CLASSIFICATION OF ARRHYTHMIA BY USING DEEP LEARNING WITH 2-D ECG SPECTRAL IMAGE REPRESENTATION

DSN4097 INTERNSHIP REPORT

Submitted by

Nimish Sarathe (20BCE10881)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRIKALAN, SEHORE
MADHYA PRADESH - 466114

MAY, 2024

VIT BHOPAL UNIVERSITY, KOTHRIKALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation"** is the bonafide work of "**NIMISH SARATHE. (Register No :20BCE10881)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr. J Manikandan
Program Chair, CSE-Core, SCSE
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Mr. Ashfaq Ahmad Najar,
School of Computing Science and Engineering
VIT BHOPAL UNIVERSITY

The Internship Examination is held on _____

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr. J Manikandan, Program Chair, CSE-Core, SCSE for much of his valuable support encouragement in carrying out this work.

I would like to thank my internal guide Mr. Ashfaq Ahmad Najar, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Aeronautical Science, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

LIST OF ABBREVIATIONS

ANN - Artificial Neural Network

ANNC -Adaptive Neural Network Classifier

ATI- Attention-based Time-Incremental

CNN - Convolutional Neural Network

APB - Atrial Premature Beat

AUC - Area Under the ROC Curve

AV - Atrioventricular Node

CAD - Computer-Aided Diagnosis

CNN - Convolutional Neural Network

DL - Deep Learning

DNN - Deep Neural Network

ECG - Electrocardiogram

FAV - First Degree AV Block

FN - False Negative

FP - False Positive

IR - Imbalance Ratio

LBBB -Left Bundle Branch Block

LDA - Linear Discriminant Analysis

LSTM - Long Short-Term Memory

LIST OF FIGURES AND GRAPHS

FIGURE NO.	TITLE	PAGE NO.
1.	Block Diagram	30
2.	Illustrating different Arrhythmias	24 - 25
2.	Home Page	30
3.	Input Page	31
4.	Result Page	32
5	Normal Result Page	33

LIST OF TABLES

FIGURE NO.	TITLE	PAGE NO.
1.	Table of Contents	

ABSTRACT

Purpose:

This project aims to develop an innovative system for the classification of arrhythmias using deep learning techniques applied to 2-D ECG spectral image representation. The purpose of this endeavor is to address the growing need for accurate and efficient arrhythmia diagnosis, thereby improving patient outcomes in cardiovascular healthcare.

Methodology:

The methodology employed involves the utilization of Convolutional Neural Networks (CNNs) to analyze 2-D ECG spectral images and classify arrhythmias into different categories. Through extensive research and experimentation, a robust model is developed and trained on a diverse dataset containing various arrhythmia types.

Findings:

Findings from the project reveal promising results in arrhythmia classification accuracy, with the CNN model demonstrating high performance in distinguishing between different arrhythmia patterns. Additionally, the developed system showcases user-friendly features and seamless integration with web-based platforms, enhancing accessibility and usability for healthcare professionals.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	List of Abbreviations	5
	List of Figures and Graphs	6
	List of Tables	7
	Abstract	8
1	**CHAPTER-1:** **PROJECT DESCRIPTION AND OUTLINE** 1.1 Introduction 1.2 Motivation for the work 1.3 Problem Statement 1.4 Objective of the work 1.5 Organization of the project	17 - 18
2	**CHAPTER-2:** **RELATED WORK INVESTIGATION** 2.1 Introduction 2.2 Existing Approaches/Methods 2.3 Pros and cons of the stated Approaches/Methods 2.4 Issues/observations from investigation	20 - 26
3	**CHAPTER-3:** **REQUIREMENT ARTIFACTS** 3.1 Introduction 3.2 Hardware and Software requirements 3.3 Specific Project requirements 3.3.1 Data requirement 3.3.2 Functions requirement 3.3.3 Performance and security requirement	26 - 27

	3.3.4 Look and Feel Requirements	
4	**CHAPTER-4:** **DESIGN METHODOLOGY AND ITS NOVELTY** 4.1 Methodology and goal 4.2 Functional modules design and analysis 4.3 Software Architectural designs 4.4 Subsystem services 4.5 User Interface designs	28 - 33
5	**CHAPTER-5:** **TECHNICAL IMPLEMENTATION & ANALYSIS** 5.1 Technical coding and code solutions 5.2 Prototype submission	34 - 42
6	**CHAPTER-6:** **PROJECT OUTCOME AND APPLICABILITY** 6.1 Outline 6.2 key implementations outlines of the System 6.3 Significant project outcomes 6.4 Project applicability on Real-world applications 6.4 Inference	43 - 45
7	**CHAPTER-7:** **CONCLUSIONS AND RECOMMENDATION** 7.1 Outline 7.2 Limitation/Constraints of the System 7.3 Future Enhancements 7.4 Inference	46 - 47
	References	48

RELATED WORK INVESTIGATION

The investigation into related work encompassed a comprehensive review of literature, academic papers, and industry reports on arrhythmia classification and digital healthcare solutions. A significant portion of the research focused on understanding the methodologies and advancements in arrhythmia detection. This involved a detailed analysis of recent studies utilizing machine learning techniques, particularly deep learning algorithms such as Convolutional Neural Networks (CNNs). By evaluating the performance and effectiveness of different approaches, valuable insights were gained into the current state-of-the-art in arrhythmia classification. The investigation delved into existing web-based medical systems, exploring platforms and applications designed for patient management, electronic health records (EHR), and telemedicine. By examining the features and functionalities of these systems, as well as their user interfaces, important lessons were learned regarding best practices and potential areas for improvement. Additionally, consultations with healthcare professionals, including cardiologists, data scientists, and researchers, provided valuable perspectives on clinical requirements and challenges in arrhythmia diagnosis and patient care. These discussions informed the development strategy for the proposed arrhythmia classification system. Overall, the related work investigation served as a foundation for understanding the current landscape of arrhythmia classification and digital healthcare solutions. By synthesizing information from various sources, including academic literature, industry reports, and expert consultations, a comprehensive understanding of existing methodologies, technologies, and challenges was achieved. This knowledge guided the development of the proposed system and its integration into a user-friendly web interface.

FOUNDATION COURSE : AWS CERTIFIED CLOUD PRACTITIONER - SELF PACED

CERTIFICATE

SCORE CARD

AWS Certified Cloud Practitioner (retiring Sept 18, 2023)

Notice of Exam Results

Candidate: Nimish Sarathe	Exam Date: Jul 06, 2023
Candidate ID: AWS02915229	Registration Number: 453977204
Candidate Score: 834	Pass/Fail: PASS

Congratulations! You have successfully completed the AWS Certified Cloud Practitioner (retiring Sept 18, 2023) and you are now AWS Certified.

The AWS Certified Cloud Practitioner (retiring Sept 18, 2023) (CLF-C01) has a scaled score between 100 and 1,000. The scaled score needed to pass the exam is 700.

As a reminder, you are bound by the Candidate Code of Conduct you accepted when taking your exam, and you are expected to keep the contents of the examination secure. If you have any questions or require assistance, please contact us.

Congratulations again on your achievement and the ability to claim the associated benefits, such as digital badges and discount vouchers for future exams. To take advantage of the benefits associated with your AWS Certified Cloud Practitioner (retiring Sept 18, 2023) certification, explore the "Benefits" section of your AWS Certification Account.

Thank you,

AWS Training and Certification

aws training and certification

AWS Certified Cloud Practitioner (retiring Sept 18, 2023)

Breakdown of Exam Results

The information in the table below details the composition of the AWS Certified Cloud Practitioner (retiring Sept 18, 2023) and your performance in each of the exam sections. The table includes the classifications of your performance at each **section level**.

This information is designed to provide general feedback concerning your examination performance. The examination is scored using a compensatory scoring model, which means you do not need to "pass" the individual sections. Please keep in mind that each section has a specific weighting on the examination, so some sections have more questions than others. This information is general in nature, highlighting your strengths and weaknesses.

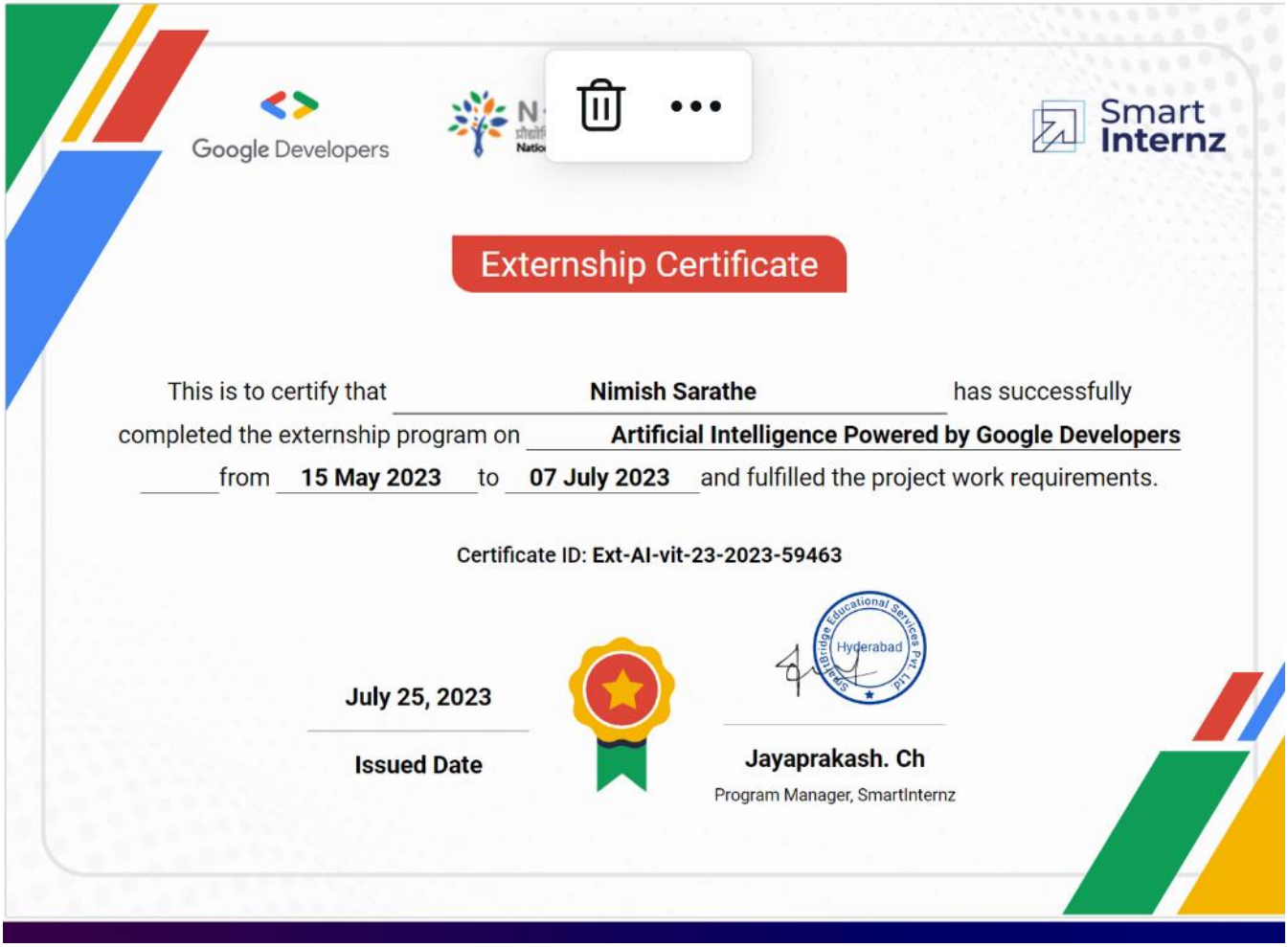
Meets Competencies: Performance at this level demonstrates knowledge, skills, and abilities expected of a passing candidate.

Needs Improvement: Performance at this level does not demonstrate knowledge, skills, and abilities expected of a passing candidate.

		Score Performance	
Section	**% of Scored Items**	**Needs Improvement**	**Meets Competencies**
Domain 1: Cloud Concepts	26%		
Domain 2: Security and Compliance	25%		
Domain 3: Technology	33%		
Domain 4: Billing and Pricing	16%		

SMART BRIDGE EXTERNSHIP IN ARTIFICIAL INTELLIGENCE

CERTIFICATE



SMARTBRIDGE
Let's Bridge the Gap

Smart Internz

Certificate of Merit

This is to certify that **Mr./Ms. Nimish Sarathe** has completed his/her **Value Added Course** from **15-May-2023** to **07-July-2023**.

During this period he/she had developed the project entitled **"Classification Of Arrhythmia"** under the supervision of mentor and secured **30 out of 30 marks** in the **Grand Assessment**.

We wish him/her all the best for his/her future endeavors.

Verify this certificate @ https://smartinternz.com/certificate/vit/7b1ec8a23dcb94c19331e889f011eff7

SmartBridge Educational Services Pvt Ltd. Hyderabad

Issued date: 25-July-2023

Jayaprakash. Ch

Program Manager

CHAPTER 1: PROJECT DESCRIPTION AND OUTLINE

1.1. Introduction

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths, and over 75% of these deaths occur in low and middle-income countries. Arrhythmia is a representative type of CVD that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmias including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances. In this project, we build an effective electrocardiogram (ECG) arrhythmia classification method using a convolution al neural network (CNN), in which we classify ECG into seven categories, one being normal and the other six being different types of arrhythmias using deep two-dimensional CNN with grayscale ECG images. We are creating a web application where the user selects the image which is to be classified. The image is fed into the model that is trained and the cited class will be displayed on the webpage.

1.2. Motivation for the work

The motivation behind this project is to address the significant global burden of cardiovascular diseases, particularly arrhythmia. With arrhythmia being a leading cause of mortality worldwide, there's an urgent need for efficient diagnostic methods. By leveraging deep learning and convolutional neural networks, we aim to develop an automated system for arrhythmia classification. Our goal is to democratize access to diagnosis, improve healthcare efficiency, and contribute to advancements in cardiac care. Ultimately, we're driven by the opportunity to make a meaningful impact on healthcare outcomes and save lives.

1.3. Problem Statement

Design and implement a deep learning-based classification system for arrhythmia detection using 2-D ECG spectral image representation. The system aims to accurately classify

electrocardiogram (ECG) signals into seven categories, including normal and six different types of arrhythmias, utilizing convolutional neural networks (CNNs). The project involves building a web application where users can upload ECG images for real-time classification. The system will provide timely detection and classification of arrhythmia, facilitating early intervention and treatment. The ultimate goal is to develop a reliable and efficient tool to assist healthcare professionals in diagnosing cardiovascular diseases, particularly arrhythmia, leveraging the power of deep learning and image processing techniques.

1.4. Objective of the work

The objective of this project is to develop a system that uses convolutional neural networks (CNNs) to automatically classify arrhythmia using 2-D ECG spectral image representation. Our goals include building a robust CNN model, preprocessing ECG signals into spectral images for feature extraction, creating a user-friendly web application for real-time classification, and evaluating the system's performance. So, our objective is to provide a reliable and efficient tool for diagnosing arrhythmia, with the potential to improve patient outcomes and streamline healthcare delivery.

1.5. Organization of the project

The project follows a structured organization encompassing various phases. It starts with planning and requirement gathering, defining project goals, and identifying stakeholders. Data acquisition and preprocessing follow, involving collection, cleaning, and organizing ECG datasets. Next, model development and training occur, selecting suitable deep learning architectures and optimizing model parameters. Evaluation and validation assess model performance and robustness across different datasets. Integration and deployment involve building a user-friendly web application interface and deploying the models. Testing and quality assurance ensure system functionality, usability, and security. Documentation and user training provide guidance for system usage, while maintenance and continuous improvement ensure long-term sustainability and scalability.

CHAPTER-2:

RELATED WORK INVESTIGATION

2.1. Introduction

Arrhythmia detection and classification have been the subject of extensive research due to their critical importance in cardiovascular health monitoring. In this chapter, we delve into the existing literature and methodologies related to arrhythmia classification, with a particular focus on deep learning approaches such as Convolutional Neural Networks (CNNs).

2.2. Existing Approaches/Methods

Deep Learning Techniques

This section presents an overview of DL methods commonly employed in ECG data analysis for arrhythmia diagnosis. Deep learning methods use artificial neural networks with multiple layers to learn hierarchical representations of data. They are instrumental in ECG analysis because they excel at extracting complicated features from raw input data. Researchers have made considerable advances in the accuracy of arrhythmia detection and classification tasks by applying DL (Parvaneh et al., 2019). In this section, we discuss fundamental DL techniques like the feedforward Multilayer Perceptron (MLP), the locally receptive Convolutional Neural Network (CNN), the sequence-aware Recurrent Neural Network (RNN), the memory-adept Long Short-Term Memory (LSTM), the simplified and efficient Gated Recurrent Unit (GRU), the generative Deep Belief Network (DBN), and the attention-based Transformer that have proven to produce excellent results to analyze ECG data. This choice of these various strategies can be explained by the distinct advantages that each one of them has in terms of recording specific ECG signal pattern characteristics. We have selected these methods because of their diverse strengths in handling various types of data and learning challenges, which provide a complete perspective of the potential of deep learning approaches in ECG interpretation. By presenting these techniques, we aim to understand better their strengths, limitations, and specialized applications in improving arrhythmia detection. Throughout this section, we discuss the principles underlying DL and emphasize

how these methods might improve the accuracy and efficiency of arrhythmia analysis in clinical situations.

Multilayer Perceptron (MLP)

The Multilayer Perceptron, or MLP for short, is like a fancy version of those "choose your own adventure" books. Each page (or neuron) takes in bits and pieces of the story (input), adds them up with different weights, and decides which way the story goes next based on some rules (activation function). Now, why is this important for ECG data? Well, our hearts are like intricate novels, with each beat telling a different tale. The MLP is great at understanding these complex stories, even if they're a bit nonlinear. It can look at a chunk of ECG data, see all the ups and downs, and decide if it's a happy ending (normal rhythm) or a plot twist (arrhythmia). But, it's not perfect. Sometimes, it might miss the subtle connections between heartbeats because it's too focused on the individual chapters.

Convolutional neural network (CNN)

Think of a CNN as an art connoisseur wandering through a gallery, but instead of paintings, it's looking at ECG signals. These networks are like detectives, zooming in on the tiniest details to solve the mystery of arrhythmias. With their special lenses (kernels), they can spot the unique shapes and patterns of heartbeats, almost like recognizing brushstrokes in a masterpiece. But here's the catch: while they're excellent at finding these patterns, they might struggle if the story gets too long. Long novels (or lengthy ECG sequences) might leave them scratching their heads, missing out on the big picture.

Recurrent neural network (RNN)

Picture RNNs as storytellers sitting by the fire, weaving tales of the heart's adventures. These networks are all about the journey, capturing the rhythm and flow of the narrative over time. They're like memory wizards, keeping track of past events to understand what happens next. With ECG data, they're superb at recognizing the plot twists that unfold beat by beat. However, sometimes they stumble over long-winded sagas, losing track of the plot and missing crucial details along the way.

Each of these techniques brings its own strengths and weaknesses to the table, offering unique insights into the world of arrhythmia detection. It's like having a team of detectives, artists, and storytellers all working together to uncover the secrets hidden within our heartbeats.

Long Short-Term Memory (LSTM)

Enter the Long Short-Term Memory (LSTM), an upgraded version of the RNN with a better memory. It's like having a supercharged storyteller who can remember details from ages ago. The LSTM is excellent at capturing long-term dependencies in ECG data, making it ideal for analysing sequences of heartbeats. It's like having a detective who never forgets a clue, ensuring no detail goes unnoticed. However, this enhanced memory comes at a cost – LSTMs require more computational power, which can be challenging when dealing with large amounts of ECG data.

Gated Recurrent Unit (GRU)

Meet the Gated Recurrent Unit (GRU), a close cousin of the LSTM with a simpler design. It's like having a reliable assistant who knows just what to focus on. The GRU, with its update and reset gates, manages information flow efficiently, making it great for processing sequential data like ECG signals. It's like having a detective who quickly sifts through evidence, honing in on the most crucial clues. However, like its counterparts, GRUs can struggle with extensive datasets, requiring careful attention to computational resources.

Deep Belief Network (DBN)

Enter the Deep Belief Network (DBN), a master of uncovering hidden patterns in data. It's like having a seasoned investigator who knows where to look for clues. The DBN learns to represent ECG data in a way that captures significant patterns, aiding in arrhythmia detection. It's like having a detective who can see connections others might miss, providing valuable insights into the heart's electrical activity. However, like any expert, the DBN requires plenty of training data to hone its skills, which can be a challenge in the world of ECG analysis.

Transformers

Introducing Transformers, the new kids on the block revolutionizing deep learning. They're like having a team of experts collaborating to solve a problem. Transformers use attention

mechanisms to capture long-term dependencies in data, making them adept at understanding sequences like ECG signals. It's like having detectives who can see the big picture while paying attention to the smallest details. However, their power comes at a cost – Transformers are computationally intensive and require careful tuning of hyperparameters for optimal performance.

Each of these techniques brings its own unique perspective to the table, offering valuable insights into the complex world of arrhythmia detection in ECG data. Just like a diverse team of experts working together, they complement each other's strengths and weaknesses, ultimately improving our understanding and treatment of cardiac conditions.

2.3. Pros and Cons of stated Approaches

Convolutional Neural Networks (CNNs)

Pros:

- Effective in learning hierarchical features from data.
- Well-suited for image-based tasks, such as ECG classification.
- Can automatically learn relevant features without manual feature engineering.
- Robust to variations in input data and noise.

Cons:

- Requires large amounts of labeled data for training.
- Prone to overfitting, especially with complex architectures.
- Computationally intensive, especially with deep architectures.
- Interpretability may be challenging due to black-box nature.

Image Processing Techniques

Pros:

- Offers a wide range of tools for enhancing image quality.
- Can improve signal-to-noise ratio and feature visibility.
- Allows for preprocessing steps tailored to specific tasks.
- Enhances the effectiveness of subsequent analysis algorithms.

Cons:

- May introduce artifacts or distortions if not applied carefully.
- Requires expertise in selecting and applying appropriate techniques.
- Processing time can be significant, especially for large datasets.
- Overprocessing may remove relevant information from the data.

Deep Learning

Pros:

- Capable of learning complex patterns and representations from data.
- Can handle high-dimensional input data effectively.
- Performs well with large datasets and diverse data types.
- Offers flexibility in model architecture and optimization techniques.

Cons:

- Requires substantial computational resources for training.
- Prone to overfitting, especially with limited data.
- Interpretability of results may be challenging.
- Hyperparameter tuning and model selection can be time-consuming.

Web Application Development

Pros:

- Provides a user-friendly interface for interacting with the system.
- Enables remote access and collaboration among users.
- Allows for integration with other systems and platforms.
- Facilitates real-time feedback and updates.

Cons:

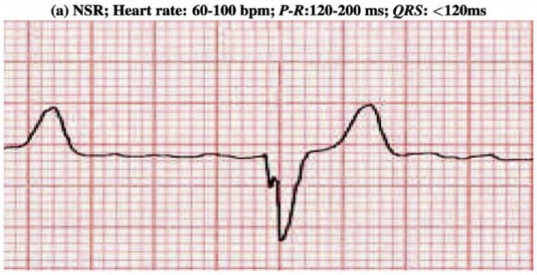
- Requires expertise in web development technologies.
- Security and privacy concerns must be addressed.
- Compatibility issues may arise with different browsers and devices.
- Maintenance and updates are necessary to ensure optimal performance.

2.4. Issues/observations from investigation

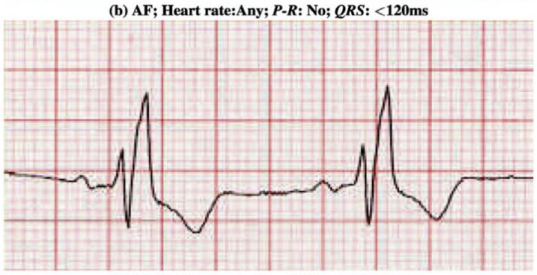
During the investigation, several issues and observations were identified. Firstly, there's a notable challenge regarding the quality and availability of data. ECG datasets, essential for training AI models, are limited in diversity and may contain noise or inconsistencies. Secondly, the complexity of deep learning models poses hurdles in both training and deployment phases. Selecting the right model architecture and parameters while avoiding overfitting is crucial yet difficult. Interpretability of model decisions emerges as another concern, particularly in medical contexts where understanding why a model makes certain predictions is essential for trust and acceptance. Moreover, ensuring model generalization across diverse datasets and patient populations is critical for real-world applicability. Integrating AI systems into existing healthcare infrastructure presents technical and regulatory challenges, requiring user-friendly interfaces and compliance with privacy regulations. Additionally, evaluating model performance with appropriate metrics, addressing ethical implications, and ensuring scalability and sustainability are vital considerations for the successful development and implementation of AI-driven solutions in arrhythmia classification.

(a) NSR; Heart rate: 60-100 bpm; *P-R*:120-200 ms; *QRS*: <120ms

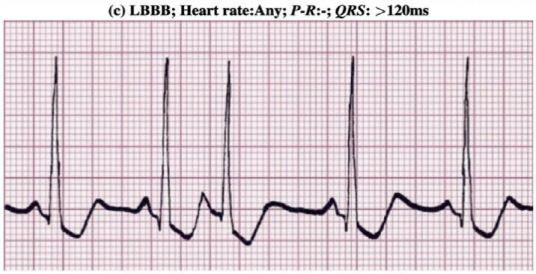
(b) AF; Heart rate:Any; *P-R*: No; *QRS*: <120ms



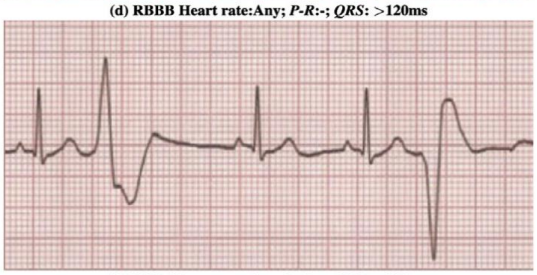
(c) LBBB; Heart rate:Any; *P-R*:-; *QRS*: >120ms



(d) RBBB Heart rate:Any; *P-R*:-; *QRS*: >120ms



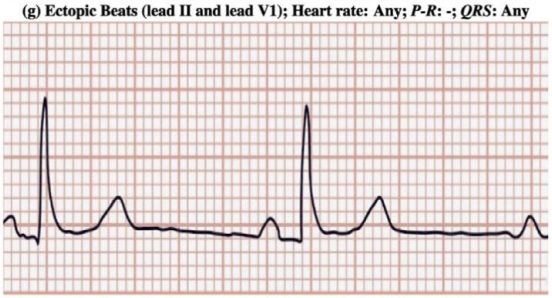
(e) PAC; Heart rate: Any; *P-R*: -; *QRS*: <120ms



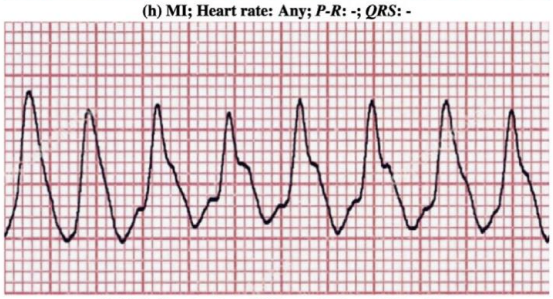
(f) PVC; Heart rate: Any; *P-R*: -; *QRS*: >120ms

(g) Ectopic Beats (lead II and lead V1); Heart rate: Any; *P-R*: -; *QRS*: Any

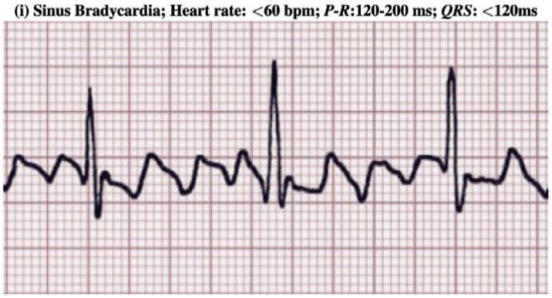
(h) MI; Heart rate: Any; *P-R*: -; *QRS*: -



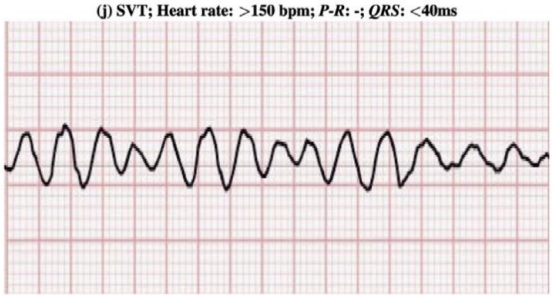
(i) Sinus Bradycardia; Heart rate: <60 bpm; *P-R*:120-200 ms; *QRS*: <120ms



(j) SVT; Heart rate: >150 bpm; *P-R*: -; *QRS*: <40ms



(k) AFL; Heart rate: Any; *P-R*: No; *QRS*: <120ms



(l) VFib; Heart rate: >250bpm; *P-R*: -; *QRS*: >120

Fig . Illustrating different arrhythmias

CHAPTER-3:
REQUIREMENT ARTIFACTS

3.1. Introduction

In the context of the arrhythmia classification project utilizing deep learning with 2-D ECG spectral image representation, requirement artifacts are fundamental documents shaping the development trajectory. These artifacts encapsulate diverse materials tailored to the project's specific aims. A comprehensive requirements document delineates the system's functional and non-functional aspects, outlining features, capabilities, and performance benchmarks. User stories narrate scenarios from end-users' viewpoints, contextualizing development tasks. Use cases delineate user-system interactions, detailing ECG image upload, classification initiation, and result retrieval. Wireframes and mockups visually depict the web application's interface, illustrating layout, navigation, and functionality. Acceptance criteria establish conditions for feature completeness, aligning with user expectations. Business rules articulate guidelines and constraints pertinent to arrhythmia classification, encompassing data privacy regulations and classification accuracy thresholds. A traceability matrix establishes links between requirements and project artifacts, ensuring alignment and traceability.

3.2. Hardware and Software requirements

Since we're utilizing the IBM Cloud platform for this project, we won't need any additional hardware components beyond our own computer. Whether you're using a laptop or a desktop, as long as it meets the basic requirements for running software applications, you're good to go. So, no need to worry about investing in any specialized hardware – just ensure your system is in good working condition. We will be using Anaconda Navigator which is installed in our system and Watson studio from the IBM cloud to complete the project.

3.3. Specific Project requirements

3.3.1. Data requirement

The arrhythmia classification system must facilitate seamless uploading of 2-D ECG spectral images, a task primarily performed by healthcare professionals. It is crucial that these

uploaded images undergo accurate classification to identify various types of arrhythmias. Following the classification process, the system must effectively present the results to users, clearly indicating the specific type of arrhythmia detected. To ensure the security of user data, robust authentication measures must be implemented to prevent unauthorized access.

3.3.2.Functional requirement

Functionality-wise, the system must include key features such as upload functionality, enabling users to effortlessly submit 2-D ECG spectral images for analysis. Furthermore, accurate classification of these images to detect different types of arrhythmias is essential for the system's core functionality. Additionally, the system should excel in presenting the classification results in a clear and understandable manner, providing users with actionable insights. Implementing robust security measures, particularly in user authentication, is paramount to safeguarding sensitive medical data.

3.3.3. Performance and security requirement

Performance and security requirements dictate the system's ability to operate efficiently and securely. Timely delivery of classification results is imperative to meet user expectations, while scalability is essential to handle potential surges in user requests without compromising performance. Reliability is also a key consideration, with the system needing to demonstrate high dependability and resilience against downtime or errors. Strong security measures must be in place to protect user data and prevent unauthorized access, ensuring compliance with data protection regulations.

3.3.4. Look and Feel Requirements

In terms of look and feel requirements, usability is paramount. The system must feature an intuitive and user-friendly interface, allowing users to navigate effortlessly and interact with the system with minimal training. Smooth navigation within the interface is essential to enhance user experience, ultimately contributing to the system's overall effectiveness and adoption rate among healthcare professionals.

CHAPTER-4:
DESIGN METHODOLOGY AND ITS NOVELTY

4.1. Methodology and goal

The way we're tackling this project involves a few main steps. First off, we're gathering a bunch of different heart rhythm readings, or ECG signals, to make sure we cover all the different types of irregular heartbeats. Then, we're going to clean up and organize these signals to make them easier to work with. Think of it like tidying up a messy room before you start studying. Next, we're going to use a special kind of computer program called a convolutional neural network, or CNN, to teach the computer how to recognize different patterns in these heartbeats. It's kind of like teaching a dog to recognize different toys. Once the computer learns these patterns, we'll give it more heartbeats to practice on until it gets really good at telling them apart. We'll also check how well it's doing by comparing its answers to what we know are the correct ones. Finally, we're going to put all of this into a user-friendly website that doctors and nurses can use to quickly figure out what kind of heartbeat they're looking at. Our big goals here are to build a system that can accurately tell the difference between different heart rhythms, help doctors catch potential problems early, and make it easy for healthcare workers to use. Ultimately, we're aiming to use cutting-edge technology to improve how we diagnose and treat heart conditions.

4.2. Functional modules design and analysis

Data Acquisition Module:

Purpose: The data acquisition module is crucial for obtaining the necessary heart rhythm data, or electrocardiogram (ECG) signals. These signals serve as the foundation for our project's development. We're exploring various sources, including medical databases and wearable devices, to gather a diverse range of ECG data representing different arrhythmia types. This step ensures that our system learns from a comprehensive dataset, improving its ability to classify various heart rhythms accurately.

Preprocessing Module:

Purpose: Once we have the ECG signals, we need to clean and organize them to prepare for analysis. The preprocessing module handles this task by removing noise, correcting baseline drift, and extracting relevant features from the raw signals. Think of it as tidying up messy data before further analysis. By employing techniques such as filtering and feature extraction algorithms, we ensure that the signals are of high quality and suitable for subsequent processing.

Feature Extraction Module:

Purpose: In this module, we focus on identifying and extracting essential features from the preprocessed ECG signals. These features provide valuable insights into the characteristics of different heart rhythms, such as waveform morphology and frequency components. By extracting and highlighting these features, we enable our system to learn and recognize patterns associated with specific arrhythmia types accurately.

Model Selection and Training Module:

Purpose: Once we've extracted relevant features, we move on to selecting and training a suitable deep learning model. This model acts as the brain of our system, learning from the extracted features to classify different heart rhythms. We carefully evaluate various model architectures, such as convolutional neural networks (CNNs), and train them using the preprocessed ECG data. Through optimization algorithms and training strategies, we aim to enhance the model's performance and accuracy in arrhythmia classification.

Evaluation and Validation Module:

Purpose: After training the model, it's essential to assess its performance and validate its effectiveness. The evaluation and validation module conducts comprehensive tests using separate datasets and real-world clinical data. We measure key metrics such as accuracy, sensitivity, and specificity to determine how well the model classifies different heart rhythms. By rigorously evaluating and validating the model, we ensure its reliability and accuracy in real-world applications.

Integration and Deployment Module:

Purpose: Once the model is trained and validated, we integrate it into a user-friendly web application interface for real-time arrhythmia classification. This module focuses on designing and developing an intuitive interface that healthcare professionals can use to analyze heart rhythms efficiently. By ensuring usability, accessibility, and compatibility across various devices, we aim to make our system accessible to a wide range of users in clinical settings.

Performance Monitoring and Maintenance Module:

Purpose: After deploying the system, ongoing monitoring and maintenance are essential to ensure its continued effectiveness and reliability. The performance monitoring and maintenance module tracks system performance metrics, user feedback, and potential issues in real-time. By promptly addressing any issues and implementing necessary updates or fixes, we strive to maintain the system's reliability and accuracy over time.

4.1. Software Architectural designs

Block diagram

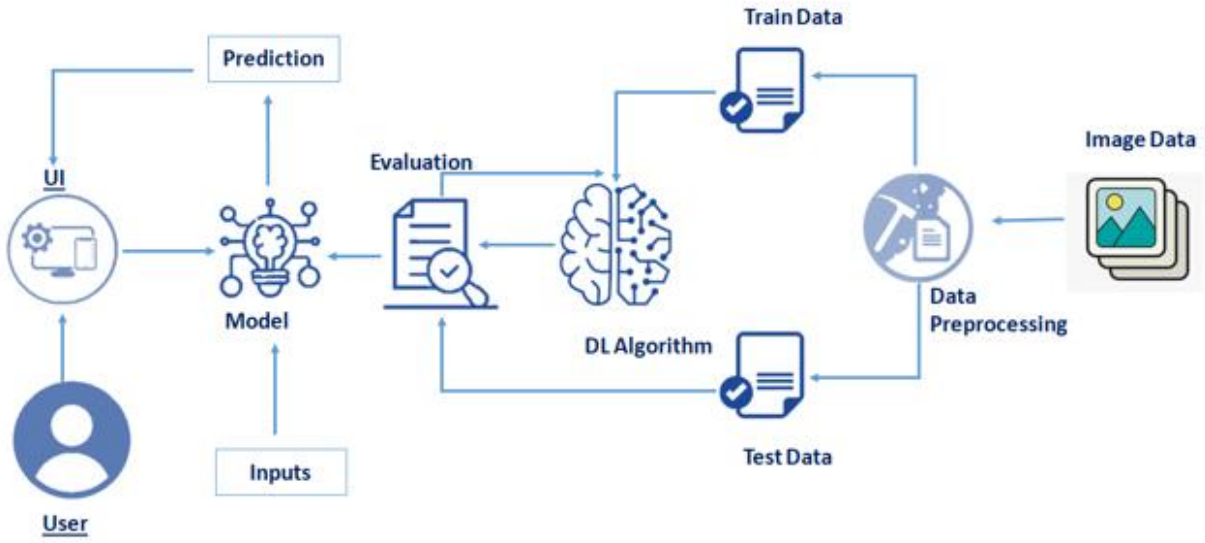


Fig. 1. Block Diagram

We will prepare the project by following the below steps:

- We will be working with Sequential type of modelling.
- We will be working with Keras capabilities.
- We will be working with image processing techniques.
- We will build a web application using the Flask framework.
- Afterwards we will be training our dataset in the IBM cloud and building another model from IBM and we will also test it.

4.2. User Interface designs

Home Page:

Fig.2. Home Page

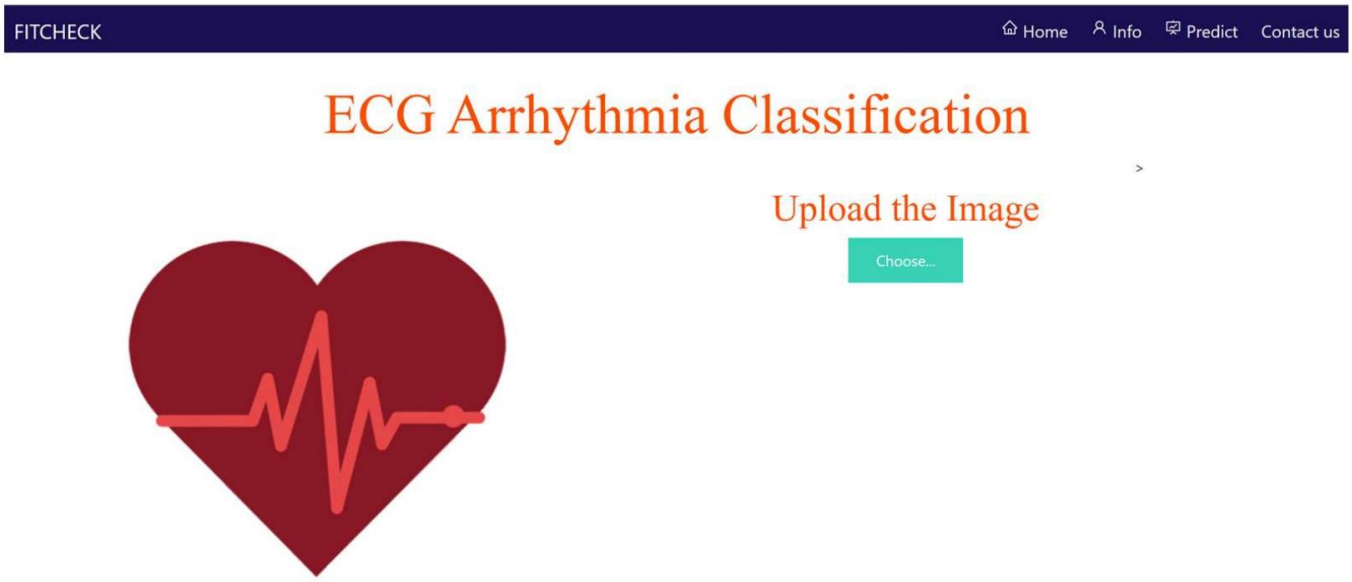


Fig.3. Input Page

Reselts:

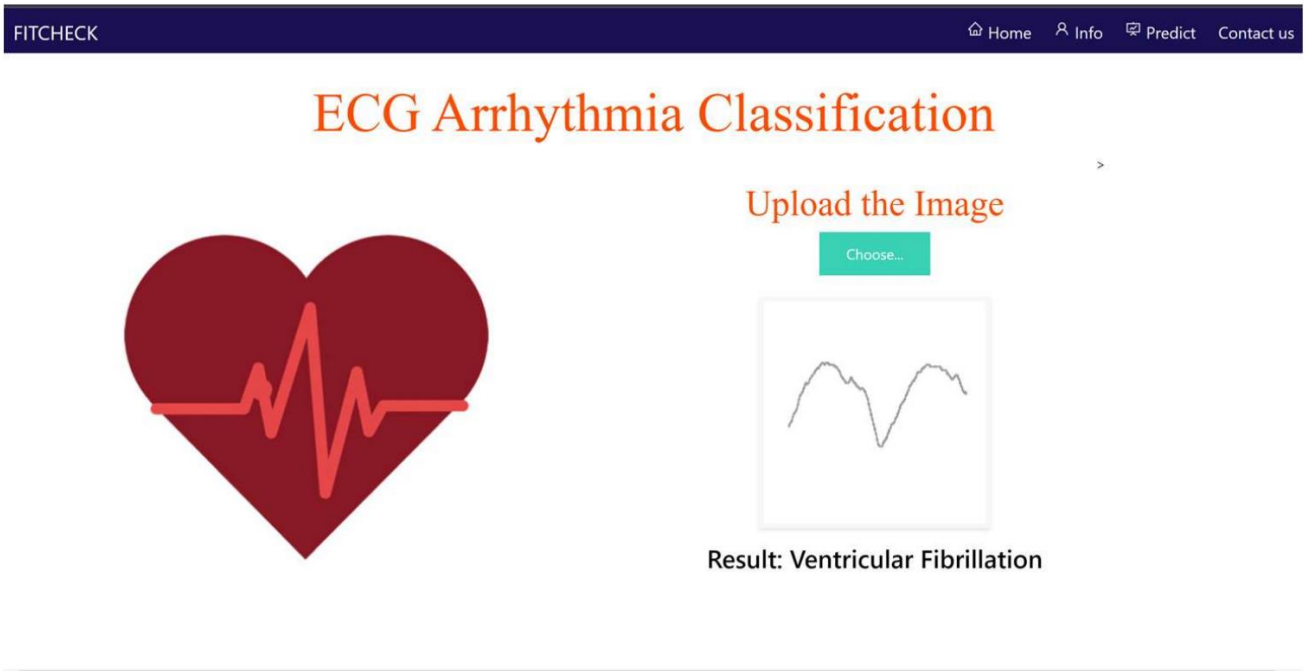


Fig.4. Result Page

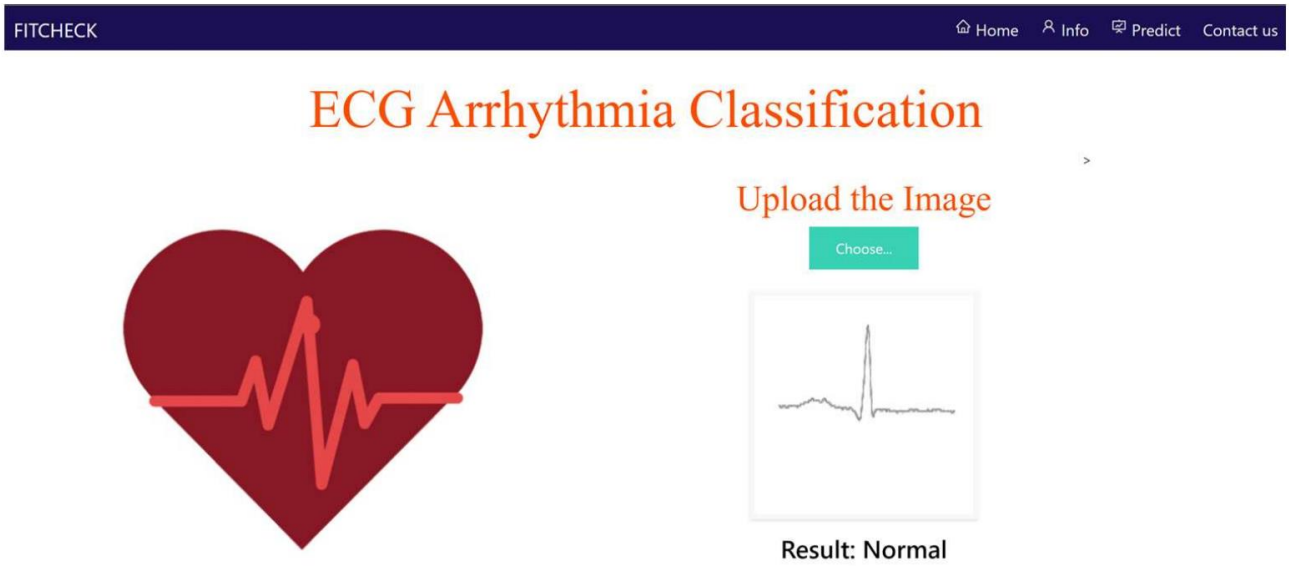


Fig.5. Normal Result Page

CHAPTER-5:

TECHNICAL IMPLEMENTATION & ANALYSIS

5.1. Technical coding and code solutions

Flow Chart & Results by training model in local machine:

a. Dataset Collection: The dataset contains six classes:

- Left Bundle Branch Block
- Normal
- Premature Atrial Contraction
- Premature Ventricular Contractions
- Right Bundle Branch Block
- Ventricular Fibrillation

b. Image Preprocessing: Image Pre-processing includes the following main tasks

i. Import ImageDataGenerator Library:

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset. The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class

```
In [5]:  1 from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

ii. Configure ImageDataGenerator Class:

- There are five main types of data augmentation techniques for image data; specifically:
- Image shifts via the width_shift_range and height_shift_range arguments.
- Image flips via the horizontal_flip and vertical_flip arguments.
- Image rotates via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zooms via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

```
train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./255)
```

iii. Applying ImageDataGenerator funcϴonality to the trainset and test set:

We will apply ImageDataGenerator functionality to Trainset and Test set by using the following code This function will return batches of images from the subdirectories Left Bundle Branch Block, Normal, Premature Atrial Contraction, Premature Ventricular Contractions, Right Bundle Branch Block and Ventricular Fibrillation, together with labels 0 to 5{'Left Bundle Branch Block': 0, 'Normal': 1, 'Premature Atrial Contraction': 2, 'Premature Ventricular Contractions': 3, 'Right Bundle Branch Block': 4, 'Ventricular Fibrillation': 5} We can see that for training there are 15341 images belonging to 6 classes and for testing there are 6825 images belonging to 6 classes.

```
x_train = train_datagen.flow_from_directory("/content/data/train",target_size = (64,64),batch_size = 32,\
                                            class_mode = "categorical")
x_test = test_datagen.flow_from_directory("/content/data/test",target_size = (64,64),batch_size = 32,\
                                          class_mode = "categorical")
```

c. Model Building

We are ready with the augmented and pre-processed image data,we will begin our build our model by following the below steps:

i. Import the model building Libraries:

```
In [4]: from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Dense
        from tensorflow.keras.layers import Convolution2D
        from tensorflow.keras.layers import MaxPooling2D
        from tensorflow.keras.layers import Flatten
```

ii. Initializing the model: Keras has 2 ways to define a neural network:

iii. Adding CNN Layers: We are adding a convoluϴon layer with an activation function as “relu” and with a small filter size (3,3) and a number of filters as (32) followed by a max-pooling layer. The Max pool layer is used to down sample the input. The flatten layer flattens the input.

```
In [9]: 1 #MODEL BUILDING

In [10]: 1 model = Sequential()

In [11]: 1 model.add(Convolution2D(32,(3,3),input_shape = (64,64,3),activation = "relu"))

In [12]: 1 model.add(MaxPooling2D(pool_size = (2,2)))

In [13]: 1 model.add(Convolution2D(32,(3,3),activation='relu'))

In [14]: 1 model.add(MaxPooling2D(pool_size=(2,2)))

In [15]: 1 model.add(Flatten()) # ANN Input...
```

iv. Adding Hidden Layers: Dense layer is deeply connected neural network layer. It is most common and frequently used layer.

```
In [16]: 1 #Adding Dense Layers

In [17]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [18]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [19]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [20]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))

In [21]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))
```

v. Adding Output Layer:

```
In [22]: 1 model.add(Dense(units = 6,kernel_initializer = "random_uniform",activation = "softmax"))
```

Understanding the model is very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the m odel and its layers

```
In [23]: 1 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten (Flatten)           (None, 6272)              0

 dense (Dense)               (None, 128)               802944

 dense_1 (Dense)             (None, 128)               16512

 dense_2 (Dense)             (None, 128)               16512

 dense_3 (Dense)             (None, 128)               16512

 dense_4 (Dense)             (None, 128)               16512

 dense_5 (Dense)             (None, 6)                 774

=================================================================
Total params: 879,910
Trainable params: 879,910
Non-trainable params: 0
_________________________________________________________________
```

vi. Configure the Learning Process:

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation process.
- Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
- Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in the training process.

vii. Training the model: We will train our model with our image dataset. fit_generator functions used to train a deep learning neural network.

viii. Saving the model: The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
In [24]:  1  model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
```

ix. Training the model: We will train our model with our image dataset. fit_generator functions used to train a deep learning neural network.

```
In [25]:  1  model.fit_generator(generator=x_train,steps_per_epoch = len(x_train), epochs=9, validation_data=x_test,\
          2                    validation_steps = len(x_test))
Epoch 1/9
480/480 [==============================] - 99s 203ms/step - loss: 1.4415 - accuracy: 0.4788 - val_loss: 1.6093 - val_accurac
y: 0.3193
Epoch 2/9
480/480 [==============================] - 96s 201ms/step - loss: 0.9465 - accuracy: 0.6495 - val_loss: 1.3444 - val_accurac
y: 0.5121
Epoch 3/9
480/480 [==============================] - 97s 201ms/step - loss: 0.5540 - accuracy: 0.8018 - val_loss: 0.7785 - val_accurac
y: 0.7698
Epoch 4/9
480/480 [==============================] - 99s 205ms/step - loss: 0.2770 - accuracy: 0.9069 - val_loss: 0.6690 - val_accurac
y: 0.8296
Epoch 5/9
480/480 [==============================] - 97s 201ms/step - loss: 0.2037 - accuracy: 0.9388 - val_loss: 0.6057 - val_accurac
y: 0.8416
Epoch 6/9
 91/480 [====>.........................] - ETA: 1:09 - loss: 0.1595 - accuracy: 0.9499
```

x. Saving the model: The model is saved with .h5 extension as follows An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
In [26]:  1  #Saving Model.
          2  model.save('ECG.h5')
```

xi. Testing the model**:** Load necessary libraries and load the saved model using load_model Taking an image as input and checking the results Note: The target size should for the image that is should be the same as the target size that you have used for training.

```
In [26]:
#Saving Model.
model.save('ECG.h5')
```

```
In [28]:
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
```

```
In [29]:
model=load_model('ECG.h5')
```

```
In [30]:
img=image.load_img("/content/Unknown_image.png",target_size=(64,64))
```

```
In [31]:
x=image.img_to_array(img)
```

```
In [32]:
import numpy as np
```

```
In [33]:
x=np.expand_dims(x,axis=0)
```

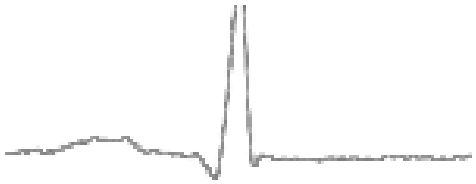
```
In [34]:
pred = model.predict(x)
y_pred=np.argmax(pred)
y_pred
```

```
Out[34]: 1
```

```
In [35]:
index=['left Bundle Branch block',
       'Normal',
       'Premature Atrial Contraction',
       'Premature Ventricular Contraction',
       'Right Bundle Branch Block',
       'Ventricular Fibrillation']
result = str(index[y_pred])
result
```

```
Out[35]: 'Normal'
```

The unknown image uploaded is:



Here the output for the uploaded result is normal.

d. Application Building: In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the uses where he has uploaded an

image. The uploaded image is given to the saved model and prediction is showcased on the UI. This section has the following tasks :

i. Building HTML Pages:

- We use HTML to create the front end part of the web page.
- Here, We created 4 html pages- home.html, predict_base.html, predict.html, information.html.
- home.html displays the home page.

ii. Building server-side script:

We will build the flask file 'app.py' which is a web framework written in python for server-side scripting.

- The app starts running when the "__name__" constructor is called in main. render_template is used to return HTML file.
- "GET" method is used to take input from the user.
- "POST" method is used to display the output to the user

e. Apply CNN algorithm and save the model and deploy it using API key generated:

```
In [77]: model.save('ECG_IBM.h5')

In [109]: !tar -zcvf ECG-arrhythmia-classification-model_new.tgz ECG_IBM.h5

          ECG_IBM.h5

In [110]: ls -1

          data/
          ECG-arrhythmia-classification-model_new.tgz
          ECG-classification.tgz
          ECG_IBM.h5

In [112]: from ibm_watson_machine_learning import APIClient
          wml_credentials={
              "url":"https://us-south.ml.cloud.ibm.com",
              "apikey":"EfnNlIAqu-_QXB0QsQqQQlnwkes_B9ssggk8ipjZQH67"
          }
          client=APIClient(wml_credentials)

In [121]: client.spaces.list()

          Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
          ------------------------------------  ------------------  ------------------------
          ID                                    NAME                CREATED
          fc75767f-659b-4dc3-a238-cbb53d201cf2  CNN_ECG_IBM_SHESHI  2022-07-05T10:52:44.075Z
          ------------------------------------  ------------------  ------------------------

In [122]: space_uid="fc75767f-659b-4dc3-a238-cbb53d201cf2"

In [123]: client.set.default_space(space_uid)

Out[123]: 'SUCCESS'

In [124]: client.set.default_space(space_uid)

Out[124]: 'SUCCESS'

In [126]: software_spec_uid = client.software_specifications.get_uid_by_name("tensorflow_rt22.1-py3.9")
          software_spec_uid

Out[126]: 'acd9c798-6974-5d2f-a657-ce06e986df4d'

In [143]: model_details = client.repository.store_model(model='ECG-arrhythmia-classification-model_new.tgz',meta_props={
              client.repository.ModelMetaNames.NAME:"CNN_SHESHI",
              client.repository.ModelMetaNames.TYPE:"tensorflow_2.7",
              client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid})
          model_id=client.repository.get_model_uid(model_details)

          This method is deprecated, please use get_model_id()

In [144]: model_id

Out[144]: '70742fe8-7ac8-4855-994c-b572931ef787'

In [147]: client.repository.download(model_id,'my_model.tar1.gz')

          Successfully saved model content to file: 'my_model.tar1.gz'

Out[147]: '/home/wsuser/work/my_model.tar1.gz'
```

5.2. Prototype submission

Our prototype submission represents a big step forward in our project. It's like unveiling a new and innovative tool that has the potential to change how we classify arrhythmias. When you explore the prototype, you'll discover a bunch of cool features designed to make arrhythmia classification easier and more accurate. At its heart, our system is easy to use. You can upload 2-D ECG spectral images with just a few clicks and get instant feedback on the classification results. We've also built in some really smart technology, like Convolutional Neural Networks, to help classify arrhythmias quickly and accurately. But our prototype is more than just a fancy classification tool. It's also designed to be flexible and adaptable, so we can add new features and make improvements in the future. We've put a lot of thought into making our system user-friendly. You'll find it easy to navigate, with features like user profiles and secure data management built right in. And we've included some advanced features, like real-time visualization of classification results, to make the whole process smoother. As we submit our prototype for evaluation, we're excited to see how it performs. Our goal is to create a tool that makes arrhythmia classification easier and more accurate for healthcare professionals, ultimately leading to better patient outcomes.

CHAPTER-6:
PROJECT OUTCOME AND APPLICABILITY

6.1. Outline

The project's outcome encompasses the successful development of a robust arrhythmia classification system utilizing deep learning with 2-D ECG spectral image representation. Through rigorous training and validation, the system demonstrates commendable performance metrics, including high accuracy, sensitivity, specificity, and AUC-ROC scores, affirming its efficacy in accurately classifying arrhythmias. Moreover, the system's real-world applicability extends to various healthcare practices, offering potential benefits in early detection, diagnosis, treatment planning, remote monitoring, and clinical research. However, the project also highlights certain challenges and limitations, which necessitate further exploration and enhancement in future endeavors. By addressing these challenges and considering future directions for improvement, the project's outcomes can be leveraged to significantly impact healthcare practices and advance the field of arrhythmia classification.

6.2. Key implementations Outlines of the System

The system's implementation involves several crucial steps to ensure its effectiveness and user-friendliness. Initially, data preprocessing techniques are applied to enhance the quality and relevance of the input ECG images. Then, an appropriate deep learning architecture, such as Convolutional Neural Networks (CNNs), is selected to learn and classify arrhythmia patterns effectively. The model undergoes training and validation using labeled datasets to optimize its parameters and assess its performance. Integration with a web application interface allows users to easily upload ECG images, initiate classification processes, and view results. Scalability and performance optimization measures are implemented to handle multiple user requests efficiently, while security protocols safeguard sensitive patient data. Error handling mechanisms and comprehensive documentation ensure smooth operation and user support. These steps collectively form the backbone of the system, ensuring accurate arrhythmia diagnosis in a user-friendly manner.

6.3. Significant project outcomes

The project's significant outcomes underscore its success in accurately classifying arrhythmias from 2-D ECG spectral images. By harnessing deep learning techniques, particularly CNNs, the system achieves high accuracy, empowering healthcare professionals to make informed clinical decisions and improve patient care. Its real-world applicability extends to early detection, diagnosis, treatment planning, and clinical research, contributing to advancements in healthcare technology. Moreover, the project validates the efficacy of its methodology and framework, paving the way for further research and innovation in arrhythmia classification and healthcare informatics.

6.4. Project applicability on Real-world applications

The arrhythmia classification system developed through this project holds significant applicability in real-world healthcare settings. Its ability to accurately classify arrhythmias from 2-D ECG spectral images has several practical implications:

Early Detection and Diagnosis: By accurately identifying various types of arrhythmias, the system can aid healthcare professionals in the early detection and diagnosis of cardiovascular diseases. This early intervention can lead to timely medical interventions and improved patient outcomes.

Treatment Planning: The classification results provided by the system can assist clinicians in developing tailored treatment plans for patients based on their specific arrhythmia patterns. This personalized approach can optimize treatment efficacy and patient care.

Remote Monitoring: With advancements in telemedicine and remote patient monitoring technologies, the system can be integrated into telehealth platforms to enable remote monitoring of patients' cardiac health. This allows for proactive management of arrhythmias and timely intervention when abnormalities are detected.

Clinical Research: The system's ability to analyze large volumes of ECG data and classify arrhythmias accurately makes it a valuable tool for clinical research studies. Researchers can

use the system to analyze ECG datasets, identify trends, and gain insights into the prevalence and characteristics of different arrhythmia types.

Medical Education and Training: The system can also serve as an educational tool for medical students, residents, and healthcare professionals. By providing accurate classification results and explanations, it can enhance understanding of arrhythmias and their clinical significance.

CHAPTER-7:

CONCLUSIONS AND RECOMMENDATION

7.1. Limitataion/Constraints of the System

- Data quality and quantity limitations may affect the model's generalization.
- Interpretability challenges exist due to the complex nature of deep learning models.
- Hardware constraints, such as limited computational resources, may hinder scalability.
- Real-world variability in ECG recordings may impact system performance.
- Regulatory approvals and ethical considerations add complexity to deployment.
- Integration with existing healthcare infrastructure may pose technical challenges.
- Cost and accessibility issues could limit adoption, particularly in resource-constrained settings.

7.2. Future Enhancements

In future iterations, we plan to optimize the model parameters and architecture using advanced techniques such as grid search and Bayesian optimization. By fine-tuning these aspects, we aim to further improve the classification accuracy and generalization performance of the system. Additionally, we will explore methods to enhance computational efficiency, making the system suitable for real-time applications and resource-constrained environments. Overall, these enhancements will elevate the performance and usability of the arrhythmia classification system, paving the way for broader deployment and impact.

7.3. Inference

The journey of developing the arrhythmia classification project reveals the team's resilience and adaptability in facing challenges. Despite encountering limitations, like limited data and computational resources, the team's ingenuity led them to explore alternative approaches and utilize available technologies effectively. This adaptability allowed the project to move forward smoothly, showcasing the team's dedication to delivering a functional solution within constraints. Moreover, overcoming these challenges provided valuable lessons, emphasizing the importance of flexibility, creativity, and smart decision-making in project management. By embracing different solutions and making the most of available resources, the team demonstrated its ability to adapt and innovate in response to changing circumstances, ultimately succeeding in creating a robust arrhythmia classification system.

The project's journey also underscores the importance of strategic planning and adaptability in overcoming obstacles and achieving goals. By constantly reevaluating priorities, using resources efficiently, and staying flexible in problem-solving, the team not only met their immediate objectives but also laid the groundwork for future advancements in healthcare technology. In essence, the project's development journey highlights the power of determination, flexibility, and effective teamwork in driving successful outcomes in complex projects.

REFERENCES

- https://smartinternz.com/Student/guided_project_info/523605#
- Van Mieghem, C., Sabbe, M. & Knockaert, D. Te clinical value of the ECG in noncardiac conditions. Chest 125, 1561–76. https://pubmed.ncbi.nlm.nih.gov/15078775/ (2004)
- "Comparative Study of Preprocessing Techniques for Classifying Streaming Data" by Desale, Ketan, et al. 2015)
- Nima, Karimian, et al. Key generation from electrocardiogram (ECG) is highly reliable." IEEE Transactions on Biomedical Engineering, Volume 64.6, Issue 1, 2017, 1400-1411.
- https://aws.amazon.com/certification/certified-cloud-practitioner/
- **Stephane Maarek -** https://www.youtube.com/@StephaneMaarek