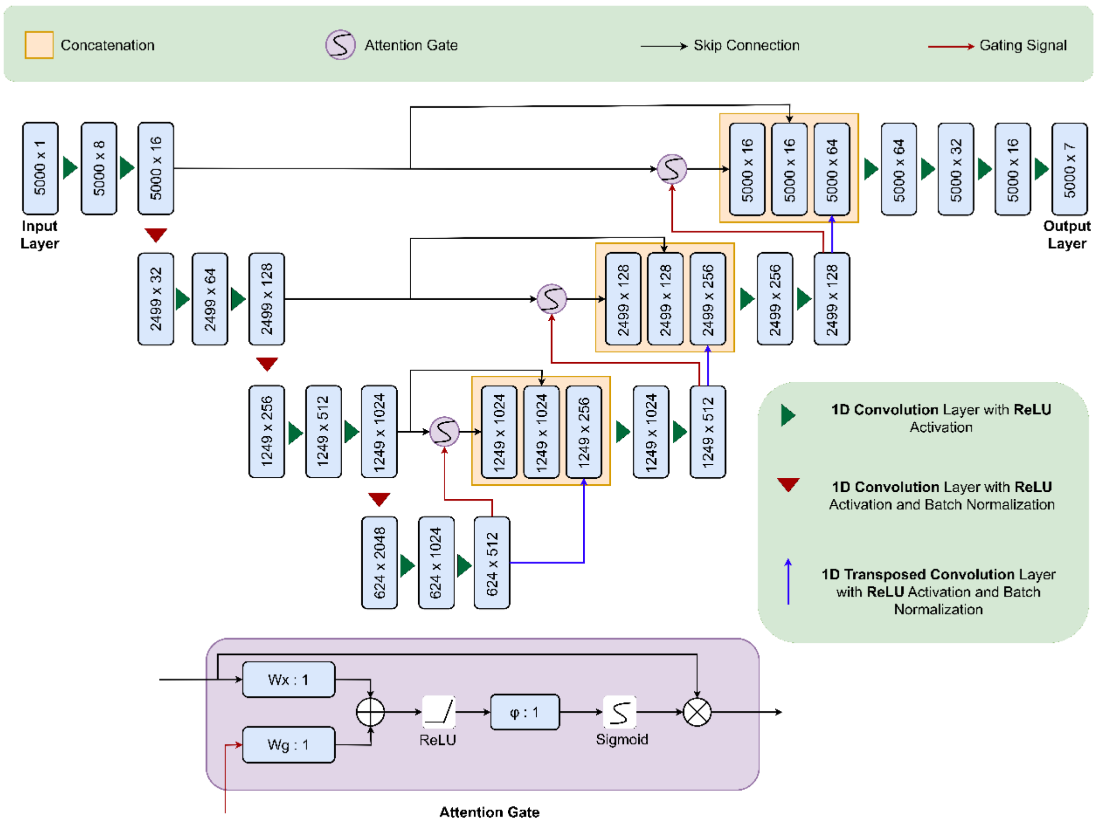


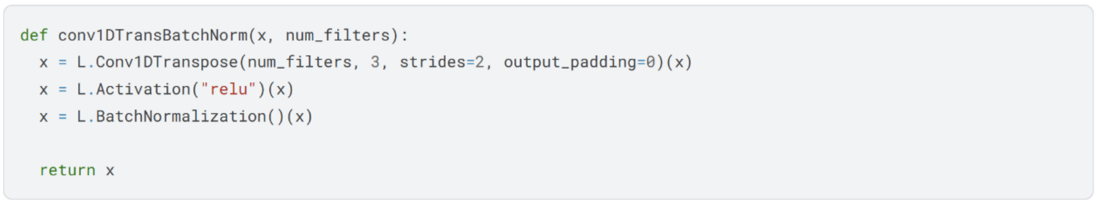
Figure 10: CardioFit's Modified Attention U-Net model architecture

```
def conv1D(x, num_filters):
  x = L.Conv1D(num_filters, 3, strides=1, padding='same')(x)
  x = L.Activation("relu")(x)

  return x
```

Figure 11: Application of the pure 1D convolution layer in code level

```
def conv1DBatchNorm(x, num_filters):
  x = L.Conv1D(num_filters, 3, strides=2)(x)
  x = L.Activation("relu")(x)
  x = L.BatchNormalization()(x)

  return x
```

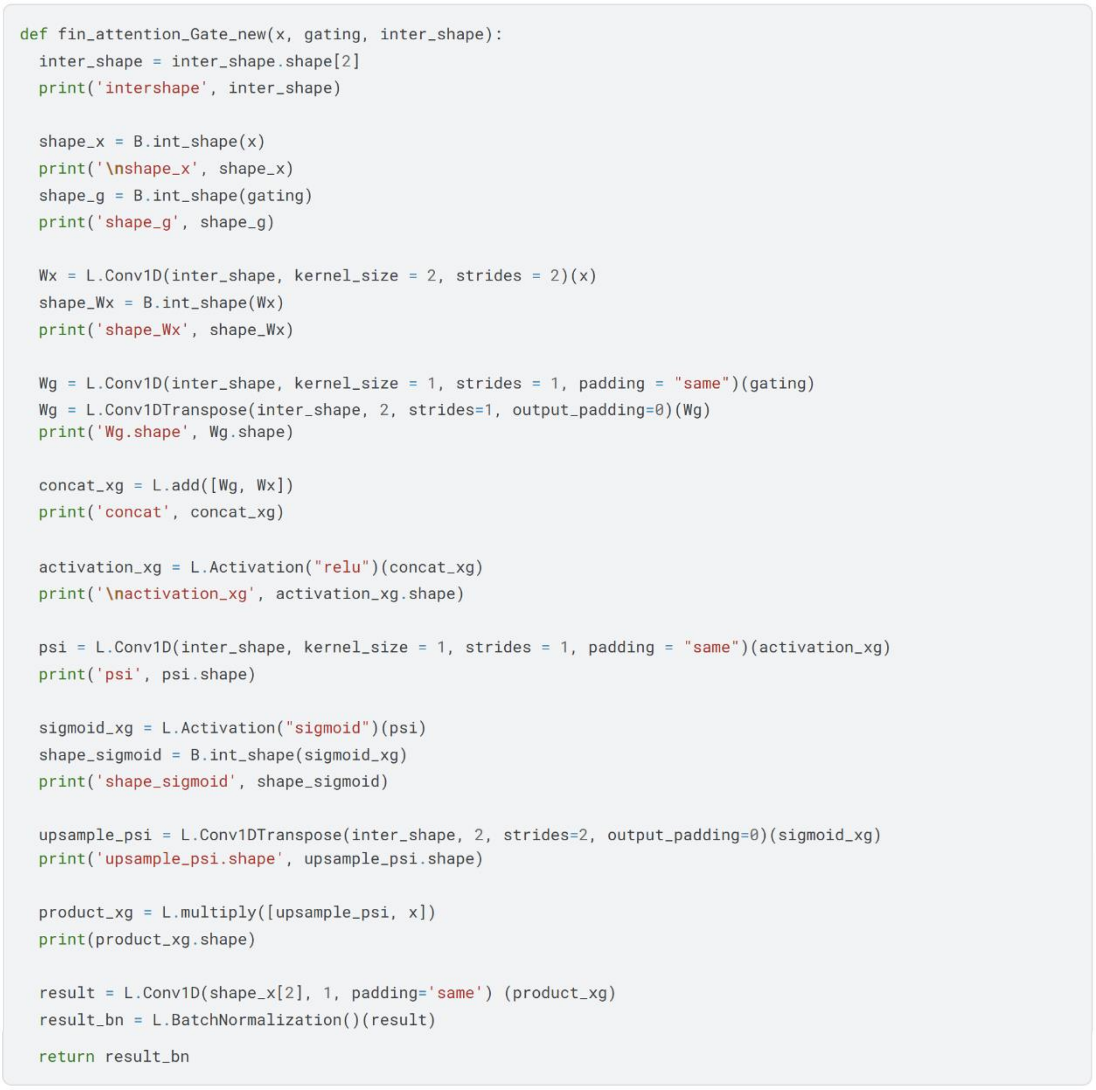
Figure 12: Application of the 1D convolution layer with batch normalization in code level



```
def conv1DTransBatchNorm(x, num_filters):
  x = L.Conv1DTranspose(num_filters, 3, strides=2, output_padding=0)(x)
  x = L.Activation("relu")(x)
  x = L.BatchNormalization()(x)

  return x
```

Figure 13: Application of 1D transposed convolution layer in code level

Furthermore, within the attention gate mechanism embedded within the architecture, intricate signal processing occurs in a sequential fashion. Initially, the skip connection and the gating signal are concatenated to incorporate both local and global context into the processing pipeline. Subsequently, the concatenated signal undergoes segmentation via the ReLU activation function, facilitating feature extraction and refinement. Following this, the processed signal is passed through 1D transposed convolution layers to promote signal reconstruction. The resultant signal is then subjected to a sigmoid activation function, enabling non-linear transformations to enhance signal fidelity. Finally, the processed signal is modulated by multiplying it with the skip connection, thereby integrating contextual information and fine-grained details to enrich the reconstructed ECG leads. The code level process of attention gate mechanism is shown in Figure 14.



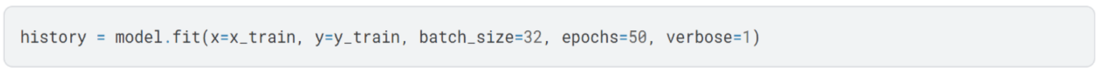
```python
def fin_attention_Gate_new(x, gating, inter_shape):
  inter_shape = inter_shape.shape[2]
  print('intershape', inter_shape)

  shape_x = B.int_shape(x)
  print('\nshape_x', shape_x)
  shape_g = B.int_shape(gating)
  print('shape_g', shape_g)

  Wx = L.Conv1D(inter_shape, kernel_size = 2, strides = 2)(x)
  shape_Wx = B.int_shape(Wx)
  print('shape_Wx', shape_Wx)

  Wg = L.Conv1D(inter_shape, kernel_size = 1, strides = 1, padding = "same")(gating)
  Wg = L.Conv1DTranspose(inter_shape, 2, strides=1, output_padding=0)(Wg)
  print('Wg.shape', Wg.shape)

  concat_xg = L.add([Wg, Wx])
  print('concat', concat_xg)

  activation_xg = L.Activation("relu")(concat_xg)
  print('\nactivation_xg', activation_xg.shape)

  psi = L.Conv1D(inter_shape, kernel_size = 1, strides = 1, padding = "same")(activation_xg)
  print('psi', psi.shape)

  sigmoid_xg = L.Activation("sigmoid")(psi)
  shape_sigmoid = B.int_shape(sigmoid_xg)
  print('shape_sigmoid', shape_sigmoid)

  upsample_psi = L.Conv1DTranspose(inter_shape, 2, strides=2, output_padding=0)(sigmoid_xg)
  print('upsample_psi.shape', upsample_psi.shape)

  product_xg = L.multiply([upsample_psi, x])
  print(product_xg.shape)

  result = L.Conv1D(shape_x[2], 1, padding='same') (product_xg)
  result_bn = L.BatchNormalization()(result)

  return result_bn
```

Figure 14: Implementation of the Attention Gate mechanism in code level

5.3 Deep Learning Model Training

The model underwent a training process that spanned 50 epochs. An epoch refers to one complete pass through the entire training dataset. By iterating over the dataset 50 times, the model had ample opportunity to learn the intricate patterns in the ECG signals. The training process was guided by the Adaptive Moment Estimation (ADAM) optimizer. ADAM is a popular optimization algorithm known for its efficiency and low memory requirements.

The batch size during training was set to 32. This means that the model was updated after every 32 samples. The number of epochs and the batch size are defined in the code level as shown in Figure 15. To have a more robust learning, the model was made to update more frequently by choosing smaller batch size. The learning rate was set at 0.0001. This is the degree of change in the model for every error predicted when updating weights of the model. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck.



```
history = model.fit(x=x_train, y=y_train, batch_size=32, epochs=50, verbose=1)
```

Figure 15: Definition of batch size and epochs

A Kaggle Notebooks environment was employed to train the model, as it had 29 Gigabytes (GB) of Random Access Memory (RAM). The computational power for training was provided by an Nvidia Tesla P100 GPU that had a memory capacity of 16 GB. This high-performance GPU cut down on the time taken to train the model quite significantly. Regrettably, the TensorFlow version used in implementing this model is 2.15.0

5.4 Extracting and Calculating ECG Leads

To obtain the remaining four ECG leads, the researchers utilize Einthoven's law [21] and Goldberger's equations [22]. Specifically, lead III is derived using Einthoven's law, as indicated in (1).

$$\text{Lead III} = \text{Lead II} - \text{Lead I} \quad (1)$$

Additionally, the augmented leads aVR, aVL, and aVF are calculated using Goldberger's equations, as outlined in (2), (3), and (4) respectively. This comprehensive approach ensures that all necessary ECG leads are accurately reconstructed for subsequent analysis.

$$\text{Lead aVR} = (\text{Lead I} + \text{Lead II}) / (-2) \tag{2}$$

$$\text{Lead aVL} = (\text{Lead I} - \text{Lead III}) / 2 \tag{3}$$

$$\text{Lead aVF} = (\text{Lead II} + \text{Lead III}) / 2 \tag{4}$$

5.5 Deep Learning Model Testing

In this study, various robust metrics were used to assess the quality of reconstructed ECG signals. One of such metrics is Pearson Correlation Coefficient (ρ). This statistical measure evaluates the degree of correlation between actual and reconstructed ECG signals. A large value for Pearson Correlation Coefficient implies great similarity between these two sets meaning that the reconstruction process was successful in preserving fundamental components of the original ECG signals. Calculation of Pearson Correlation Coefficient was made possible by SciPy, a powerful Python library featuring many statistical functions.

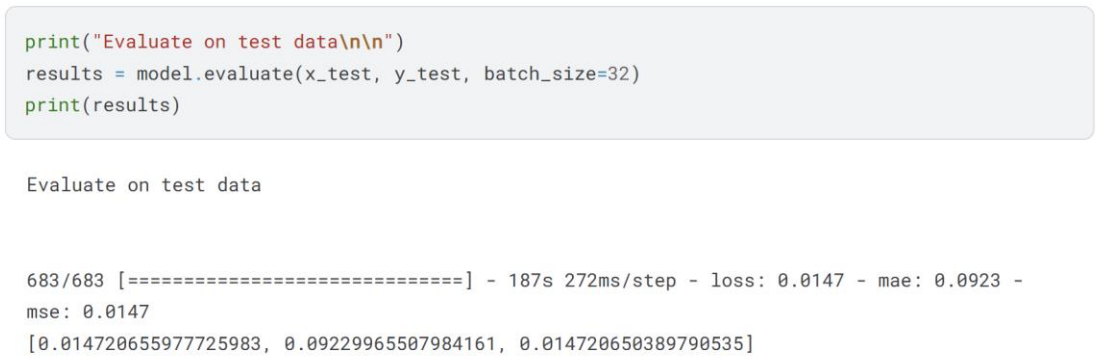
Also, mean squared error (MSE) was utilized as another evaluation metric beside Pearson Correlation Coefficient. MSE gives a measure on average of squared difference between real and reconstructed ECG Signals. Having smaller MSE indicates low error in reconstruction implying that reconstructed signals are close to real ones. This metric is particularly useful for revealing any deviations or inconsistencies in the recovered pulse forms.

At last, I measured the accuracy of the model. In machine learning, this measure of performance is referred to as training accuracy and refers to what proportion of correct predictions made by the model on its training set (to be precise). The high value of training accuracy signifies the fact that the model has learned well from the data and thus shall perform well on unseen data. Reconstructed ECG signals were evaluated thoroughly using all these metrics.

6. RESULTS AND DISCUSSIONS

Garg, Venkataramani and Priyakumar reconstructed all 11 leads using lead II [16], however by calculating leads aVR, aVL and aVF using lead I, lead II and lead III, I have been able to obtain a much higher Pearson's correlation coefficient for majority of reconstructed augmented leads. And as a result, since the deep learning network now has to focus on fewer leads to reconstruct, there is a significant increase in the Pearson's correlation coefficient of the other leads as seen in Table 5.

While the evaluation of the trained model, MSE and mean absolute error (MAE) were taken to measure the error when predicting the values. As illustrated in Figure 16, the trained model achieved a MSE of 0.0147 and a MAE of 0.0923. These error metrics provide insight into the model's accuracy, indicating its ability to closely predict the target values with minimal deviation.



```
print("Evaluate on test data\n\n")
results = model.evaluate(x_test, y_test, batch_size=32)
print(results)
```

```
Evaluate on test data


683/683 [==============================] - 187s 272ms/step - loss: 0.0147 - mae: 0.0923 -
mse: 0.0147
[0.014720655977725983, 0.09229965507984161, 0.014720650389790535]
```

Figure 16: Error metrics of the trained model: MSE and MAE

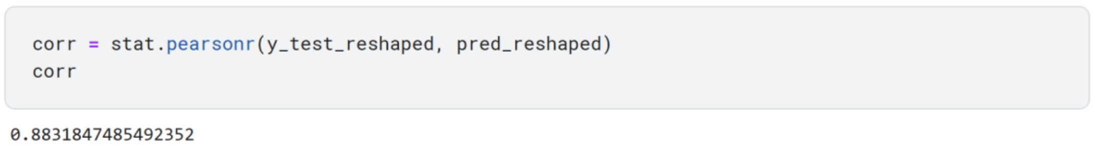
Table 6 presents a clear comparison of the overall results and performance metrics between this study and the study conducted by Garg, Venkataramani and Priyakumar. Evaluation of the Pearson Correlation Coefficient and R^2 value in code level are as shown in Figure 17 and Figure 18 respectively. This comparison demonstrates the improved outcomes achieved by the model of this study.

Table 5: Comparison of metrics between Modified Attention U-Net and proposed model.

ECG Lead	**Modified Attention U-net**	**Proposed Model**
	ρ	ρ
Lead I	0.847	-
Lead II	-	0.917
Lead III	0.663	0.923
Lead aVR	0.659	0.971
Lead aVL	0.935	0.974
Lead aVF	0.892	0.895
Lead V	0.818	0.827
Lead V	0.778	0.921
Lead V	0.728	0.812
Lead V	0.785	0.856
Lead V	0.859	0.911
Lead V	0.888	0.939

Table 6: Comparison of the overall performance metrics.

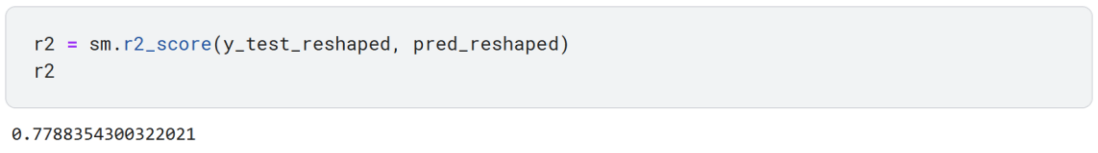
Metric	**Modified Attention U-net**	**Proposed Model**
Overall Pearson Correlation Coefficient (ρ)	0.805	0.883
R^2 Value	0.639	0.779



```
corr = stat.pearsonr(y_test_reshaped, pred_reshaped)
corr
```

```
0.8831847485492352
```

Figure 17: Evaluating Pearson correlation coefficient



```
r2 = sm.r2_score(y_test_reshaped, pred_reshaped)
r2
```

```
0.7788354300322021
```

Figure 18: Evaluating R^2 value

During this research, I have experimented with four more additional models that had structural modifications before reaching the final conclusions. The following are the highlights and results for these models.

- Model 1:
  - Garg, Venkataramani and Priyakumar suggested an architecture for this model. Nonetheless, I minimized the output layer channels from 11 to 7 since we were only interested in reconstructing 7 ECG leads. The number and type of convolutional layers and the number of attention gates in the original architecture remained unchanged. Figure 19 shows a pictorial description of this model architecture.
  - This model achieved Pearson Correlation Coefficient value as high up to 0.8589 (Figure 20) and R² value that is approximately equal to 0.7345 (Figure 21).

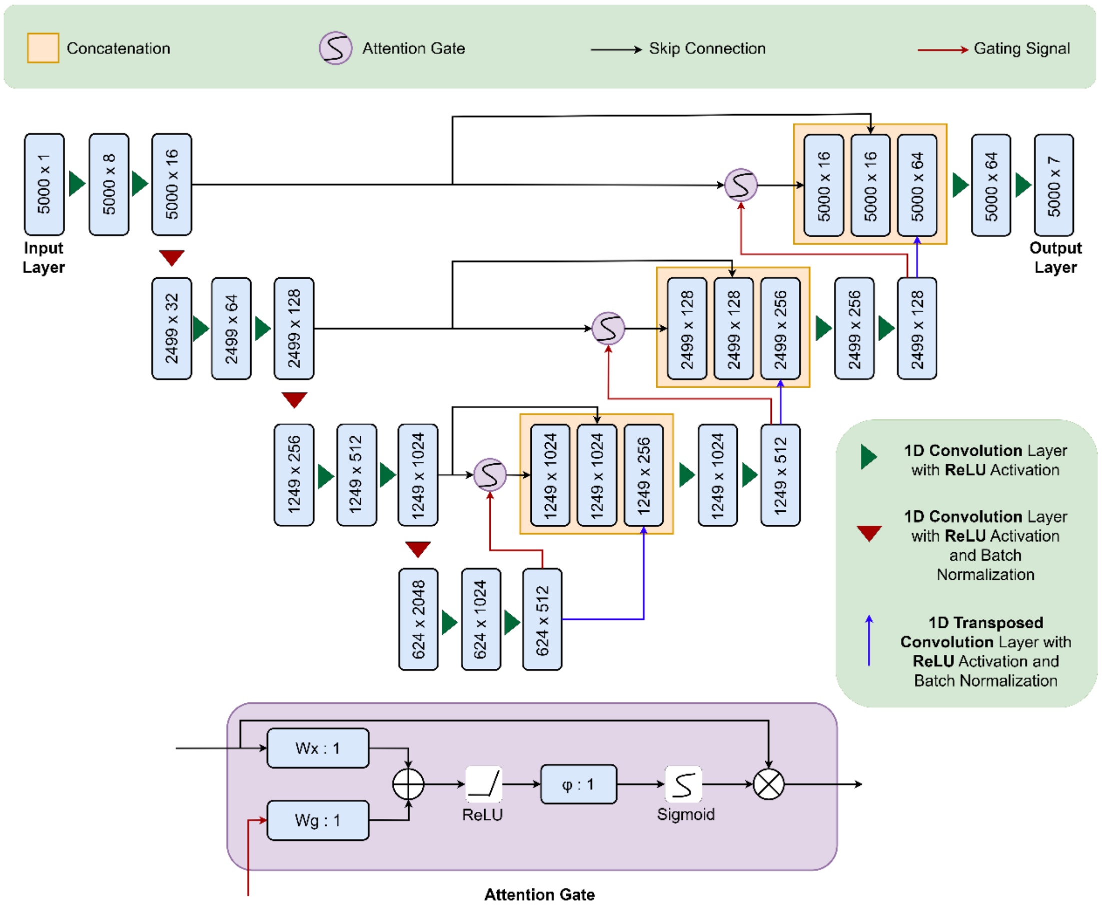
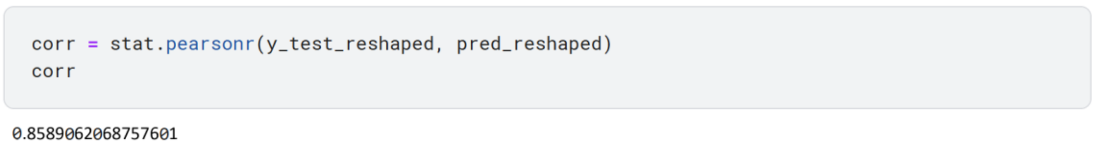


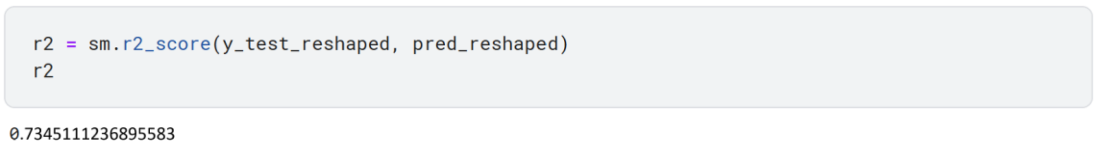
Figure 19: Architecture of model 1



```
corr = stat.pearsonr(y_test_reshaped, pred_reshaped)
corr
```

```
0.8589062068757601
```

Figure 20: Pearson correlation coefficient of model 1



```
r2 = sm.r2_score(y_test_reshaped, pred_reshaped)
r2
```

```
0.7345111236895583
```

Figure 21: R^2 value of model 1

- Model 2:
    - The Garg, Venkataramani and Priyakumar's model also served as the basis for this one. However, in this case, I removed the final set of convolutional layers (Figure 22). The alteration resulted in a decrease of six convolutional layers and one attention gate.
    - According to Figure 23 and Figure 24, the Pearson Correlation Coefficient was found to be 0.7873 while R^2 value was estimated as 0.6197 by means of this model.

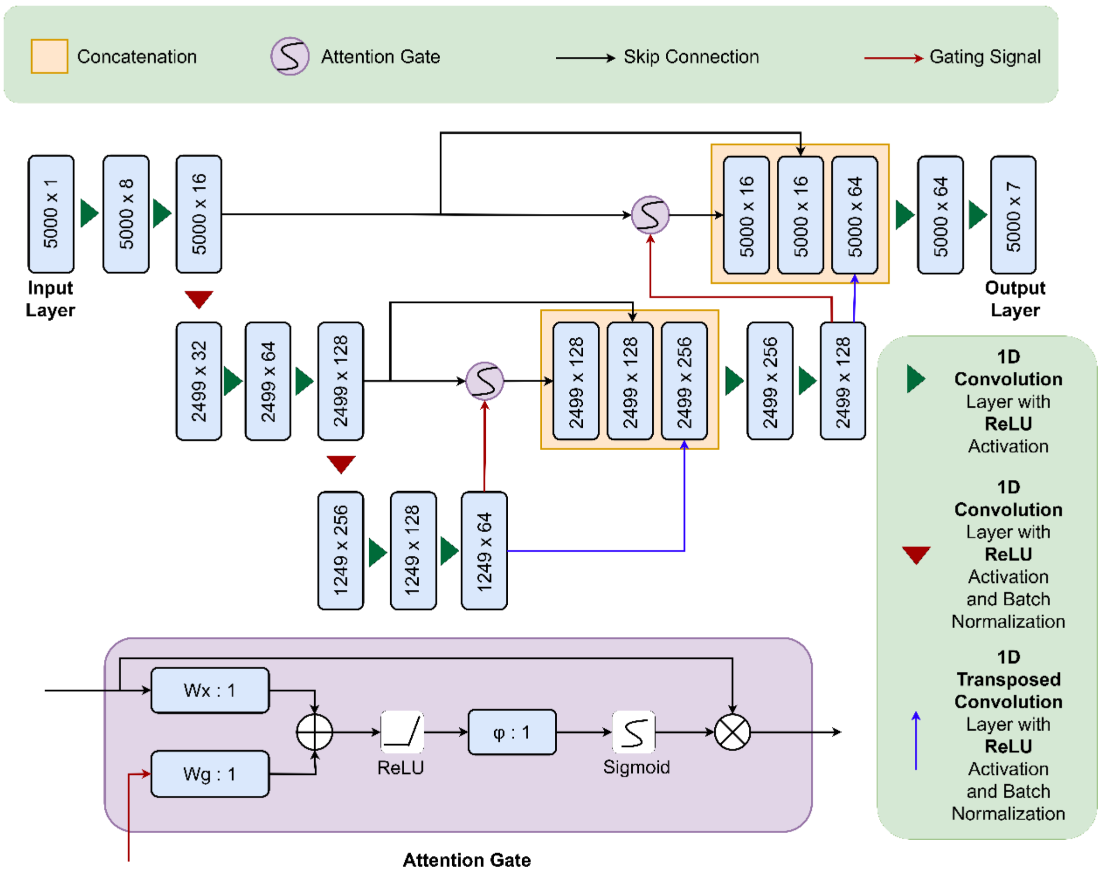
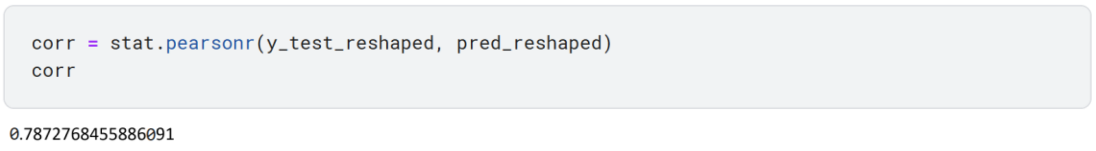


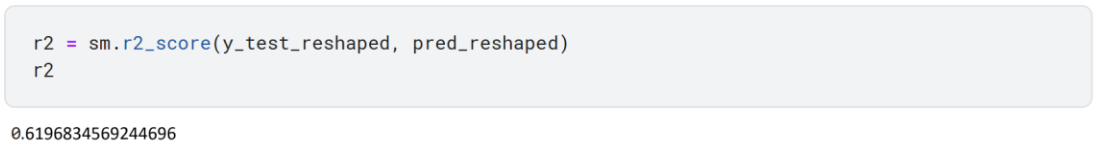
Figure 22: Architecture of model 2



```
corr = stat.pearsonr(y_test_reshaped, pred_reshaped)
corr
```

```
0.7872768455886091
```

Figure 23: Pearson correlation coefficient of model 2



```
r2 = sm.r2_score(y_test_reshaped, pred_reshaped)
r2
```

```
0.6196834569244696
```

Figure 24: R^2 value of model 2

- Model 3 (Proposed Model):
    - The model was initially developed based on Model 1, and it included two more 1D convolutional layers with ReLU activation functions before the output layer. It can be seen in Figure 10.
    - Pearson correlation of this model is found to be 0.8832 while R^2 is equal to 0.7788 as shown in Figure 17 and Figure 18 respectively.

- Model 4:
    - An 1D convolutional layer was added after the input layer to enhance this model, following Model 3. This is illustrated in Figure 25.
    - Figure 26 shows a Pearson correlation coefficient of 0.8652 and an R^2 value of 0.7459 in Figure 27 for this model respectively.

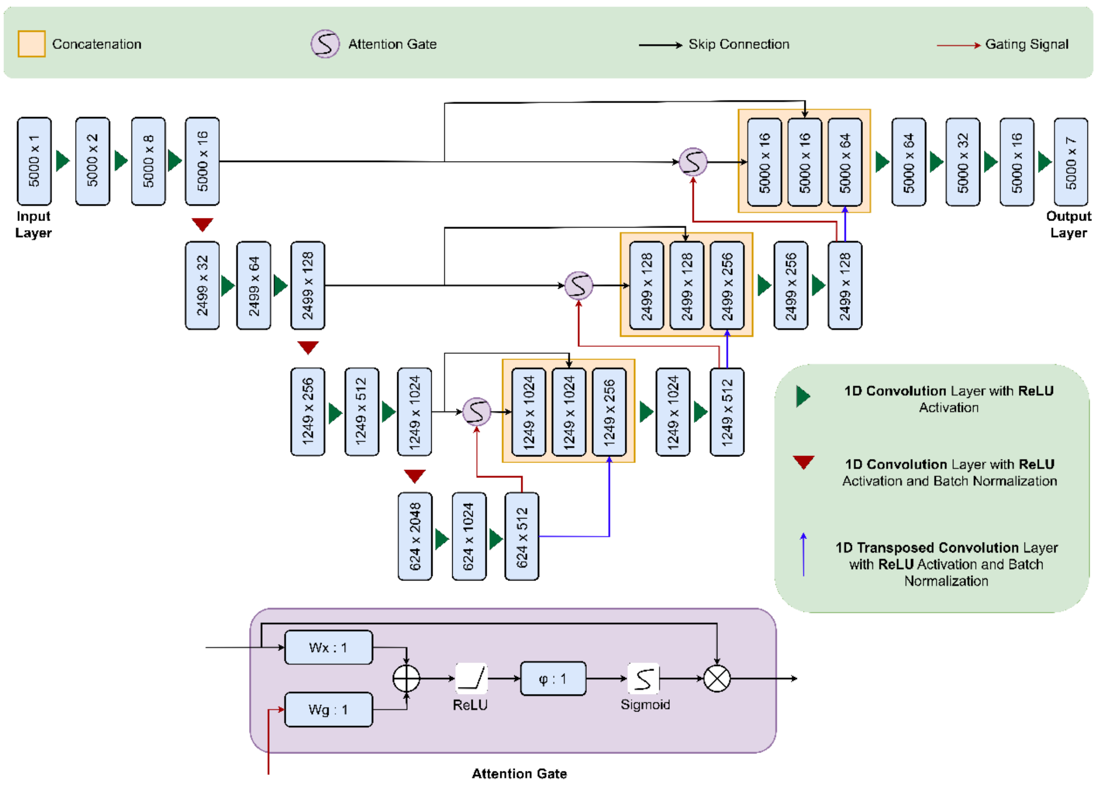
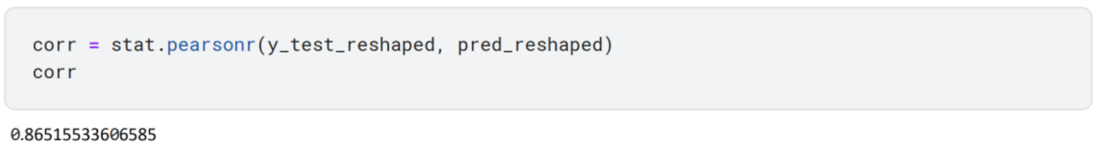


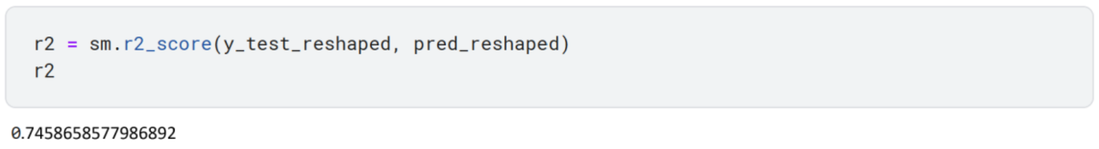
Figure 25: Architecture of model 4



```
corr = stat.pearsonr(y_test_reshaped, pred_reshaped)
corr
```

```
0.86515533606585
```

Figure 26: Pearson correlation coefficient of model 4



```
r2 = sm.r2_score(y_test_reshaped, pred_reshaped)
r2
```

```
0.7458658577986892
```

Figure 27: R^2 value of model 4

- Model 5:
  - At the end of Model 4, one more 1D convolution layer with a ReLU activation function was used after the input layer as shown in Figure 28.
  - The Pearson's correlation coefficient for this model was 0.7282 and an R^2 value of 0.5037 as displayed by Figure 29 and Figure 30 respectively.

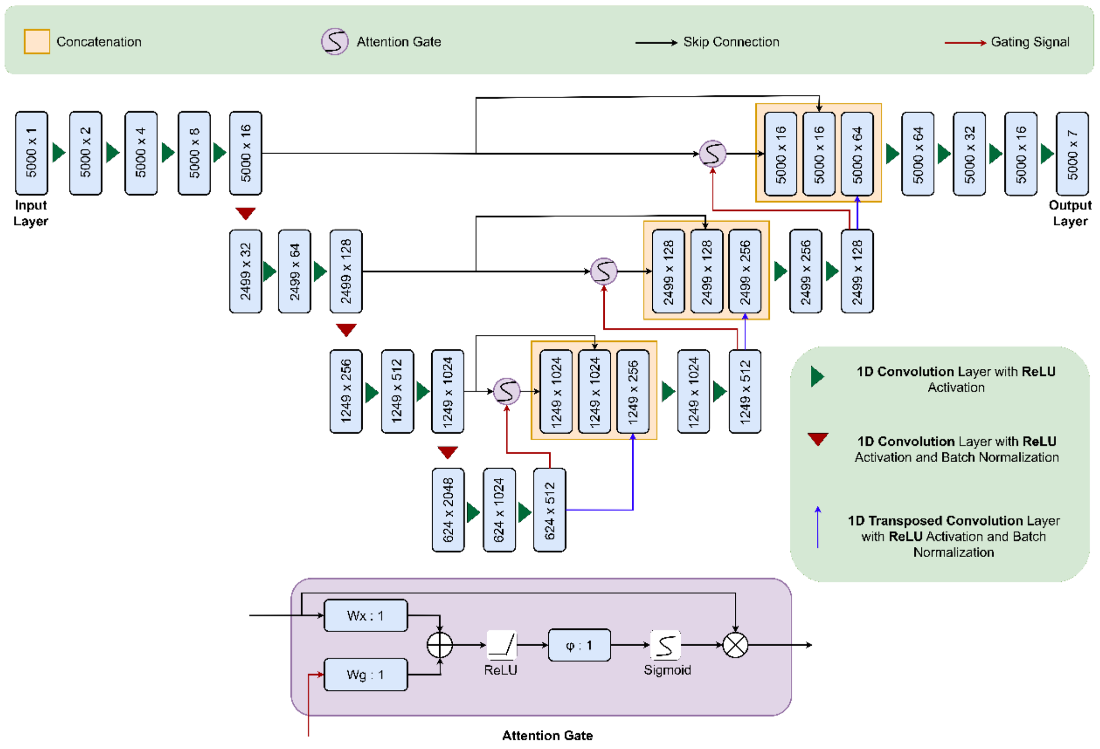
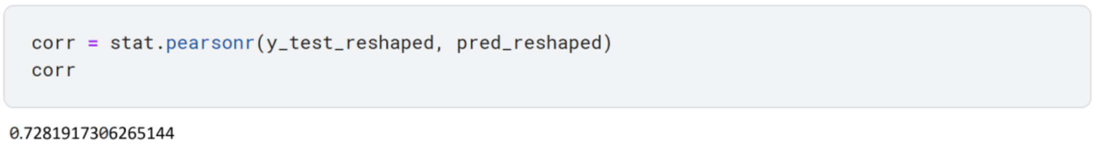


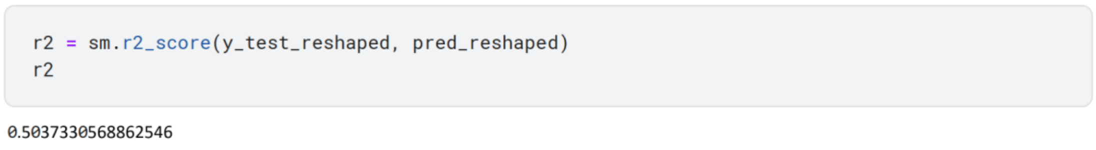
Figure 28: Architecture of model 5



```
corr = stat.pearsonr(y_test_reshaped, pred_reshaped)
corr
```

```
0.7281917306265144
```

Figure 29: Pearson correlation coefficient of model 5



```
r2 = sm.r2_score(y_test_reshaped, pred_reshaped)
r2
```

```
0.5037330568862546
```

Figure 30: R^2 value of Model 5

Table 7 contains the performance metrics summary for all models. Model 3 emerged as the best among them based on a comparison of several indicators. It had both highest Pearson Correlation Coefficient as well as R^2 value to other models.

Table 7: Summary of Performance Metrics Across Different Model Architectures

Model	**Pearson Correlation Coefficient (ρ)**	**R^2 Value**
Model 1	0.8589	0.7345
Model 2	0.7873	0.6197
Model 3 (Proposed Model)	**0.8832**	**0.7788**
Model 4	0.8652	0.7459
Model 5	0.7282	0.5037

7. PRODUCT DEVELOPMENT AND DEPLOYMENT

7.1 Mobile Application Development

This research has resulted in a mobile application that can be used directly by the end user. It is meant to work with Android devices running version 5.0 (Lollipop) and above, targeting specifically minimum software development kit (SDK) version of 21 devices. The application needs 2GB RAM on a device to run effectively as it should do for optimal performance. This compatibility ensures that the application can run efficiently on a wide range of modern Android devices, providing accessibility to a broad user base.

The Flutter framework was preferred in the creation of our mobile application because it produces cross-platform applications which are compatible across both Android and iOS gadgets. In order to make it easier for us during development, we relied upon Android Studio as an integrated development environment (IDE) while coding, debugging and testing because of its robust features for these purposes. We applied software engineering best practices like using version control system through the entire life cycle of development relying on GitHub as our tool for keeping different versions. Consequently, we were able to track changes systematically and maintain consistent updates which had occurred in an effective history of development that ensured reliability and integrity towards such evolution of this product.

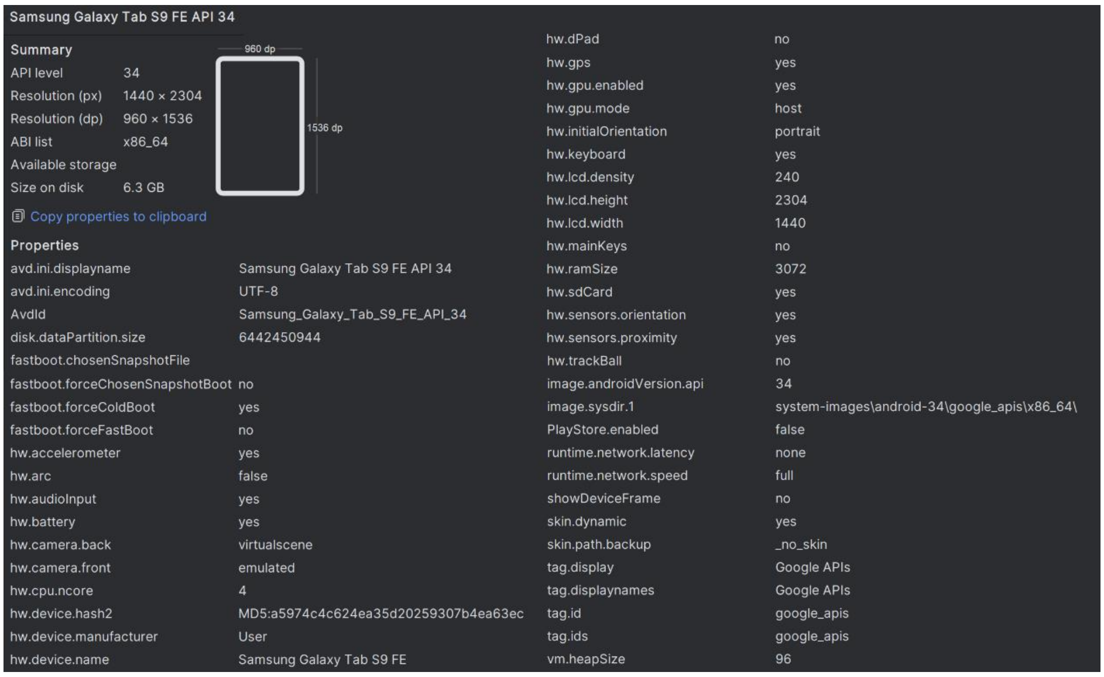
GitHub Repository link: https://github.com/SanuthiVihansa/CardioFit-AI

To ensure user-friendly access and interpretation of cardiac data, it is essential to plot the reconstructed ECG leads within the mobile application. To achieve this efficiently, I selected the `fl_chart` Flutter package [23] as a dependency, enabling precise and responsive plotting of the ECG signals.

The reconstructed ECG signals are stored for future reference, allowing users to view their data at any time. To manage this storage effectively, we utilized Firebase Firestore, a real-time NoSQL database known for its scalability and ease of use. The ECG data are stored as documents within a collection named "ECG" in Firebase. The JSON based format of each document is in the same structure, comprising of email address of the user and data for each ECG lead (lead I, lead II, lead III, lead aVR, lead aVL, lead aVF, lead V1, lead V2, lead V3, lead V4, lead V5 and lead V6), as well as timestamping when the data was recorded. A single document follows this structure:

```
{
    "email": "<user’s email>",
    "l1": [<ECG lead I data>],
    "l2": [<ECG lead II data>],
    "l3": [<ECG lead III data>],
    "avr": [<ECG lead aVR data>],
    "avl": [<ECG lead aVL data>],
    "avf": [<ECG lead aVF data>],
    "v1": [<ECG lead V1 data>],
    "v2": [<ECG lead V2 data>],
    "v3": [<ECG lead V3 data>],
    "v4": [<ECG lead V4 data>],
    "v5": [<ECG lead V5 data>],
    "v6": [<ECG lead V6 data>],
    "datetime": Timestamp Object
}
```

During the development phase of the application, both a physical tablet device and an emulator were employed to run and test the application under various conditions. Specifically, a Samsung Galaxy Tab S9 FE (2023) was utilized as the physical device, providing a real-world testing environment to ensure the application’s performance and responsiveness on actual hardware. In addition to this, a customized emulator was used, configured with the specifications detailed in Figure 31. This emulator allowed for rigorous testing in a controlled environment, simulating a wide range of device conditions and ensuring the application's compatibility across different Android platforms.


Samsung Galaxy Tab S9 FE API 34

Summary

API level	34
Resolution (px)	1440 × 2304
Resolution (dp)	960 × 1536
ABI list	x86_64
Available storage	
Size on disk	6.3 GB

Properties

avd.ini.displayname	Samsung Galaxy Tab S9 FE API 34
avd.ini.encoding	UTF-8
AvdId	Samsung_Galaxy_Tab_S9_FE_API_34
disk.dataPartition.size	6442450944
fastboot.chosenSnapshotFile	
fastboot.forceChosenSnapshotBoot	no
fastboot.forceColdBoot	yes
fastboot.forceFastBoot	no
hw.accelerometer	yes
hw.arc	false
hw.audioInput	yes
hw.battery	yes
hw.camera.back	virtualscene
hw.camera.front	emulated
hw.cpu.ncore	4
hw.device.hash2	MD5:a5974c4c624ea35d20259307b4ea63ec
hw.device.manufacturer	User
hw.device.name	Samsung Galaxy Tab S9 FE
hw.dPad	no
hw.gps	yes
hw.gpu.enabled	yes
hw.gpu.mode	host
hw.initialOrientation	portrait
hw.keyboard	yes
hw.lcd.density	240
hw.lcd.height	2304
hw.lcd.width	1440
hw.mainKeys	no
hw.ramSize	3072
hw.sdCard	yes
hw.sensors.orientation	yes
hw.sensors.proximity	yes
hw.trackBall	no
image.androidVersion.api	34
image.sysdir.1	system-images\android-34\google_apis\x86_64\
PlayStore.enabled	false
runtime.network.latency	none
runtime.network.speed	full
showDeviceFrame	no
skin.dynamic	yes
skin.path.backup	_no_skin
tag.display	Google APIs
tag.displaynames	Google APIs
tag.id	google_apis
tag.ids	google_apis
vm.heapSize	96

Figure 31: Emulator specifications

Figure 32 illustrates the user interface (UI) of the mobile application, which is designed to display the reconstructed ECG leads. The primary feature of this interface is the ECG plot, which visually represents the ECG signals over time, with the duration clearly marked in seconds. The UI also includes a dropdown menu that allows users to select and view different ECG leads. Additionally, the interface provides buttons for navigating to other components of the application, including the heart disease diagnosis module and the home menu.

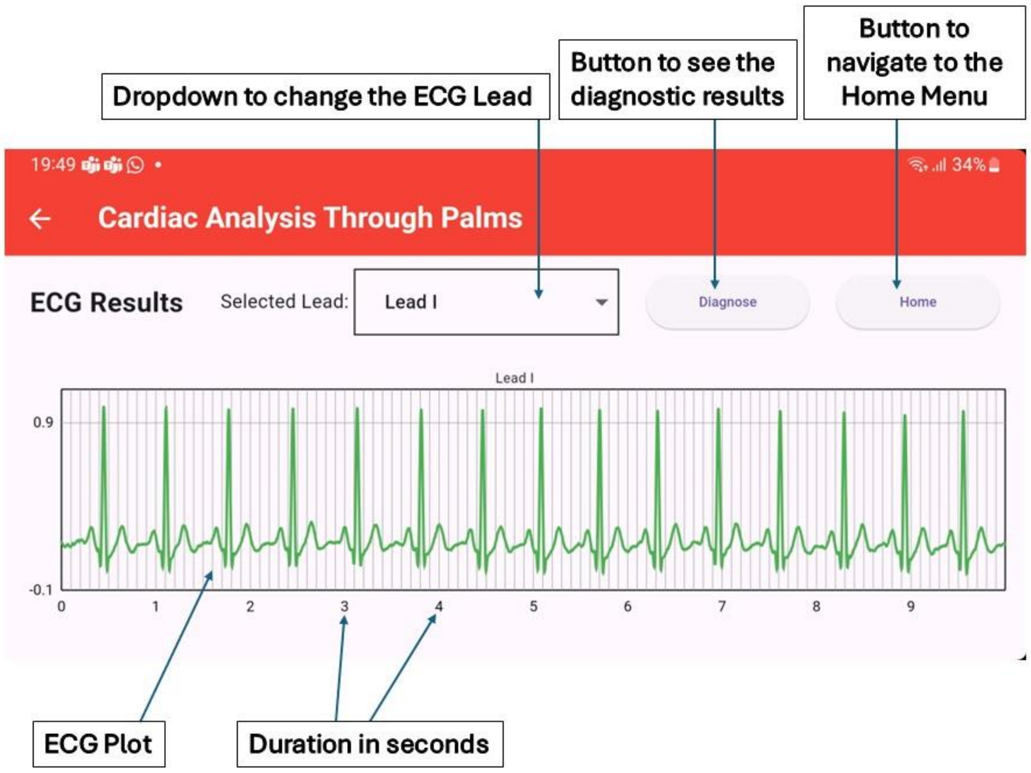

Dropdown to change the ECG Lead
Button to see the diagnostic results
Button to navigate to the Home Menu
19:49
34%
Cardiac Analysis Through Palms
ECG Results
Selected Lead:
Lead I
Diagnose
Home
Lead I
0.9
-0.1
0
1
2
3
4
5
6
7
8
9
ECG Plot
Duration in seconds

Figure 32: ECG reconstruction results screen with lead I

In Figure 32, the ECG plot for lead I is prominently displayed in green. This lead is sourced directly from the user's palms and is not reconstructed, distinguishing it from the other ECG leads. Conversely, as shown in Figure 33, the remaining 11 reconstructed ECG leads are displayed in blue, clearly differentiating them from the directly measured lead I.

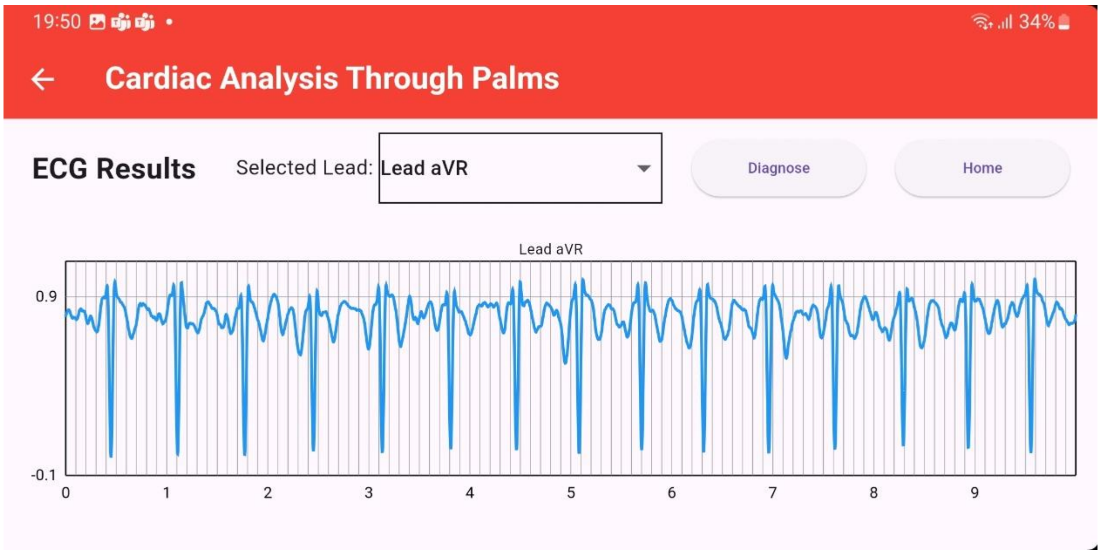


Figure 33: ECG reconstruction results screen with a reconstructed lead

The dropdown menu, as depicted in Figure 34 and Figure 35, offers 12 selectable options corresponding to the different ECG leads. This feature provides users with the flexibility to view specific leads as needed, enhancing the usability and functionality of the application.

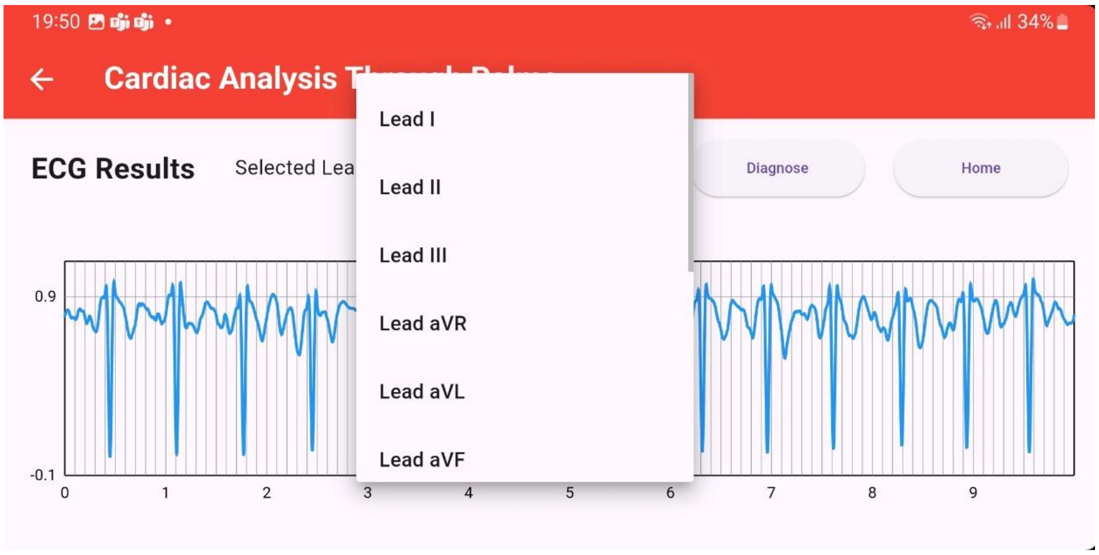


Figure 34: ECG reconstruction results screen with dropdown options - 1

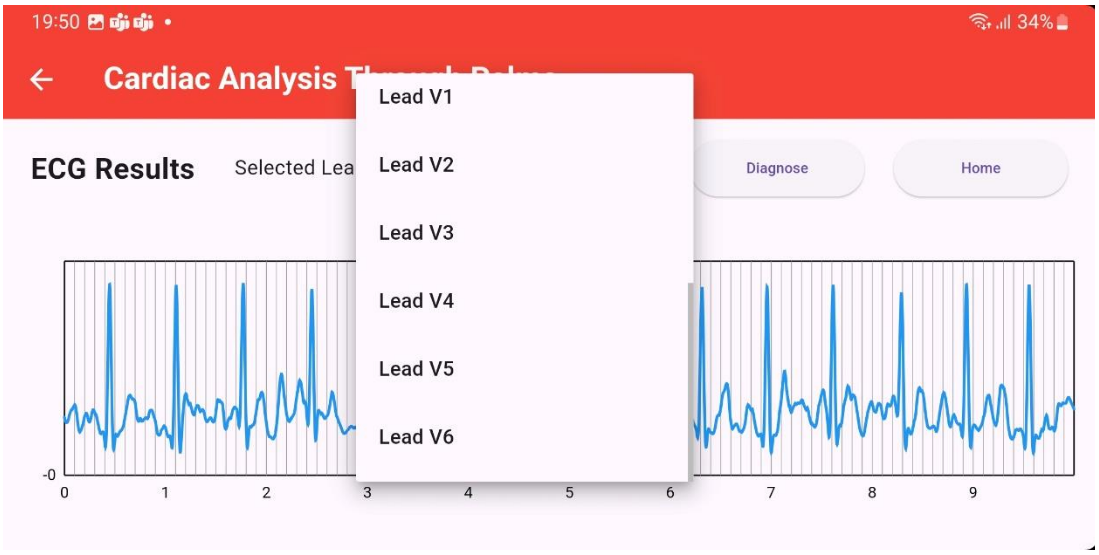


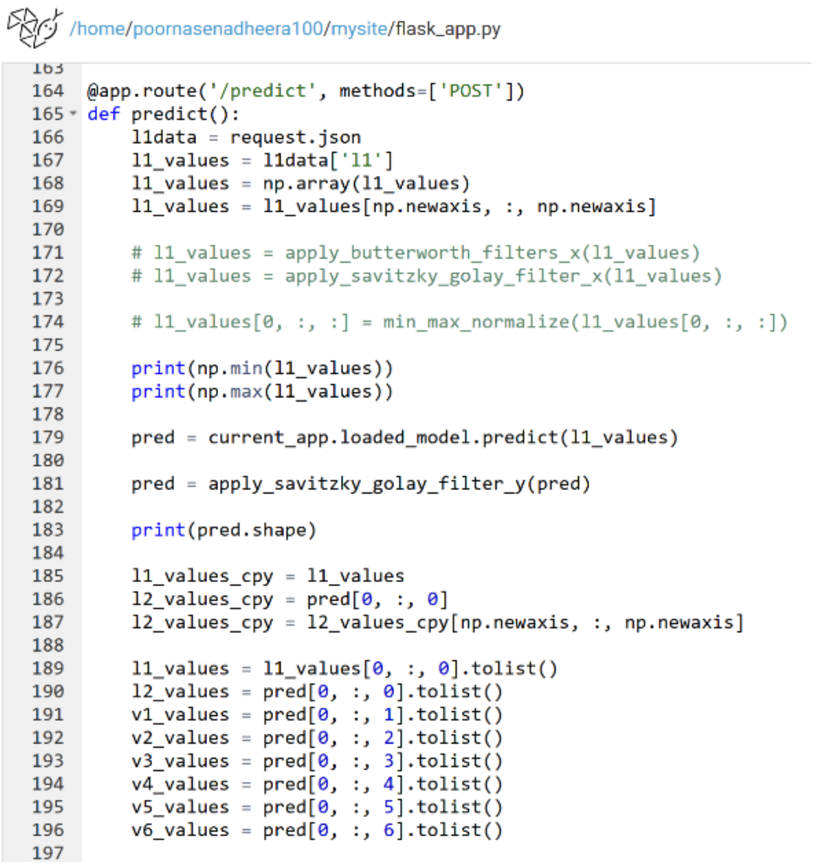
Figure 35: ECG reconstruction results screen with dropdown options - 2

7.2 Flask Server Deployment

Running an application that incorporates a deep learning model on a mobile or tablet device typically demands significant computational resources, particularly in terms of RAM and storage. This requirement can limit the number of devices capable of effectively running the application. To address this challenge and ensure broader accessibility, I opted to deploy the deep learning model responsible for reconstructing multiple ECG leads within a cloud environment.

The PythonAnywhere platform (https://www.pythonanywhere.com) was selected as our cloud solution, leveraging its robust infrastructure to host the deep learning model. The model was deployed using a Flask server on this platform, enabling efficient handling of computations and data processing that would otherwise strain mobile devices. To facilitate seamless communication between the Flutter application and the cloud-based model, an application programming interface (API) endpoint was established, as illustrated in Figure 36. This endpoint allows the mobile application to send requests and receive predictions via the hypertext transfer protocol (HTTP) protocol.

API endpoint: http://poornasenadheera100.pythonanywhere.com/predict


/home/poornasenadheera100/mysite/flask_app.py

```python
@app.route('/predict', methods=['POST'])
def predict():
    l1data = request.json
    l1_values = l1data['l1']
    l1_values = np.array(l1_values)
    l1_values = l1_values[np.newaxis, :, np.newaxis]

    # l1_values = apply_butterworth_filters_x(l1_values)
    # l1_values = apply_savitzky_golay_filter_x(l1_values)

    # l1_values[0, :, :] = min_max_normalize(l1_values[0, :, :])

    print(np.min(l1_values))
    print(np.max(l1_values))

    pred = current_app.loaded_model.predict(l1_values)

    pred = apply_savitzky_golay_filter_y(pred)

    print(pred.shape)

    l1_values_cpy = l1_values
    l2_values_cpy = pred[0, :, 0]
    l2_values_cpy = l2_values_cpy[np.newaxis, :, np.newaxis]

    l1_values = l1_values[0, :, 0].tolist()
    l2_values = pred[0, :, 0].tolist()
    v1_values = pred[0, :, 1].tolist()
    v2_values = pred[0, :, 2].tolist()
    v3_values = pred[0, :, 3].tolist()
    v4_values = pred[0, :, 4].tolist()
    v5_values = pred[0, :, 5].tolist()
    v6_values = pred[0, :, 6].tolist()
```

Figure 36: API endpoint configuration in Flask server

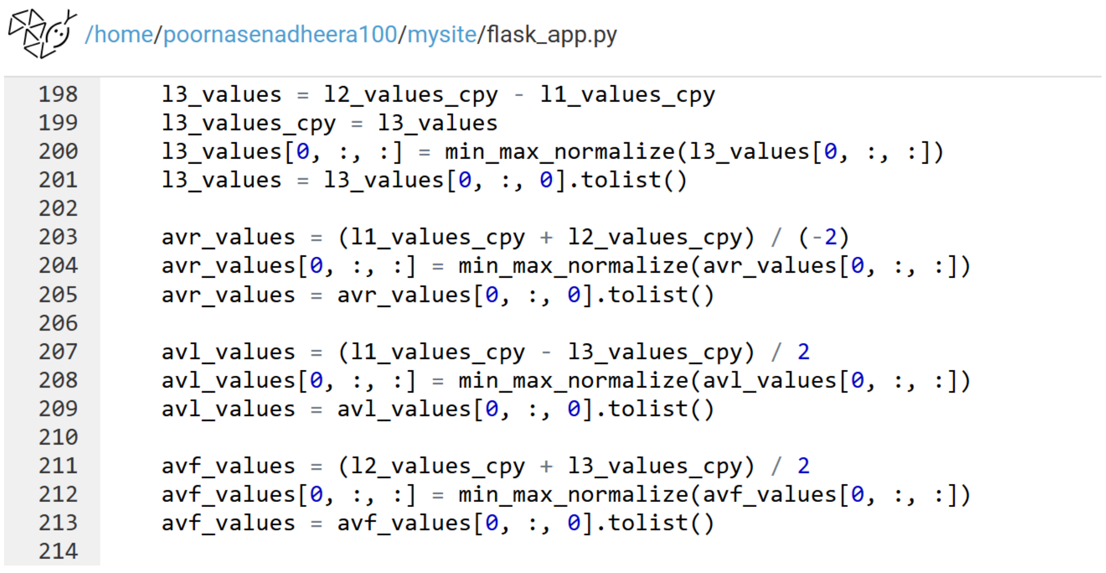
The HTTP route was defined as a POST request, designed to handle the transmission of filtered and smoothed ECG data for lead I. The request accepts this data in the body of the request, formatted in JSON. The JSON structure is specifically crafted to encapsulate the lead I ECG data, ensuring that the deep learning model receives the necessary input for accurate reconstruction of the additional ECG leads. The structure of the JSON body is as follows:

{

"l1": [<ECG lead I data>]

}

Upon receiving the input data, the Flask server initiates a two-step process to reconstruct and calculate the required ECG leads. The first step is the reconstruction by the server of ECG leads II, V1, V2, V3, V4, V5 and V6 using the given lead I data. Consequently, in the second step, other leads: lead III, lead aVR, lead aVL and lead aVF are computed through (1), (2), (3) and (4). The code-level implementation of these calculations within the Flask server is depicted in Figure 37.


/home/poornasenadheera100/mysite/flask_app.py

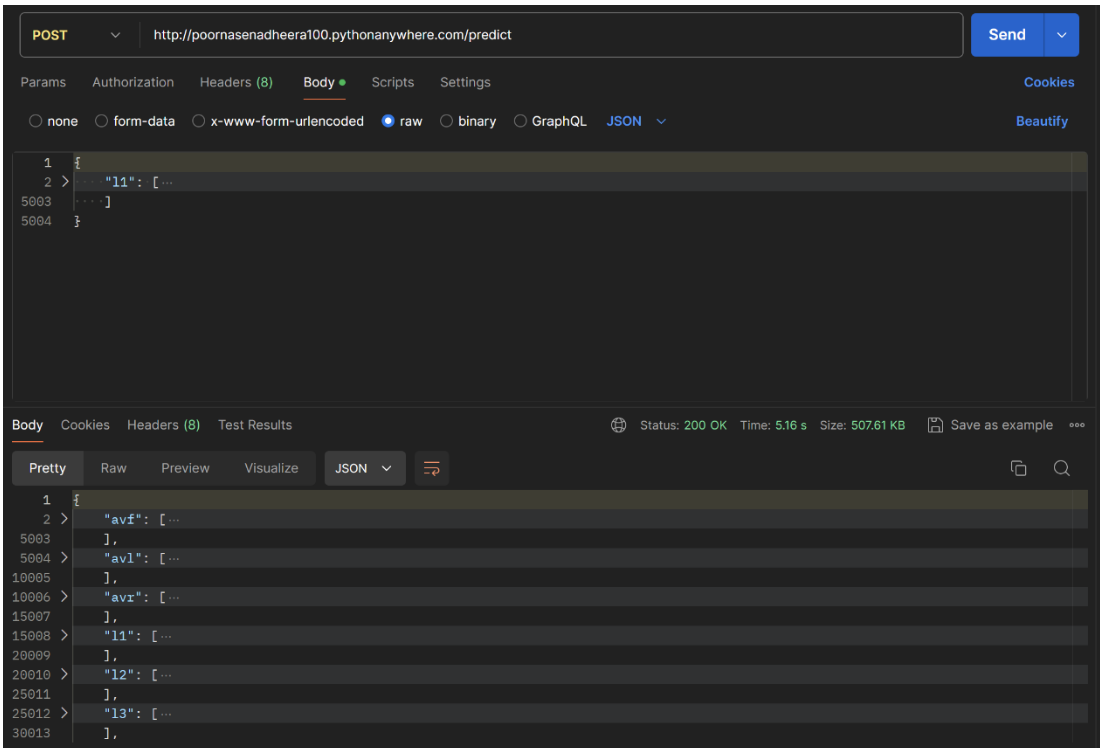
```
    l3_values = l2_values_cpy - l1_values_cpy
    l3_values_cpy = l3_values
    l3_values[0, :, :] = min_max_normalize(l3_values[0, :, :])
    l3_values = l3_values[0, :, 0].tolist()

    avr_values = (l1_values_cpy + l2_values_cpy) / (-2)
    avr_values[0, :, :] = min_max_normalize(avr_values[0, :, :])
    avr_values = avr_values[0, :, 0].tolist()

    avl_values = (l1_values_cpy - l3_values_cpy) / 2
    avl_values[0, :, :] = min_max_normalize(avl_values[0, :, :])
    avl_values = avl_values[0, :, 0].tolist()

    avf_values = (l2_values_cpy + l3_values_cpy) / 2
    avf_values[0, :, :] = min_max_normalize(avf_values[0, :, :])
    avf_values = avf_values[0, :, 0].tolist()

```

Figure 37: Code implementation for calculation of lead III, aVR, aVL and aVF

Once the reconstruction and calculations are complete, the server processes the output and prepares the response data in JSON format. This response is then sent back to the Flutter application, enabling the application to display the full set of ECG leads to the user. The structure of the response data in JSON format is as follows:

```
{
    "avf": [<ECG lead aVF data>],
    "avl": [<ECG lead aVL data>],
    "avr": [<ECG lead aVR data>],
    "l1": [<ECG lead I data>],
    "l2": [<ECG lead II data>],
    "l3": [<ECG lead III data>],
    "v1": [<ECG lead V1 data>],
    "v2": [<ECG lead V2 data>],
    "v3": [<ECG lead V3 data>],
    "v4": [<ECG lead V4 data>],
    "v5": [<ECG lead V5 data>],
    "v6": [<ECG lead V6 data>]
}
```

Figure 38 and Figure 39 demonstrate the testing of this endpoint using the Postman tool.


POST http://poornasenadheera100.pythonanywhere.com/predict

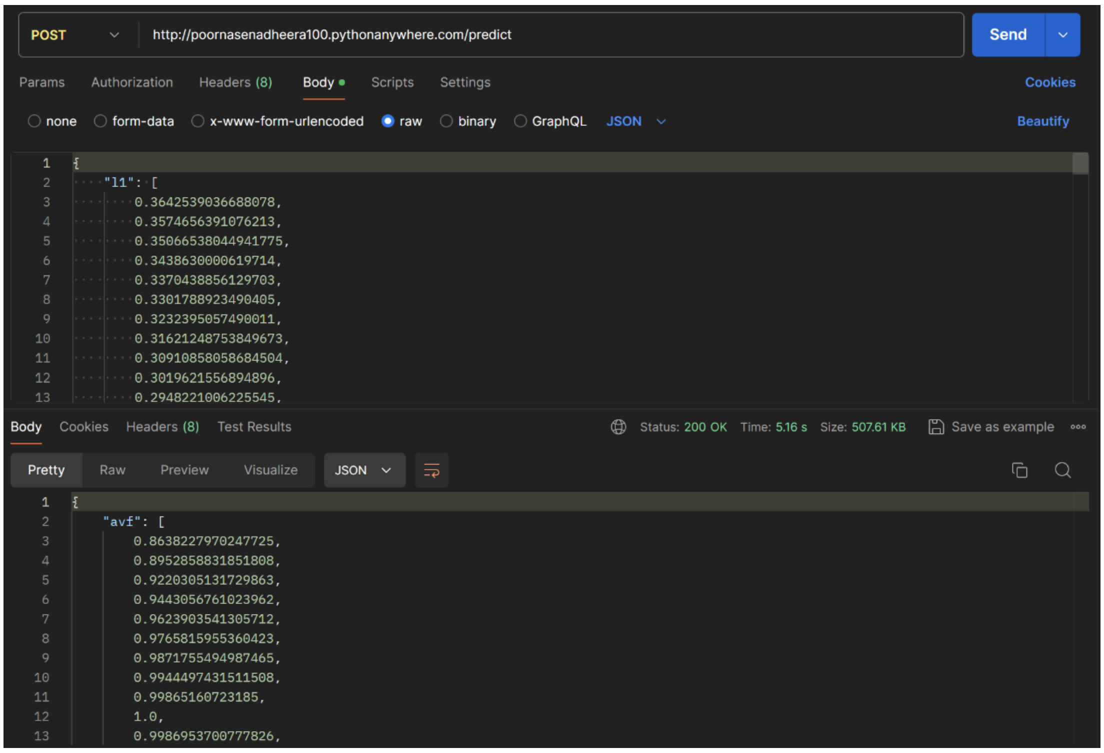
Body: raw, JSON

```
{
    "l1": [⋯
    ]
}
```

Status: 200 OK Time: 5.16 s Size: 507.61 KB

```
{
    "avf": [⋯
    ],
    "avl": [⋯
    ],
    "avr": [⋯
    ],
    "l1": [⋯
    ],
    "l2": [⋯
    ],
    "l3": [⋯
    ],
```

Figure 38: Postman test of HTTP endpoint with JSON request and response - 1


POST http://poornasenadheera100.pythonanywhere.com/predict

Body: raw, JSON

```
{
    "l1": [
        0.3642539036688078,
        0.357465639107621З,
        0.35066538044941775,
        0.3438630000619714,
        0.3370438856129703,
        0.3301788923490405,
        0.3232395057490011,
        0.31621248753849673,
        0.30910858058684504,
        0.3019621556894896,
        0.294822100622554 5,
```

Status: 200 OK Time: 5.16 s Size: 507.61 KB

```
{
    "avf": [
        0.8638227970247725,
        0.8952858831851808,
        0.922030513172986З,
        0.9443056761023962,
        0.9623903541305712,
        0.9765815955360423,
        0.9871755494987465,
        0.9944497431511508,
        0.99865160723185,
        1.0,
        0.9986953700777826,
```

Figure 39: Postman test of HTTP endpoint with JSON request and response – 2

8. CONCLUSION

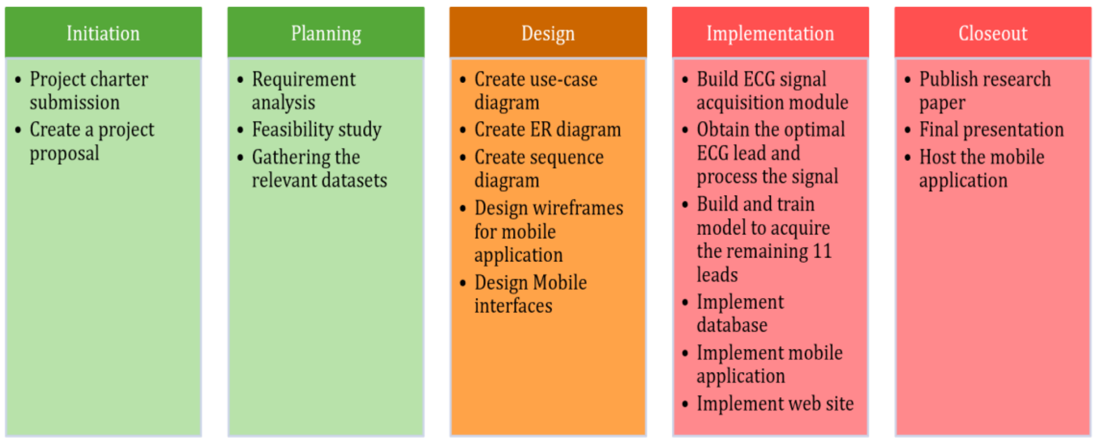
The component focused on "Reconstructing Multiple ECG Leads from a Single Lead" has demonstrated significant potential in revolutionizing cardiac monitoring, particularly in environments where conventional 12-lead ECG setups are impractical. By leveraging a deep learning approach, specifically the Modified Attention U-Net framework, the research has successfully reconstructed a full set of 12-lead ECG signals using only a single lead, predominantly lead I. This achievement not only simplifies the hardware requirements but also maintains a high level of accuracy, as evidenced by the strong Pearson correlation coefficients and R^2 values obtained during the testing phase.

Reconstructing process of extracting lead I from a personalized ambulatory ECG device and using deep learning to generate the rest of the leads is a way to deal with key difficulties in nowadays ECG acquisition methods. Many conventional ways have demanded several electrodes placed at specific anatomical points which can be inconvenient and uncomfortable for remote monitoring. On the other hand, the proposed solution provides an alternative that is easier to use but does not compromise on the diagnostic quality of ECG signals.

In practical sense, this technology has potential game-changing capabilities in remote health monitoring systems by enabling real-time accurate cardiac assessments with minimal equipment. This would be very important in cases of emergency or outpatient departments without necessarily having the burden of traditional ECG setups. Besides, commercial prospects for this technology are good given that there is increasing demand for telemedicine and portable diagnostic tools.

Research in the field of biomedical engineering becomes significantly enriched by this research, as it establishes the possibility of building 12-lead ECGs from single leads. Subsequent work could be aimed at making the model more perfect for even higher accuracy, examining its compatibility with current telemedicine platforms and applying it to other forms of physiological signals. This system has been successfully implemented and tested, thus opening up avenues for improved cardiac monitoring solutions that are accessible and efficient within the healthcare industry.

9. WORK BREAKDOWN STRUCTURE (WBS)



Initiation
Project charter submission
Create a project proposal
Planning
Requirement analysis
Feasibility study
Gathering the relevant datasets
Design
Create use-case diagram
Create ER diagram
Create sequence diagram
Design wireframes for mobile application
Design Mobile interfaces
Implementation
Build ECG signal acquisition module
Obtain the optimal ECG lead and process the signal
Build and train model to acquire the remaining 11 leads
Implement database
Implement mobile application
Implement web site
Closeout
Publish research paper
Final presentation
Host the mobile application

Figure 40: Work breakdown chart of the component

REFERENCES

[1] “Topic: Diseases,” Statista, Jan. 10, 2024. [Online]. Available: https://www.statista.com/topics/2070/diseases/#topicOverview. [Accessed: 26.02.2024].

[2] “ECG test,” Better Health Channel. [Online]. Available: https://www.betterhealth.vic.gov.au/health/conditionsandtreatments/ecg-test#types-of-ecg. [Accessed: 26.02.2024].

[3] A. Rawshani and A. Rawshani, “The ECG leads: Electrodes, limb leads, chest (precordial) leads and the 12-Lead ECG,” Cardiovascular Education, Jun. 25, 2023. [Online]. Available: https://ecgwaves.com/topic/ekg-ecg-leads-electrodes-systems-limb-chest-precordial/. [Accessed: 26.02.2024].

[4] L. Potter, “Understanding an ECG | ECG Interpretation | Geeky Medics,” Geeky Medics, Nov. 24, 2023. [Online]. Available: https://geekymedics.com/understanding-an-ecg/. [Accessed: 26.02.2024].

[5] “12-lead Electrocardiogram (EKG) to Diagnose Cardiac Arrhythmias.” [Online]. Available: https://www.washingtonhra.com/ekg-monitoring/12-lead-electrocardiogram-ekg.php. [Accessed: 26.02.2024].

[6] Thejackattack, “EKG Help: 3 vs 5 vs 12 lead,” allnurses, May 24, 2023. [Online]. Available: https://allnurses.com/ekg-help-vs-vs-lead-t283901/. [Accessed: 26.02.2024].

[7] CardiacDirect, “12-Lead ECG Placement Guide - CardiacDirect,” CardiacDirect, Sep. 29, 2023. [Online]. Available: https://www.cardiacdirect.com/12-lead-ecg-placement-guide/

[8] Zhu H, Pan Y, Cheng KT, Huan R. A lightweight piecewise linear synthesis method for standard 12-lead ECG signals based on adaptive region segmentation. PloS one. 2018;13(10):e0206170.

[9] Wang Ld, Zhou W, Xing Y, Liu N, Movahedipour M, Zhou Xg. A novel method based on convolutional neural networks for deriving standard 12-lead ECG from serial 3-lead ECG. Frontiers of Information Technology & Electronic Engineering. 2019;20(3):405-13.

[10] Kapfo A, Datta S, Dandapat S, Bora PK. LSTM based Synthesis of 12-lead ECG Signal from a Reduced Lead Set. In: 2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES). vol. 1. IEEE; 2022. p. 296-301.

[11] Sohn J, Yang S, Lee J, Ku Y, Kim HC. Reconstruction of 12-lead electrocardiogram from a three-lead patch-type device using a LSTM network. Sensors. 2020;20(11):3278.

[12] Gundlapalle V, Acharyya A. A Novel Single Lead to 12-Lead ECG Reconstruction Methodology Using Convolutional Neural Networks and LSTM. In: 2022 IEEE 13th Latin America Symposium on Circuits and System (LASCAS). IEEE; 2022. p. 01-4

[13] Yoon GW, Joo S, Kyungmin C, Seo HC. Chest-Lead Generation with Single-Lead;. 2022 Computing in Cardiology. Available from: https://cinc.org/2022/Program/accepted/75 Preprint.pdf.

[14] Bruyne MCd, Kors JA, Hoes AW, Klootwijk P, Dekker JM, Hofman A, et al. Both decreased and increased heart rate variability on the standard 10-second electrocardiogram predict cardiac mortality in the elderly: the Rotterdam Study. American journal of epidemiology. 1999;150(12):1282-8.

[15] Xu K, Li C, Zhu J, Zhang B. Understanding and stabilizing GANs' training dynamics using control theory. In: International Conference on Machine Learning. PMLR; 2020. p. 10566-75

[16] A. Garg, V. V. Venkataramani and U. D. Priyakumar, "Single-Lead to Multi-Lead Electrocardiogram Reconstruction Using a Modified Attention U-Net Framework," 2023 International Joint Conference on Neural Networks (IJCNN), Gold Coast, Australia, 2023, pp. 1-8, doi: 10.1109/IJCNN54540.2023.10191213.

[17] “PTB-XL, a large publicly available electrocardiography dataset v1.0.3,” Nov. 09, 2022. [Online]. Available: https://physionet.org/content/ptb-xl/1.0.3/. [Accessed: 27.02.2024].

[18] M. Cadogan and M. Cadogan, “ECG Lead positioning,” Life in the Fast Lane LITFL, Jan. 30, 2022. [Online]. Available: https://litfl.com/ecg-lead-positioning/

[19] Daud S, Sudirman R. Butterworth bandpass and stationary wavelet transform filter comparison for electroencephalography signal. In: 2015 6th International Conference on Intelligent Systems, Modelling and Simulation. IEEE; 2015. p. 123-6.

[20] Schafer RW. What is a Savitzky-Golay filter?[lecture notes]. IEEE Signal processing magazine. 2011;28(4):111-7.

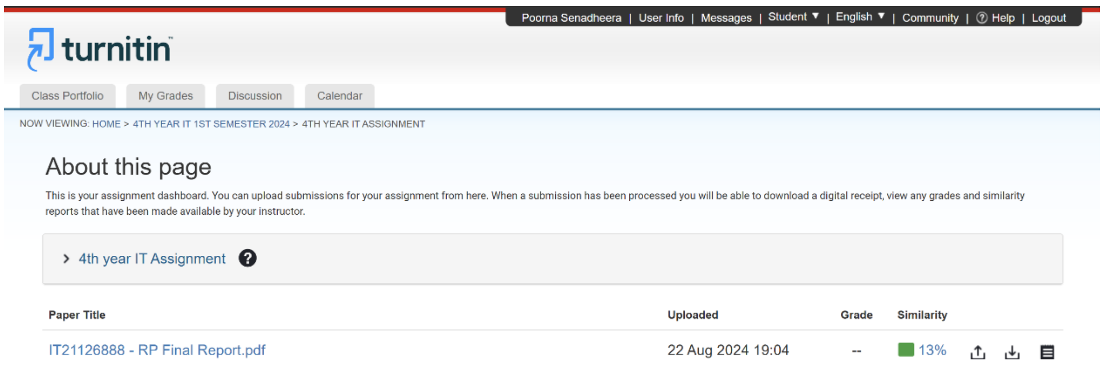
[21] P. J. Zhao, “Einthoven’s Triangle Revisited: A Mathematical Proof,” arXiv (Cornell University), Jan. 2022, doi: 10.48550/arxiv.2205.06772. Available: https://arxiv.org/abs/2205.06772

[22] A. L. Goldberger, Z. D. Goldberger, and A. Shvilkin, *Goldberger's Clinical Electrocardiography: A Simplified Approach*, 10th ed. Philadelphia, PA: Elsevier, 2017.

[23] “fl_chart | Flutter package,” Dart Packages. Available: https://pub.dev/packages/fl_chart

APPENDICES

Appendix A: Plagiarism report



Poorna Senadheera | User Info | Messages | Student ▼ | English ▼ | Community | ⓘ Help | Logout

turnitin

Class Portfolio | My Grades | Discussion | Calendar

NOW VIEWING: HOME > 4TH YEAR IT 1ST SEMESTER 2024 > 4TH YEAR IT ASSIGNMENT

About this page

This is your assignment dashboard. You can upload submissions for your assignment from here. When a submission has been processed you will be able to download a digital receipt, view any grades and similarity reports that have been made available by your instructor.

› 4th year IT Assignment

Paper Title	Uploaded	Grade	Similarity
IT21126888 - RP Final Report.pdf	22 Aug 2024 19:04	--	🟩 13%