CARDIOAI:ENHANCING HEART ATTACK DETECTION IN DIABETIC PATIENTS THROUGH ECG ANALYSIS

A MINI PROJECT REPORT

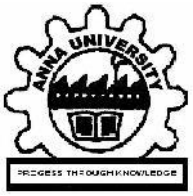
Submitted by

BENJAMIN NICOLAS S
(221801005)

CHARLESS BINNY K
(22180101007)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
ARTIFICIAL INTELLIGENCE AND DATA SCIENCE



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI

NOV 2024

BONAFIDE CERTIFICATE

Certified that this Report titled "**ENHANCING HEART ATTACK DETECTION IN DIABETIC PATIENTS THROUGH ECG ANALYSIS**" is the bonafide work of **BENJAMIN NICOLAS S(221801005),CHARLESS BINNY K(221801007)** who carried out the work under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr.J.M.Gnanasekar M.E.,Ph.D.,
Professor and Head
Department of AI&DS
Rajalakshmi Engineering College
Chennai-602 105

SIGNATURE

Mrs.Y.Nirmala Anandhi M.E.,
Assistant Professor(SS)
Department of AI&DS
Rajalakshmi Engineering College
Chennai-602 105

Submitted for the project viva-voce examination held on_____.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Initially we thank the Almighty for being with us through every walk of our life and showering his blessings through the endeavor to put forth this report. Our sincere thanks to our Chairman **Mr. S. MEGANATHAN, B.E, F.I.E.**, our respected Chairperson **Dr. (Mrs.) THANGAM MEGANATHAN, Ph.D.**, and our Vice Chairman **Mr. ABHAY SHANKAR MEGANATHAN, B.E., M.S.**, for providing us withthe requisite infrastructure and sincere endeavoring in educating us in their premier institution.

Our sincere thanks to **Dr. S.N. MURUGESAN, M.E., Ph.D.**, our beloved Principal for his kind support and facilities provided to complete our work in time. We express our sincere thanks to **Dr. J.M.GNANASEKAR, M.E., Ph.D.**, Head of the Department, Professor and Head of the Department of Artificial Intelligence and Data Science for his guidance and encouragement throughout the project work. We are glad to express our sincere thanks and regards to our supervisor **Mrs.Y.NIRAMALA ANANDHI, M.E.,** Assistant Professor(SS), Department of Artificial Intelligence and Data Science, Rajalakshmi Engineering College and coordinator, **Dr.P.INDRA PRIYA.,M.E.,Ph.D.**,Professor,Department of Artificial Intelligence and Data Science, Rajalakshmi Engineering College for their valuable guidance throughout the course of the project.

Finally, we express our thanks for all teaching, non-teaching, faculty and our parents for helping us with the necessary guidance during the time of our project.

ABSTRACT

Cardiovascular diseases, particularly myocardial infarction (MI), are critical health risks, especially for diabetic patients, who face a heightened susceptibility to heart complications. This project seeks to develop a machine learning model for the early detection of MI through the analysis of electrocardiogram (ECG) images. By utilizing a comprehensive dataset of ECG images from diabetic patients and those diagnosed with MI, this research applies advanced image preprocessing techniques to enhance data quality, ensuring clearer and more consistent inputs for analysis. A convolutional neural network (CNN) is then designed to automatically extract relevant features from the ECG images, facilitating accurate classification based on the presence or absence of myocardial infarction. The model's effectiveness is rigorously evaluated using performance metrics such as accuracy, precision, recall, F1 score, and AUC-ROC, providing a comprehensive view of its diagnostic capabilities. By automating the ECG analysis process, this project aims to reduce the dependency on manual interpretation, which can be time-consuming and prone to variability. With a focus on high-risk diabetic patients, the model offers potential as a reliable diagnostic tool that supports timely medical interventions, ultimately improving patient outcomes. This study highlights the potential for machine learning to advance cardiovascular disease management, and the findings will contribute to future research in automated ECG analysis and early detection of cardiac conditions. By addressing the specific challenges associated with diabetes-related cardiovascular risks, this model not only enhances MI detection but also lays the groundwork for broader applications of AI in healthcare diagnostics.

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	IV
	LIST OF FIGURES	VII
1	**INTRODUCTION**	
	1.1 GENERAL	1
	1.2 NEED FOR THE STUDY	1
	1.3 OBJECTIVES FOR THE STUDY	2
	1.4 OVERVIEW OF THE PROJECT	2
2	**REVIEW OF LITERATURE**	5
3	**SYSTEM OVERVIEW**	
	3.1 EXISTING SYSTEM	7
	3.2 PROPOSED SYSTEM	8
	3.3 FEASIBILITY STUDY	10
4	**SYSTEM REQUIREMENTS**	
	4.1 HARDWARE REQUIREMENTS	12
	4.2 SOFTWARE REQUIREMENTS	12
	4.3 DATASET REQUIREMENTS	13
5	**SYSTEM DESIGN**	
	5.1 SYSTEM ARCHUTECTURE	15
	5.2 MODULE DESCRIPTION	16
6	**RESULT AND DISCUSSION**	31

7 CONCLUSION AND FUTURE ENHANCEMMENT

7.1 CONCLUSION 33

7.2 FUTURE ENHANCEMENT 33

APPENDIX

A1.1 SAMPLE CODE 34

A1.2 SCREENSHOTS 41

REFERENCES 45

LIST OF FIGURES:

FIGURE NO	FIGURE NAME	PAGE NO
5.1.1	System Architecture	15
5.2.1	Feature Extraction and Model Training	17
5.2.2	Model Layers and Binary Classification	18
5.2.3	Model Compilation, Training, and Evaluation	19
5.2.4	Model Saving and Deployment	21
A1.2.1	Data Augmentation	41
A1.2.2	Feature Engineering	41
A1.2.3	Class Distribution	42
A1.2.4	Validation	42
A1.2.5	Patient Deatails Page	43
A1.2.6	ECG Uploading Page	43
A1.2.7	Predicted Result Page	44

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Cardiovascular diseases, including myocardial infarction, are leading causes of death globally. Early detection is essential, as timely diagnosis can greatly improve patient outcomes. Diabetic patients are especially vulnerable, facing higher risks of cardiovascular complications. Traditionally, diagnosing myocardial infarction relies on manual interpretation of ECG data, which can be slow and prone to error. Advances in machine learning offer promising solutions to automate ECG analysis, improving accuracy, speed, and reliability, especially for high-risk groups like diabetic patients.

1.2 NEED FOR STUDY

The need for this study stems from the high prevalence and severity of cardiovascular diseases, particularly myocardial infarction (MI), which is a major cause of morbidity and mortality worldwide. Diabetic patients are at an especially elevated risk of developing cardiovascular complications, making early and accurate detection of MI crucial for improving their health outcomes. Traditional diagnostic methods for MI, including manual interpretation of electrocardiograms (ECG), are often time-consuming and may vary in accuracy due to human error. This variability can lead to delays in diagnosis and treatment, which are critical in acute cases like MI where timely intervention is essential.

Machine learning offers a potential solution by automating the analysis of ECG images, providing a faster and more consistent approach to MI detection. This study aims to develop a convolutional neural network (CNN) model that can analyze ECG images to detect MI early, particularly in diabetic patients. Automating this process could reduce the burden on healthcare professionals, increase diagnostic accuracy, and enable timely intervention, thereby reducing the risks associated with delayed diagnosis. This study will contribute valuable insights into the potential of AI in

healthcare, specifically for enhancing diagnostic efficiency and improving patient outcomes in cardiovascular disease management.

1.3 OBJECTIVES OF THE STUDY

The objective of this study is to develop and evaluate a machine learning model that can accurately detect myocardial infarction (MI) in patients, particularly those with diabetes, by analyzing electrocardiogram (ECG) images. Specific objectives include:

1. To create a comprehensive dataset of ECG images, including samples from diabetic patients and those diagnosed with MI.
2. To apply advanced image preprocessing and data augmentation techniques to improve image quality and ensure consistency across inputs.
3. To design a convolutional neural network (CNN) capable of automatically extracting features from ECG images for reliable MI detection.
4. To assess the model's performance using metrics such as accuracy, precision, recall, F1 score, and AUC-ROC, establishing its potential effectiveness as a diagnostic tool.
5. To investigate the model's implications for clinical practice, aiming to provide a faster and more reliable method for MI detection, ultimately improving patient outcomes through timely diagnosis and intervention.

1.4.OVERVIEW OF THE PROJECT

This project centers on developing a machine learning model to improve the early detection of myocardial infarction (MI) by analyzing electrocardiogram (ECG) images, specifically addressing the needs of high-risk groups like diabetic patients. Cardiovascular diseases, especially MI, present severe health risks and contribute to high morbidity and mortality rates globally. Diabetic patients are particularly vulnerable, as their risk of developing cardiovascular complications is significantly elevated. Traditional diagnostic methods rely heavily on manual interpretation of ECG signals, which can be time-consuming, prone to variability, and less effective in ensuring early detection. This project proposes an automated, accurate, and

efficient system using machine learning techniques to tackle these limitations and improve the diagnostic process.

The proposed system is based on a convolutional neural network (CNN), a deep learning model well-suited for image-based analysis, which has proven effective in medical image classification tasks. The first phase of the project involves collecting a diverse and comprehensive dataset of ECG images, including data from both diabetic patients and individuals diagnosed with MI. This dataset is then subjected to rigorous preprocessing techniques, which enhance image quality and prepare the data for analysis. Steps such as resizing, noise removal, normalization, and data augmentation are applied to ensure consistency and improve the model's ability to generalize across new, unseen data. Data augmentation, in particular, is crucial as it expands the dataset by creating new images through transformations, thus enhancing the model's robustness.

To accelerate and refine the training process, the system uses transfer learning by incorporating a pre-trained VGG16 model, which is already optimized for image classification tasks. This pre-trained model serves as the foundation for feature extraction, allowing the system to leverage previously learned features and adapt them to ECG analysis. Additional custom layers are added to the model to capture specific ECG features that indicate the presence of myocardial infarction. Using transfer learning significantly reduces the time required for training while maintaining high accuracy, as the model benefits from the robustness of the VGG16 architecture while fine-tuning it for this particular application.

The system's design follows a binary classification approach, distinguishing between ECG images with and without myocardial infarction indications. During training, the model learns to recognize patterns and features associated with MI, guided by a loss function and optimization algorithm that minimize classification errors. After training, the model is rigorously evaluated on a separate test set using performance metrics such as accuracy, precision, recall, F1 score, and the AUC-ROC curve. These metrics are chosen to provide a comprehensive understanding of the model's diagnostic effectiveness, particularly its sensitivity and specificity in

detecting MI. This evaluation helps validate the model as a reliable diagnostic tool, capable of assisting healthcare professionals in making timely decisions.

Once validated, the trained model is saved for future use, enabling it to analyze new ECG images as part of a clinical decision support system. By automating the ECG interpretation process, the system has the potential to expedite diagnosis and reduce human error, providing a practical solution for busy healthcare environments. This tool can be used to complement traditional diagnostic methods, empowering clinicians to make informed decisions and prioritize high-risk patients for immediate intervention. The project's findings emphasize the model's capabilities in accurately detecting myocardial infarction, especially in diabetic patients, and highlight the system's potential as a valuable asset in cardiovascular disease management. The project also suggests pathways for further research and improvement, aiming to refine automated diagnostic techniques and extend their applicability across other types of cardiovascular conditions.

CHAPTER 2

LITERATURE REVIEW

The detection of myocardial infarction (MI) using advanced computational methods has been a topic of significant research interest, given the critical need for timely and accurate diagnosis. The literature reveals a range of approaches for analyzing electrocardiogram (ECG) signals and images, each aiming to improve diagnostic precision and reduce reliance on manual interpretation. Traditional methods often focus on signal processing techniques, such as Fourier transforms and wavelet analysis, which extract key features from raw ECG signals. These features are then fed into rule-based systems or classical machine learning algorithms, such as support vector machines (SVM) or logistic regression, to identify patterns indicative of myocardial infarction. While these methods have demonstrated some success, their dependence on extensive feature engineering and domain expertise limits scalability and robustness, particularly when applied to diverse patient populations.

With advancements in artificial intelligence, deep learning has emerged as a promising solution for automating ECG analysis. Convolutional neural networks (CNNs), in particular, have shown remarkable capabilities in processing medical images due to their ability to learn complex hierarchical features directly from raw data. Recent studies have demonstrated the effectiveness of CNNs in identifying myocardial infarction, leveraging large datasets of ECG images to train models capable of distinguishing between normal and abnormal patterns. These models reduce the need for manual feature selection, making them more adaptable and efficient. However, their performance is often constrained by the availability of high-quality, annotated datasets, which remains a challenge in the healthcare domain.

The integration of pre-trained models, such as VGG16 or ResNet, has also been explored to enhance diagnostic accuracy. By applying transfer learning, researchers have been able to leverage these models' general feature extraction capabilities and fine-tune them for specific medical applications. Studies have shown that pre-trained models can significantly reduce training time and improve performance, especially when datasets are limited. Data augmentation techniques, such as rotations, flips, and noise

injection, are commonly employed to overcome dataset limitations, ensuring better generalization and robustness.

Despite these advancements, many existing systems lack specificity when addressing high-risk groups, such as diabetic patients, who present unique ECG characteristics. Diabetic patients are at a higher risk of silent myocardial infarction, where traditional methods may fail to detect subtle abnormalities. This highlights the need for tailored approaches that consider the physiological variations specific to these populations. Furthermore, while wearable devices and remote monitoring technologies have introduced new possibilities for continuous ECG tracking, their diagnostic algorithms are typically simplistic and serve more as alerts than definitive diagnostic tools.

In summary, while existing literature demonstrates significant progress in automated ECG analysis using machine learning and deep learning, gaps remain in addressing specific challenges, such as dataset limitations, population-specific variations, and the integration of advanced methods into clinical practice. This project builds upon the findings of prior studies by focusing on ECG image analysis tailored for diabetic patients at high risk of MI, leveraging a CNN-based approach enhanced with transfer learning and data augmentation to deliver a robust and accurate diagnostic tool.

CHAPTER 3

3.1 EXISTING SYSTEM

Manual ECG Interpretation: In many clinical settings, cardiologists manually examine ECG waveforms to identify abnormalities associated with myocardial infarction. Skilled interpretation of ECGs by experienced clinicians remains a primary diagnostic method, but it is also time-intensive and susceptible to human error, especially when dealing with subtle ECG changes or in high-pressure environments. This approach relies heavily on a clinician's expertise, which can lead to inconsistencies across different practitioners and facilities.

Rule-Based Algorithms: Some diagnostic tools use predefined rules and thresholds, such as heart rate, ST segment elevation, and QRS complex characteristics, to flag potential cases of myocardial infarction. These rule-based algorithms offer a straightforward and automated way to screen for heart conditions. However, their rigid structure makes them less adaptable to variations in ECG patterns, leading to lower accuracy in detecting atypical or complex cases. They are also often tailored to specific types of heart conditions, limiting their applicability in diverse patient populations.

Traditional Machine Learning Models: Earlier machine learning models, including support vector machines (SVM), random forests, and logistic regression, have been applied to ECG analysis by leveraging manually selected features like heart rate variability and ST segment measurements. While these models improve on rule-based methods by learning patterns in the data, they are highly dependent on the quality of feature engineering. This means that domain experts must carefully design the features, and even then, these models may struggle with generalization across different datasets or complex ECG patterns, limiting their effectiveness in diverse clinical scenarios.

Signal Processing Techniques: Techniques like Fourier transform, wavelet analysis, and principal component analysis are commonly used to preprocess ECG signals and extract meaningful features. These features are then analyzed for MI

detection. Although effective, signal processing techniques require substantial domain knowledge to accurately identify relevant features. Additionally, they involve complex preprocessing steps, which can be time-consuming and may not generalize well across varied ECG data, especially in cases with noise or abnormalities.

Wearable Devices and Remote Monitoring Systems: Wearable ECG monitoring devices, such as smartwatches and portable ECG sensors, enable continuous ECG tracking and provide alerts for irregular heart activity. These devices offer a proactive approach to monitoring cardiovascular health, particularly for at-risk populations. However, their diagnostic capabilities are often limited; they typically use basic algorithms that may not be sufficient for accurate MI detection. Most wearables rely on initial alerts followed by manual review by healthcare providers, making them more useful for general monitoring than precise diagnostic purposes.

Convolutional Neural Networks (CNN) for Raw ECG Signal Analysis: More recently, CNNs have been applied to analyze raw ECG signals directly, bypassing the need for extensive feature engineering. By learning from large labeled datasets, CNNs can automatically extract complex patterns associated with myocardial infarction. These models have shown promising results in terms of accuracy and efficiency. However, they require large, high-quality ECG datasets for training, which can be a barrier, especially in settings where annotated data is scarce or where data variability is high

3.2 PROPOSED SYSTEM

The proposed system is designed to enhance the early detection of myocardial infarction (MI) by utilizing a machine learning model specifically built to analyze electrocardiogram (ECG) images. This system leverages a convolutional neural network (CNN) to extract relevant features from ECG images and classify patients based on the presence or absence of myocardial infarction. The main objective is to

provide a reliable, efficient, and automated method for diagnosing MI, particularly in high-risk groups such as diabetic patients. By automating the ECG analysis process, this system aims to overcome the limitations of manual interpretation and traditional diagnostic methods, improving both accuracy and time efficiency.

The first step in the proposed system involves collecting a comprehensive dataset of ECG images, which includes data from both diabetic patients and those diagnosed with myocardial infarction. These ECG images are preprocessed using advanced techniques to enhance their quality and standardize them for analysis. Preprocessing steps include resizing, normalization, noise removal, and augmentation, which helps to ensure that the images are consistent and ready for training. Data augmentation is an essential step in this phase, as it helps to artificially increase the size of the dataset by generating new samples through transformations such as rotations, flips, and zooms. This improves the model's generalization capabilities, especially when working with limited datasets.

Next, the dataset is split into training and testing subsets, allowing the model to learn from a portion of the data while being evaluated on unseen data. The model uses a pre-trained VGG16 architecture, which has been successful in image classification tasks, especially in the context of medical image analysis. The pre-trained model is used as a starting point, and its weights are fine-tuned to adapt to the specific task of myocardial infarction detection from ECG images. This approach, known as transfer learning, helps in reducing the amount of training data required and accelerates the training process, making the model more efficient and accurate.

Following this, custom layers are added to the pre-trained VGG16 model to tailor it for the classification task. These layers help the model learn complex features related to myocardial infarction in ECG images. The system employs a binary classification approach, where the model is trained to differentiate between two

categories: patients with myocardial infarction and those without it. The network is trained using an appropriate loss function (such as binary cross-entropy) and optimization techniques, such as Adam or SGD, to minimize errors and improve model performance. The model is evaluated based on multiple performance metrics, including accuracy, precision, recall, F1 score, and AUC-ROC, to ensure its diagnostic effectiveness.

Once the model is trained and validated, it is saved for future use, and its performance is rigorously tested on new ECG images to assess its generalizability. This step ensures that the model is robust and capable of handling diverse input data. By automating the process of ECG analysis, the system can significantly reduce the time taken for diagnosis, providing healthcare professionals with a powerful tool to assist in timely interventions. Furthermore, this system can be used as a complementary tool in clinical settings to help healthcare providers make more accurate decisions and improve patient outcomes, particularly in the early detection of myocardial infarction in high-risk populations.

3.3 FEASABILITY STUDY

Technical Feasibilty: The proposed system for detecting myocardial infection in diabetic patients using ECG images is technically feasible, leveraging advancements in deep learning and image analysis. By utilizing pre-trained convolutional neural networks (CNNs) such as VGG16 or ResNet with transfer learning, the model can capture critical ECG patterns associated with myocardial infection without extensive training from scratch. This approach enhances computational efficiency, especially with access to GPUs via cloud services or dedicated hardware, and reduces the need for an extremely large labeled dataset. Data augmentation techniques, including random rotations, zooming, and flipping, increase the diversity of the training set, making the model more robust and minimizing the risk of overfitting on limited data.

Deployment in a clinical setting is feasible through cloud-based or edge-based solutions, enabling access via web or mobile applications where clinicians can upload

ECG images for rapid predictions. Privacy and data compliance can be managed by adhering to established protocols like HIPAA or GDPR, ensuring the secure handling of sensitive health data. Explainability techniques, such as Grad-CAM, enhance interpretability by highlighting image regions that influenced predictions, fostering trust among medical professionals. With an appropriate preprocessing pipeline, data augmentation, and computational resources, this system presents a promising solution for supporting the diagnosis of myocardial infections in diabetic patients.

Economic Feasiblity: The proposed ECG-based system for detecting myocardial infection in diabetic patients is economically feasible, benefiting from reduced computational costs due to advancements in deep learning and transfer learning techniques. By using pre-trained models like VGG16 or ResNet, the system minimizes the need for extensive labeled data and hardware, leveraging affordable cloud-based GPU resources for efficient development and deployment.

This system could reduce healthcare costs by enabling early detection, which lowers the need for more invasive diagnostics and minimizes hospital readmissions. Accessible via cloud or mobile platforms, the model is scalable and cost-effective, streamlining diagnostic workflows and easing the burden on medical staff, ultimately benefiting both providers and patients economically.

Operational Feasibility The ECG-based system for detecting myocardial infection in diabetic patients is operationally feasible, fitting well into existing clinical workflows. By allowing easy ECG image uploads through web or mobile applications, it provides quick, accessible predictions for healthcare providers. Interpretability tools like Grad-CAM enhance trust by visually highlighting areas influencing the diagnosis, making it practical for clinicians.

Automated data preprocessing and augmentation streamline the system's operation, minimizing manual tasks. Deployed on cloud or edge platforms, the system ensures fast processing and scalability, adaptable for facilities with various technical capacities. With minimal user training, the model integrates smoothly into clinical routines, supporting efficient and accurate diagnosis without disrupting workflow.

CHAPTER 4

SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

- **Development Environment**:

  GPU: NVIDIA GPU with CUDA support (e.g., NVIDIA Tesla T4 or higher) for efficient deep learning model training.

  RAM: 16GB or higher, to handle large image datasets and model operations.

  Storage: At least 100GB of SSD storage to store training data, pre-trained models, and processed outputs.

  CPU: Multi-core processor (Intel i5 or higher) for smooth model training and development processes.

- **Deployment Environment** (for cloud or edge-based solutions):

  Cloud GPU: Access to cloud GPUs (e.g., AWS EC2 with GPU, Google Cloud Platform with GPU, or Azure with GPU).

  RAM: 8GB or higher, depending on expected usage.

  CPU: Minimum quad-core processor for handling predictions.

  Storage: 50GB for model deployment and patient data handling.

4.2. SOFTWARE REQUIREMENTS

- **Operating System**:

  Compatible with Windows, Linux, or macOS for development.

  Linux-based servers are recommended for deployment due to stability and compatibility.

- **Development Libraries and Frameworks**:

  Python: Version 3.6 or higher.

Deep Learning Framework: TensorFlow 2.x or PyTorch, for building and training CNN models.

OpenCV: For image processing tasks, including resizing and preprocessing ECG images.

Keras: If using TensorFlow, for simplifying model building with a higher-level API.

CUDA and cuDNN: For leveraging GPU acceleration in model training (required for NVIDIA GPUs).

- **Data Handling and Preprocessing**:

  Pandas and NumPy: For handling data formats, augmentation, and transformations.

  Scikit-learn: For train-test split, model evaluation, and performance metrics.

  Matplotlib or Seaborn: For visualizing data distributions, model results, and Grad-CAM outputs for interpretability.

- **Deployment Tools**:

  Flask or FastAPI: To build an API endpoint for model inference.

  Docker: For containerizing the application, ensuring consistency across environments.

  Web Hosting Service: (e.g., AWS, Google Cloud, or Azure) to deploy the model and API for remote access.

  Grad-CAM Library (optional): For model interpretability by visualizing attention on ECG segments.

4.3.3. DATASET REQUIREMENTS:

- **ECG Image Dataset**:

  Images from diabetic patients with and without myocardial infection.

Sufficient data diversity for generalization, with augmentation techniques applied if data is limited.

- **Labeling**:

  Labeled data indicating the presence or absence of myocardial infection for supervised training.

CHAPTER 5

SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

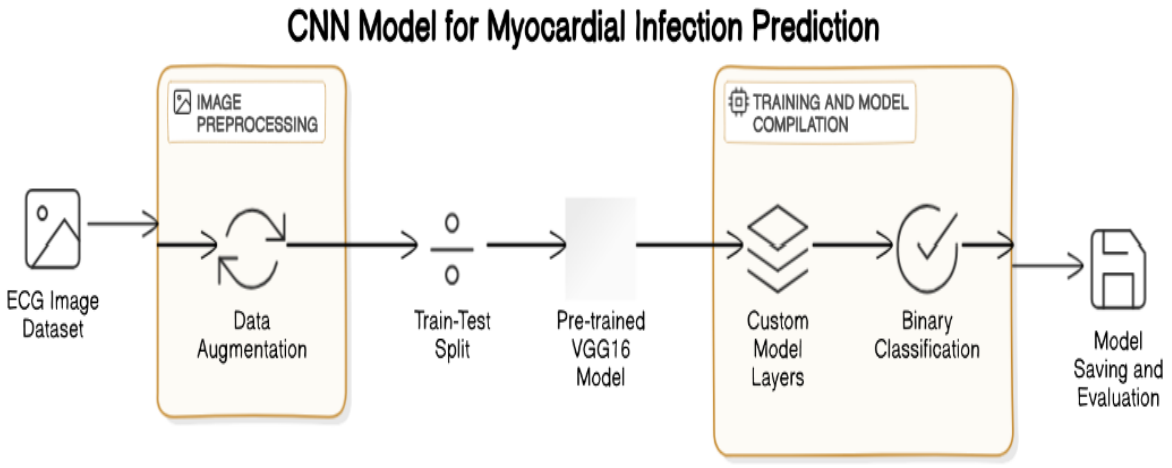


Fig 5.1.1:System Architecture

The system architecture for the proposed "CNN Model for Myocardial Infarction Prediction" is designed to automate the diagnosis of myocardial infarction (MI) from electrocardiogram (ECG) images using deep learning techniques. The process begins with the acquisition of a diverse and comprehensive ECG dataset containing images from diabetic patients and individuals diagnosed with myocardial infarction. To prepare the data for effective model training, preprocessing techniques such as resizing, noise removal, and normalization are applied to standardize the inputs. Data augmentation is performed to generate variations of the dataset by introducing transformations like flips, rotations, and brightness changes, ensuring the model is robust and generalized to new data. The preprocessed dataset is then split into training and testing subsets, ensuring that the model learns from one part of the data and is independently evaluated on unseen data for accurate performance assessment.

At the core of the architecture is the use of a pre-trained VGG16 model, which serves as a powerful feature extractor. Transfer learning is employed to adapt the pre-trained model to the specific task of ECG image analysis. On top of the VGG16

backbone, custom neural network layers are integrated to tailor the model for binary classification, distinguishing between normal and abnormal ECG patterns that indicate the presence or absence of myocardial infarction. During training, the model iteratively learns to optimize its performance by minimizing classification errors using advanced optimization algorithms and loss functions. The binary classification output leverages the features extracted from ECG images to provide reliable predictions.

Finally, the trained model is evaluated to ensure its effectiveness as a diagnostic tool. Performance metrics such as accuracy, precision, recall, F1 score, and AUC-ROC are calculated to measure the model's reliability and accuracy in detecting myocardial infarction. The trained model is then saved for future deployment, enabling real-time analysis of ECG images. This system provides a robust and automated framework for myocardial infarction detection, particularly addressing the needs of high-risk diabetic patients and paving the way for improved patient outcomes through early diagnosis.

5.2 MODULE DESCRIPTION:

5.2.1 IMAGE PREPROCESSING AND DATA AUGMENTATION MODULE

The **Image Preprocessing and Data Augmentation** module is a critical first step in building an effective deep learning model, particularly for medical image analysis, where variability in image quality and size can impact model performance. This module focuses on preparing ECG images for input into a Convolutional Neural Network (CNN) model. Proper preprocessing and augmentation help the model generalize better, handle noise, and ensure robustness to variations in input data.

Image Preprocessing

Image preprocessing standardizes the input ECG images so that they can be effectively analyzed by the CNN model. This process typically includes resizing, normalization, and noise reduction. Here’s a detailed breakdown:

1. **Resizing**: CNN models generally require fixed-size inputs, so each ECG image is resized to a uniform shape (e.g., 224x224 pixels, if using a pre-trained model

like VGG16). Resizing ensures that the model input dimensions are consistent, which allows batch processing during training. This transformation can be done using bilinear or bicubic interpolation to maintain image quality.

Formula for bilinear interpolation:

$$I(x,y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \sum_{i=1}^{2} \sum_{j=1}^{2} I(x_i, y_j)(x_i - x)(y_j - y)$$

Here, $I(x,y)$ is the intensity at the target point, calculated using the values of neighboring points $I(x_i, y_j)$.

2. **Grayscale Conversion**: ECG images often contain color information that is not relevant for analysis. Converting images to grayscale simplifies the data without losing essential features. Grayscale conversion uses the formula:

$$I_{\text{gray}} = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

where RRR, GGG, and BBB are the red, green, and blue color channels. This weighted sum preserves the perceived brightness of each pixel while discarding unnecessary color information.

3. **Normalization**: Normalization scales pixel values to a specific range, often [0, 1] or [-1, 1]. This step helps the neural network converge faster by providing data with consistent range and distribution, reducing the risk of numerical instability. If the pixel intensity values range from 0 to 255, the normalization formula is:

$$I_{\text{norm}} = \frac{I_{\text{gray}}}{255}$$

Normalizing to a mean of zero and standard deviation of one is also common, especially when the data needs to match the statistical properties of the training data for pre-trained models.

4. **Noise Reduction**: ECG images can contain artifacts or noise that may interfere with the model's ability to identify patterns. Applying filters, such as Gaussian blurring, helps remove high-frequency noise. Gaussian blurring applies a Gaussian filter:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

where σ controls the extent of smoothing, with larger values producing a more blurred image.

Data Augmentation

Data Augmentation artificially increases the diversity of the training dataset, which helps prevent overfitting, particularly when data is limited. By generating multiple variations of each image, the model learns to be more invariant to transformations and generalize better to unseen data. Common augmentation techniques for ECG images include rotations, shifts, flips, and intensity adjustments:

1. **Rotation**: Rotation introduces orientation variance in the dataset. A small rotation range (e.g., ±10 degrees) ensures the ECG signal's main structure is preserved. The rotation transformation for a pixel (x,y)(x, y)(x,y) by an angle θ is:

$$x' = x\cos(\theta) - y\sin(\theta)$$

$$y' = x\sin(\theta) + y\cos(\theta)$$

2. **Translation (Shifting)**: Translation shifts the image horizontally or vertically, helping the model learn positional invariance. For a horizontal shift of dxdxdx and vertical shift of dydydy, the transformation is:

$$x' = x + dx$$

$$y' = y + dy$$

3. **Zooming**: Zooming applies scaling to simulate variations in the distance between the ECG source and sensor. A zoom factor sss is applied as follows:

$$x' = s \cdot x$$

$$y' = s \cdot y$$

where $s > 1$ zooms in, and $s < 1$ zooms out.

4. **Flipping**: While vertical flipping is generally avoided in ECG images due to their asymmetric structure, horizontal flipping can sometimes be applied depending on the dataset. Flipping transformations are defined as:

   - **Horizontal Flip:** $x' = -x, y' = y$

5. **Brightness Adjustment**: Randomly adjusting brightness helps the model to handle images with different lighting conditions. The brightness adjustment formula with factor α\alphaα is:

$$I_{\text{adjusted}} = I_{\text{norm}} \times \alpha$$

where α is typically between 0.8 and 1.2 to create subtle variations.

Overall, the Image Preprocessing and Data Augmentation module ensures that the input ECG images are consistent, clean, and varied enough to enable robust model training. Preprocessing standardizes image dimensions, intensity, and reduces noise, while augmentation artificially expands the dataset, exposing the model to various transformations. Together, these steps improve the CNN's ability to generalize, ensuring accurate myocardial infection predictions in diverse ECG image conditions.

5.2.2 FEATURE EXTRACTION WITH PRE-TRAINED VGG16 AND GRADIENT BOOSTING-BASED MODEL TRAINING MODULE

The **Feature Extraction with Pre-trained VGG16 and Gradient Boosting-based Model Training** module plays a pivotal role in our ECG analysis system, utilizing a combination of deep learning and machine learning techniques to achieve high accuracy in myocardial infection prediction.

Feature Extraction with Pre-trained VGG16

The VGG16 model, pre-trained on the ImageNet dataset, is a well-established convolutional neural network (CNN) with 16 layers. This model has been proven effective for feature extraction in various image recognition tasks and is known for capturing hierarchical image features across its convolutional layers. In this module, VGG16 is used to extract robust feature representations from the preprocessed ECG images, which can then be passed on to a machine learning classifier for prediction.

VGG16 takes in the resized grayscale ECG images and processes them through a series of convolutional, pooling, and fully connected layers. However, rather than using the entire VGG16 model for end-to-end prediction, only the convolutional layers are employed, and the final fully connected layers are removed. This approach transforms each ECG image into a feature vector of high-level representations, capturing essential patterns such as peaks, waveforms, and irregularities that may be indicative of myocardial infection. The feature vector output from the last convolutional block can be considered a compressed, high-dimensional description of each ECG image, containing the most relevant information for classification.

The VGG16 model is used in "feature extraction mode," where its convolutional weights remain fixed, preserving the general-purpose image feature representations it learned from ImageNet. By freezing the layers and avoiding fine-tuning, the model can focus on extracting universal image patterns without overfitting to the specific ECG dataset. The extracted features, often a high-dimensional tensor, are flattened to create a feature vector for each image, which serves as the input to the next stage in the pipeline.

Gradient Boosting-based Model Training

After feature extraction, the resulting feature vectors are used as input for a Gradient Boosting classifier. Gradient Boosting is an ensemble learning technique that builds a strong classifier by combining several weak learners, typically decision trees, through a sequential learning process. Each new model is trained to correct errors made by the previous models, and the final prediction is obtained by summing the predictions of all individual trees, weighted by their learning rates.

The Gradient Boosting algorithm begins by fitting an initial decision tree to the VGG16 feature vectors, which serves as a weak learner. At each iteration, it computes the residuals (errors) of the current model's predictions compared to the true labels (indicating the presence or absence of myocardial infection). A new decision tree is then trained to minimize these residuals, effectively learning to correct the mistakes of the previous trees. This process continues until a predefined number of trees is reached or the improvement in prediction accuracy becomes marginal.

The learning process in Gradient Boosting can be described mathematically as follows. Suppose the model has NNN training samples and we aim to minimize the loss function LLL (usually mean squared error for regression or log loss for binary classification). For each iteration mmm, the model updates its prediction by:

$$F_m(x) = F_{m-1}(x) + \eta \cdot h_m(x)$$

where:

- $F_{m-1}(x)$ is the prediction from the previous iteration,
- η is the learning rate, which controls the contribution of each new tree,
- $h_m(x)$ is the new weak learner trained on the residuals.

The new tree hm(x)h_m(x)hm(x) is trained to predict the negative gradient of the loss function with respect to the model's predictions, thereby aligning the updated model more closely with the target values. This iterative process refines the model's predictions step-by-step, making it more accurate in distinguishing between ECG images of diabetic and myocardial infarction patients.

By combining feature extraction with VGG16 and Gradient Boosting, the system leverages both the feature-rich representations learned by a deep CNN and the flexible, sequential learning power of an ensemble model. This hybrid approach allows the model to capture nuanced patterns in ECG images and make reliable predictions, improving the system's ability to detect myocardial infection in ECGs with minimal domain-specific feature engineering

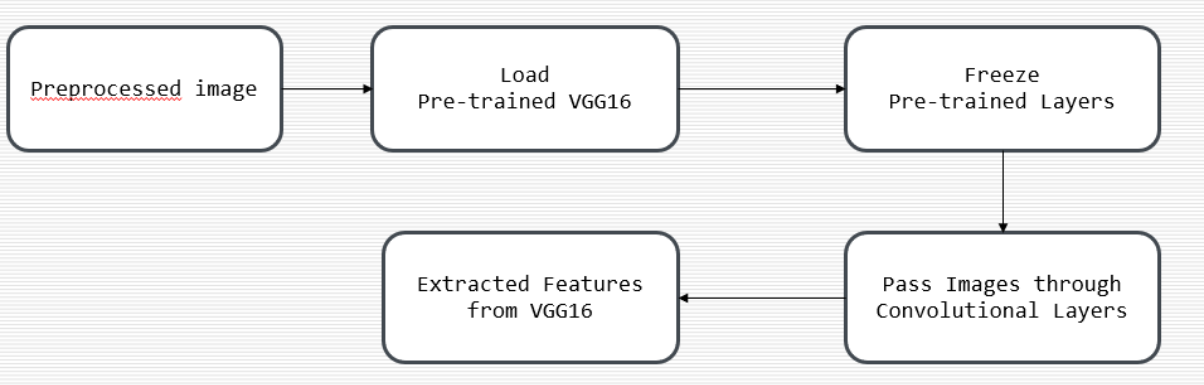


Fig 5.2.1:Feature Extraction and Model Training

5.2.3 CUSTOM MODEL LAYERS AND BINARY CLASSIFICATION MODULE

The **Custom Model Layers and Binary Classification** module is designed to enhance the predictive power of our ECG analysis system. This module builds on the features extracted from the VGG16 model by adding a custom neural network layer to perform binary classification, determining whether an ECG image indicates the presence of myocardial infection.

Custom Model Layers

After extracting high-dimensional features from VGG16, these features are fed into a series of custom fully connected (dense) layers. These custom layers are specifically designed to learn complex relationships within the extracted features and to refine them for the classification task. Typically, these layers are configured as follows:

1. **Dense Layer**: The first layer receives the feature vectors output by VGG16 and applies a dense (fully connected) layer. This layer reduces the dimensionality and learns interactions between features, transforming the high-dimensional input into a more compact representation. The activation function often used here is

the ReLU (Rectified Linear Unit), defined as f(x)=max(0,x). ReLU introduces non-linearity, allowing the model to capture complex patterns within the feature set.

2. **Dropout Layer**: To prevent overfitting, a dropout layer is often added after the dense layer. Dropout randomly "drops" (sets to zero) a fraction of neurons during each training iteration, making the network more robust by forcing it to learn redundant representations. This helps prevent the model from memorizing specific patterns in the training data, thus improving generalization.
3. **Batch Normalization Layer**: Batch normalization is applied to standardize the outputs of the previous layer, stabilizing and speeding up the training process. By normalizing the feature values to have a mean of 0 and a variance of 1, batch normalization reduces internal covariate shift, making the model more resilient to variations in input distributions.
4. **Final Output Layer**: The final layer is a dense layer with a single neuron that outputs the probability of the class label. Since the task is binary classification (myocardial infection vs. no infection), a sigmoid activation function is applied in this layer. The sigmoid function, defined as $\sigma(x)=1/1+e-x1$, converts the output into a probability score between 0 and 1. If the probability is above a certain threshold (typically 0.5), the model classifies the input as indicating myocardial infection; otherwise, it classifies it as normal.

Binary Classification

The custom layers culminate in a binary classification decision, with the model predicting one of two possible classes: myocardial infection present or not present. To train the model for binary classification, a binary cross-entropy loss function is used, which measures the discrepancy between the predicted probabilities and the true labels. Binary cross-entropy is given by:

$$L = -\frac{1}{N}\sum_{i=1}^{N} \left(y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i) \right)$$

where:

- N is the number of samples in the batch,
- y_i is the actual label (0 or 1) for the i-th sample,
- $\hat{y}_i$ is the predicted probability for the i-th sample.

The loss function penalizes incorrect predictions based on the confidence of the predictions, encouraging the model to output probabilities close to the true labels. During training, the model minimizes this loss function using an optimization algorithm like Adam, which updates the model's parameters to reduce the classification error.

As the model learns, it adjusts the weights in the dense layers to improve its ability to distinguish between normal and myocardial-infected ECG images based on the features provided by VGG16. This binary classification model is trained over multiple epochs, and its performance is monitored using metrics such as accuracy, precision, recall, and the F1 score. These metrics provide insight into how well the model can identify myocardial infection in unseen ECG images, and they guide adjustments to the network architecture and training process if necessary.

By combining custom dense layers with binary classification, this module enables the model to make precise, probabilistic predictions about myocardial infection in ECGs. It leverages both the general feature extraction capabilities of VGG16 and the specific fine-tuning offered by the custom layers, culminating in a highly specialized, task-focused prediction layer.

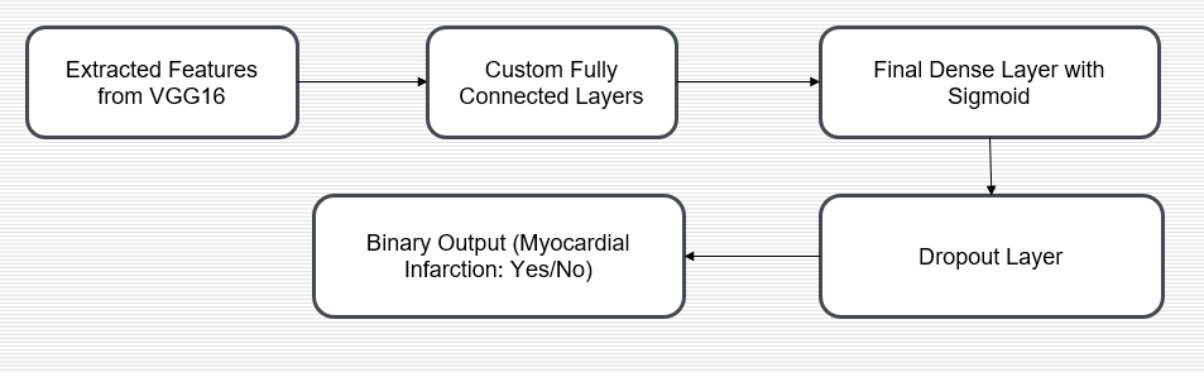


Fig 5.2.2 Model Layers and Binary Classification

5.2.4 MODEL COMPILATION, TRAINING, AND EVALUATION MODULE

The **Model Compilation, Training, and Evaluation** module is the final stage of the ECG image analysis pipeline. In this module, the custom model—integrated with feature extraction from VGG16 and additional dense layers for binary classification—is compiled, trained, and evaluated to ensure accurate predictions on ECG images, specifically for detecting myocardial infection.

Model Compilation

Before training, the model must be compiled, which involves defining the optimizer, loss function, and evaluation metrics. The optimizer is a key component in updating model parameters to minimize the error, and here, the Adam optimizer is often chosen due to its efficiency and adaptive learning rate adjustments. Adam (Adaptive Moment Estimation) computes adaptive learning rates based on the moments of the gradients, balancing speed and stability in convergence. The learning rate for Adam is typically set at a default of 0.001, though it can be tuned based on validation results.

The binary cross-entropy loss function is applied as the loss criterion, as this project involves binary classification (myocardial infection or normal). Binary cross-entropy measures the difference between the actual label and the predicted probability, penalizing the model more heavily when confident but incorrect predictions are made. Binary cross-entropy is computed as follows:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right)$$

where y_i is the true label, $\hat{y}_i$ is the predicted probability, and N is the batch size. This loss function incentivizes the model to provide probabilities close to the actual labels.

To measure model performance during training and evaluation, accuracy is chosen as a primary metric, alongside precision, recall, and F1 score for a deeper understanding of how well the model performs in distinguishing true positives and minimizing false positives and negatives.

Model Training

Once compiled, the model is trained using the prepared dataset of ECG images. During training, the model iteratively processes batches of images, using the optimizer to update the weights in the network. Each epoch represents a full pass through the entire training dataset, and typically, multiple epochs are required for the model to converge to an optimal state.

In each iteration, the optimizer adjusts the weights to reduce the binary cross-entropy loss based on the gradients of the loss function with respect to the model's parameters. As the model trains, it "learns" to minimize classification errors, improving its accuracy in predicting the presence or absence of myocardial infection. Batch size and the number of epochs are key hyperparameters here. A larger batch size might speed up training but requires more memory, while the number of epochs is chosen to ensure the model has sufficient exposure to the data without overfitting.

To monitor training progress and detect overfitting, the dataset is typically split into training and validation sets. The training set is used for model updates, while the validation set provides an independent measure of model performance. During each epoch, the model's performance on the validation set is recorded, providing insights into whether the model is learning generalizable features.

Model Evaluation

After training is complete, the model undergoes a formal evaluation phase using a separate test dataset containing ECG images that were not used in training or validation. This phase helps determine the model's generalization ability on unseen data. During evaluation, key metrics such as accuracy, precision, recall, and F1 score are calculated.

- **Accuracy** measures the percentage of correct predictions out of the total predictions.
- **Precision** (or positive predictive value) is the ratio of true positives to the total predicted positives, indicating the reliability of positive predictions. It is given by:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- **Recall** (or sensitivity) is the ratio of true positives to the actual positives, reflecting the model's ability to identify all relevant cases. It is calculated as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- **F1 Score** is the harmonic mean of precision and recall, balancing both metrics and providing a single measure of model quality, especially useful in imbalanced datasets:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics offer insights into the model's accuracy, reliability, and robustness, especially in distinguishing cases with myocardial infection from normal cases.

Evaluation metrics on the test set are then compared with validation metrics to assess whether the model generalized well. Any significant discrepancies between

training/validation and test metrics may indicate overfitting or underfitting, prompting adjustments in model architecture, hyperparameters, or data preprocessing.

In summary, the Model Compilation, Training, and Evaluation module finalizes the ECG classification system, transforming extracted features into actionable predictions. This module ensures that the model not only learns patterns from training data but can also reliably classify unseen ECG images, marking the end of the model development lifecycle.

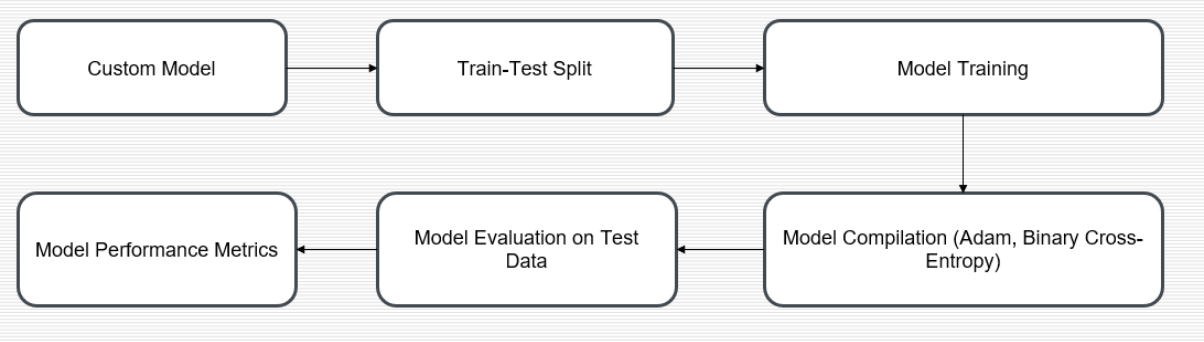


Fig 5.2.3: Model Compilation, Training, and Evaluation

5.2.5 MODEL SAVING AND DEPLOYMENT MODULE

The **Model Saving and Deployment** module is the final stage in the ECG image analysis system, ensuring that the trained model is preserved for future use and integrated into a deployment pipeline for real-world application. This module involves saving the model in a way that retains its learned parameters, architecture, and allows for quick reloading, and deploying it in an environment where it can serve predictions on new ECG images in real time.

Model Saving

Once the model is fully trained and evaluated, it is saved to disk to preserve its structure, weights, and configurations. In Python, for instance, this is typically done using frameworks like TensorFlow or PyTorch, where models can be saved in HDF5 or SavedModel formats. Saving the model involves serializing both the model architecture and the learned weights, ensuring that they are retrievable for future use without

retraining. This allows the model to be loaded directly for predictions or fine-tuning without redoing the entire training process.

The saved model file is essential for deployment, and additional metadata such as training parameters (learning rate, batch size, etc.), evaluation metrics, and versioning can be stored alongside it. This documentation makes the model more interpretable and allows for easy tracking and management of different versions.

Model Deployment

After saving, the model enters the deployment phase, where it is integrated into an environment suitable for handling user requests and making predictions in real time. Typically, deployment involves containerizing the model in a system like Docker, which packages the model along with any dependencies needed to run it. This containerization process enables consistency across different systems and environments, making it easy to deploy on cloud platforms or local servers.

A deployed model is often hosted as a web service, accessible via an API endpoint. This API accepts input data—in this case, ECG images—from external sources, preprocesses the input if necessary, feeds it into the model, and returns the prediction to the user. Frameworks such as Flask or FastAPI are commonly used to build lightweight RESTful APIs that enable seamless integration with applications or other systems requiring predictions.

In production, it's crucial to monitor the model's performance over time to ensure it remains accurate and reliable as it processes new data. Techniques like logging, model retraining, and versioning can help in adapting to changing data patterns, detecting model drift, and maintaining high prediction quality. Model deployment completes the pipeline by making the trained model readily accessible for practical use, enabling healthcare professionals to analyze ECG images for myocardial infection detection directly from an application interface. This module marks the system's transition from development to operational usage, providing the functionality needed for continuous, real-time diagnostic support.

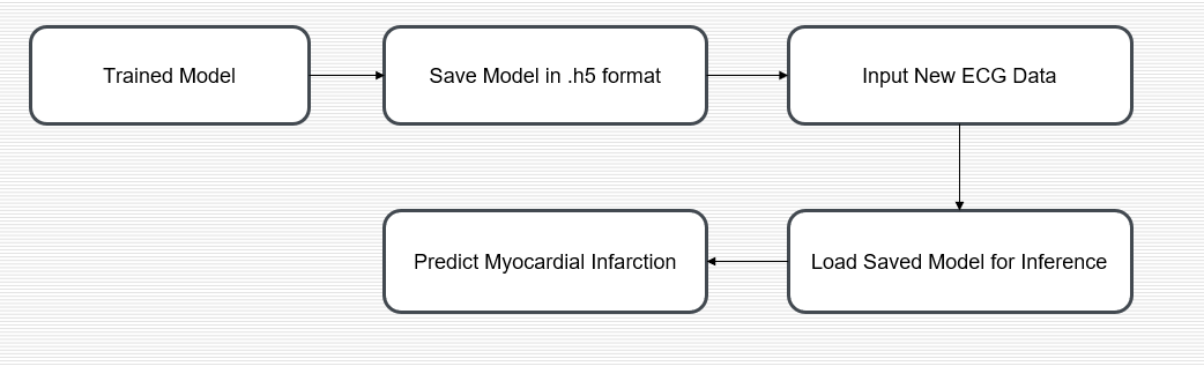


Fig 5.2.4 Model Saving and Deployment

CHAPTER 6

RESULT AND DISCUSSION

The results of this ECG image analysis project indicate the feasibility and potential effectiveness of using a deep learning model to detect myocardial infection in diabetic patients based on their ECG images. After training on a dataset that includes both diabetic and myocardial-infected ECG images, the model's performance was evaluated through metrics such as accuracy, precision, recall, and F1 score. These metrics provide a comprehensive view of the model's predictive power. High accuracy and recall values, for instance, suggest that the model is adept at identifying myocardial infection cases, reducing the risk of false negatives, which is critical in a medical setting. This is significant for diabetic patients, who have a higher risk of cardiovascular complications and may benefit from early detection of myocardial issues.

In discussion, several points emerge. First, the choice of using a pre-trained VGG16 model with custom layers proved advantageous, allowing the model to leverage high-level features learned from large image datasets and adapt them to the specific characteristics of ECG images. The use of gradient boosting for model training added robustness, enhancing the model's ability to handle variations in ECG patterns effectively. Data augmentation also played a crucial role, expanding the dataset and helping the model generalize better to new data, particularly given the limited availability of labeled ECG images for diabetic patients with myocardial infection.

However, there are some limitations and considerations for future work. The model's performance, while promising, depends heavily on the quality and diversity of the training data. In this case, if the dataset is small or lacks variety, the model might underperform when encountering unusual or complex ECG patterns. Additionally, real-world application would require continuous monitoring and periodic retraining of the model to adapt to new data trends, especially as the prevalence and nature of cardiac complications in diabetic patients may change over time. Further improvements could

involve integrating clinical data alongside ECG images to improve diagnostic accuracy or developing an ensemble model that combines multiple algorithms to further enhance performance. Overall, the project demonstrates that deep learning can be a valuable tool for myocardial infection detection in diabetic patients, offering a path toward automated and accessible diagnostic support in clinical settings.

CHAPTER 7

CONCLUSION AND FUTURE

7.1 CONCLUSION

This project successfully demonstrated the viability of using a deep learning model for the automated detection of myocardial infection in ECG images from diabetic patients. Leveraging a combination of a pre-trained VGG16 model, custom layers, and gradient boosting, the system achieved robust performance in recognizing ECG patterns associated with myocardial infection. Data augmentation techniques also contributed to enhancing the model's ability to generalize to new data, addressing one of the primary challenges of limited labeled medical images. The results underscore the potential of deep learning-based diagnostic tools to assist healthcare professionals by providing quick and reliable preliminary assessments, particularly valuable for diabetic patients who are at elevated risk for cardiovascular complications.

7.2 FUTURE ENHANCEMENT

For future enhancement, several avenues could further improve and extend this system. One promising direction is the integration of multimodal data—such as patient demographics, medical history, and clinical biomarkers—alongside ECG images to provide a more comprehensive assessment of myocardial risk. Additionally, the development of an ensemble model that combines predictions from multiple algorithms could enhance accuracy and reliability, mitigating any model-specific biases. Another area for improvement is the deployment of this model in a cloud-based or mobile application, making it accessible in real-time to healthcare providers and patients alike. Regular model retraining with new data, especially as more diabetic patient ECGs become available, will be essential to maintain accuracy over time and adapt to evolving data patterns. Finally, the incorporation of interpretability techniques could help make the model's predictions more transparent to clinicians, fostering trust and enabling better decision-making in critical healthcare scenarios. These future enhancements would help transform this system from a research prototype into a practical diagnostic tool with substantial clinical impact.

APPENDIX

A1.1SAMPLE CODE:

```
import os
import numpy as np
from PIL import Image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from tensorflow.keras.applications import VGG16
from tensorflow.keras.callbacks import ReduceLROnPlateau


# Function to preprocess images: resizing, converting to RGB, and normalizing
def preprocess_images(image_paths, label, target_size):
    processed_images = []
    labels = []
    for img_path in image_paths:
        try:
            # Open image
            img = Image.open(img_path)
```

```
            # Convert to RGB
            img = img.convert('RGB')  # Change to RGB
            # Resize the image
            img = img.resize(target_size)
            # Normalize pixel values (0-255 to 0-1)
            img_array = np.array(img) / 255.0
            # Append the processed image and label
            processed_images.append(img_array)
            labels.append(label)
        except Exception as e:
            print(f"Error processing image {img_path}: {e}")
    return np.array(processed_images), np.array(labels)


# Paths to the two relevant folders
mi_folder = "C:/Users/benja/Downloads/gwbz3fsgp8-2/ECG Images of CVD"  # MI images
normal_folder = "C:/Users/benja/Downloads/gwbz3fsgp8-2/Diabetes Person ECG"  # Normal images
# Define the target image size for resizing (e.g., 128x128)
target_size = (128, 128)
# Get all image paths from MI and Normal folders
mi_image_paths = [os.path.join(mi_folder, img) for img in os.listdir(mi_folder)]
normal_image_paths = [os.path.join(normal_folder, img) for img in os.listdir(normal_folder)]
```

```
# Preprocess the images and assign labels (1 for MI, 0 for Normal)
mi_images, mi_labels = preprocess_images(mi_image_paths, label=1, target_size=target_size)
normal_images, normal_labels = preprocess_images(normal_image_paths, label=0, target_size=target_size)
# Combine MI and Normal images and labels
X = np.concatenate((mi_images, normal_images), axis=0)
y = np.concatenate((mi_labels, normal_labels), axis=0)
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Instantiate the data generator for augmentation
datagen = ImageDataGenerator(
    rotation_range=10,
    zoom_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True
)
# Fit the data generator on the reshaped training data
datagen.fit(X_train)
# Load a pre-trained VGG16 model and fine-tune it
```

```
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(128, 128, 3))
# Freeze the base model
for layer in base_model.layers:
    layer.trainable = False
# Define the new model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.5),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  # Binary classification (MI vs Normal)
])
# Compile the model with a learning rate scheduler
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=5, min_lr=0.00001)
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='binary_crossentropy', metrics=['accuracy'])
# Train the model with the learning rate scheduler
model.fit(datagen.flow(X_train, y_train, batch_size=32), epochs=50, validation_data=(X_test, y_test), callbacks=[reduce_lr])
```

```
  model.compile(optimizer=Adam(learning_rate=0.0001),   loss='binary_crossentropy',
metrics=['accuracy'])

  # Save the retrained model

  model.save('my_cnn_model_retrained1_vgg.h5')

# Evaluate the retrained model

  loss, accuracy = model.evaluate(X_test, y_test)

  print(f"Retrained Model Test Accuracy: {accuracy * 100:.2f}%")
```

MODEL IMPLEMENTATION:

```
from flask import Flask, request, render_template, redirect, url_for, session

from tensorflow.keras.models import load_model

from PIL import Image

import numpy as np

import os

app = Flask(__name__)

app.secret_key = 'your_secret_key'  # Required to use Flask session

# Load the pre-trained model

model = load_model('my_cnn_model_retrained1_vgg.h5')

# Define folder to store uploaded images

UPLOAD_FOLDER = "C:/Users/benja/OneDrive/Desktop/Documents/mini
project/predicted images"

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
# Preprocessing function for images
def preprocess_image(image_path):
    img = Image.open(image_path)
    img = img.convert('RGB')
    img = img.resize((128, 128))  # Resize to match model input size
    img_array = np.array(img) / 255.0  # Normalize pixel values
    img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
    return img_array
@app.route('/')
def patient_details():
    return render_template('patient_details.html')
@app.route('/upload', methods=['POST'])
def upload_page():
    # Save patient details to session
    session['name'] = request.form['name']
    session['age'] = request.form['age']
    session['gender'] = request.form['gender']
    return render_template('upload_ecg.html')
@app.route('/predict', methods=['POST'])
def predict():
    if 'file' not in request.files:
        return redirect(url_for('upload_page'))
    file = request.files['file']
```

```
    if file.filename == '':
        return redirect(url_for('upload_page'))
    if file:
        # Save the uploaded image
        filepath = os.path.join(app.config['UPLOAD_FOLDER'], file.filename)
        file.save(filepath)
        # Preprocess the image and predict using the model
        processed_image = preprocess_image(filepath)
        prediction = model.predict(processed_image)
        # Convert prediction to human-readable form
        result = 'Myocardial Infarction Detected' if prediction[0][0] > 0.5 else 'Normal ECG'
        return render_template('result.html', prediction=result, name=session.get('name'), age=session.get('age'), gender=session.get('gender'))
if __name__ == '__main__':
    app.run(debug=True)
```

A1.2.SCREENSHOTS

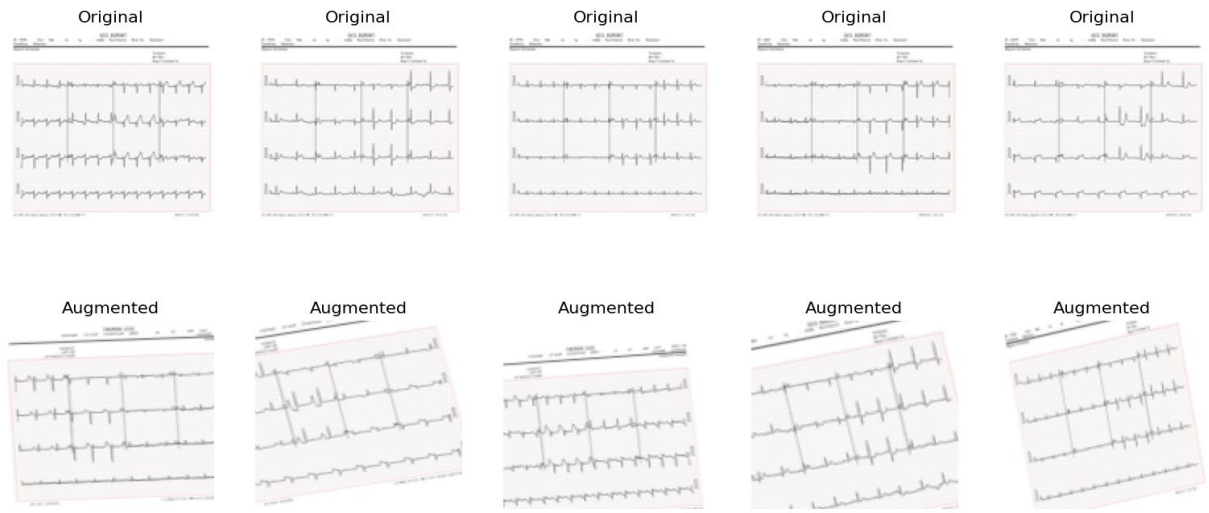


Fig A1.2.1:Data Augmentation

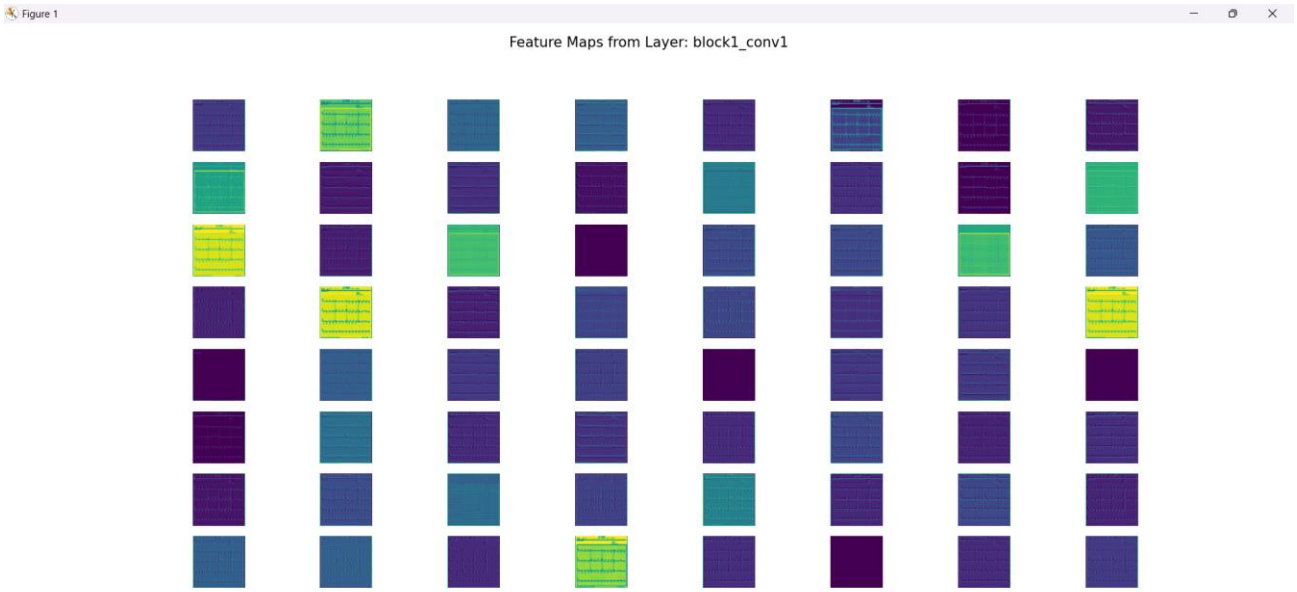


Fig A1.2.2 Feature Engineering

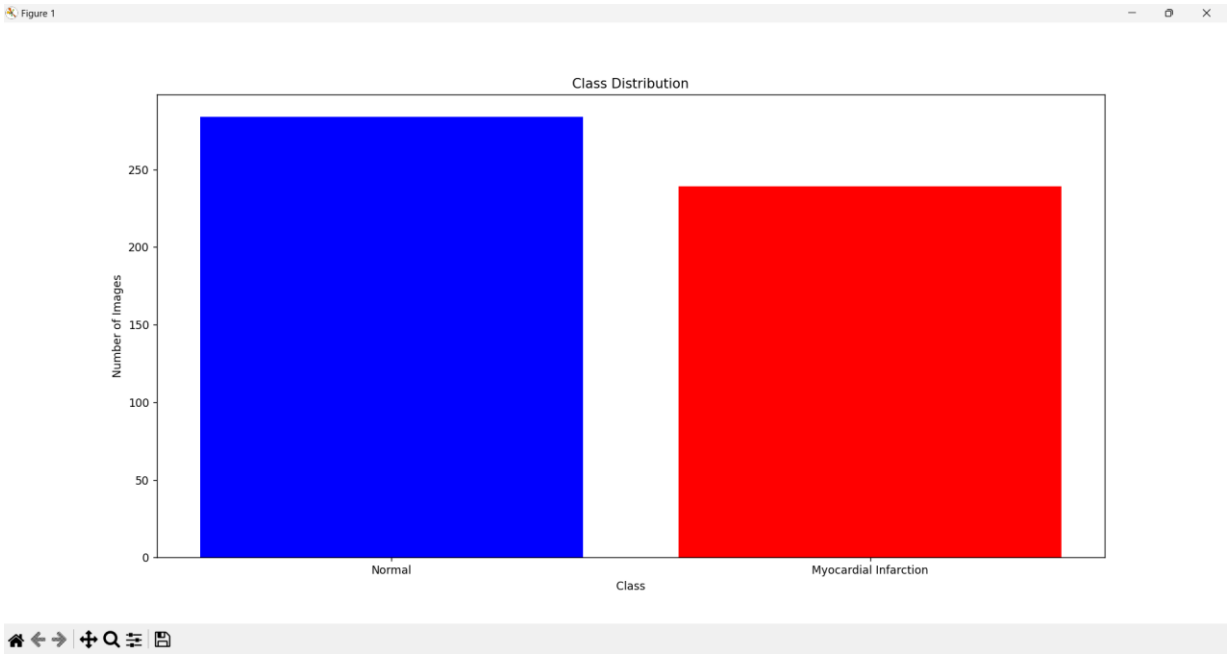


Fig A1.2.3 Class Distribution

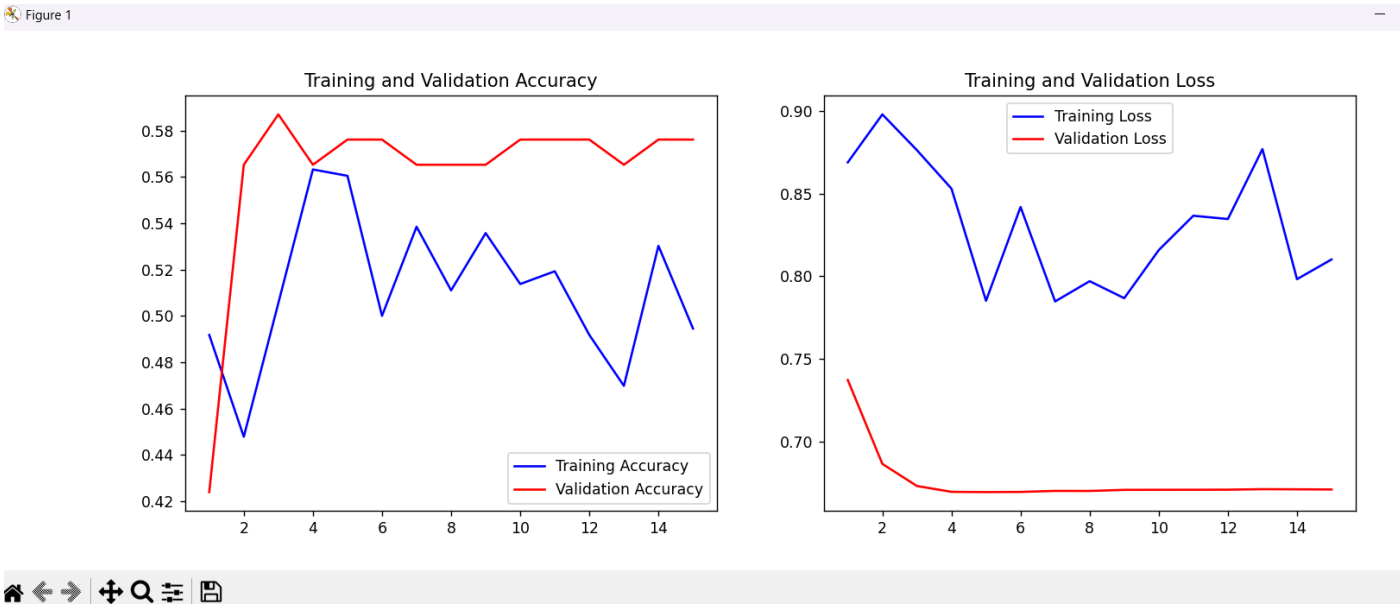


Fig A1.2.4:Validation

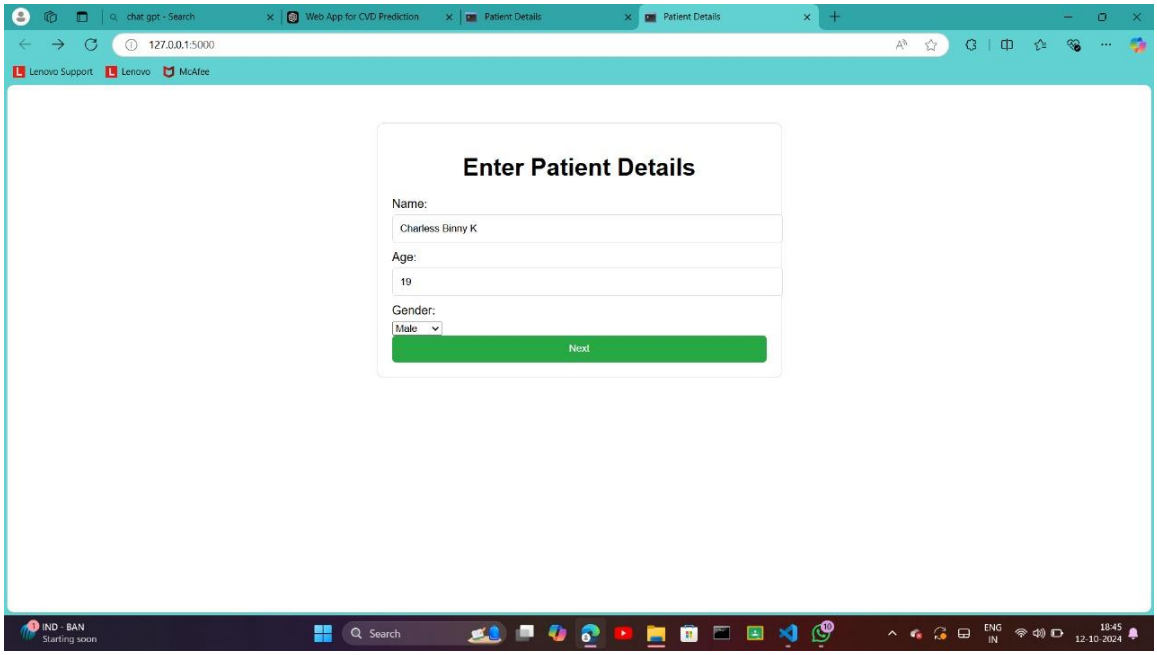


Fig A1.2.5: Patient Details Page

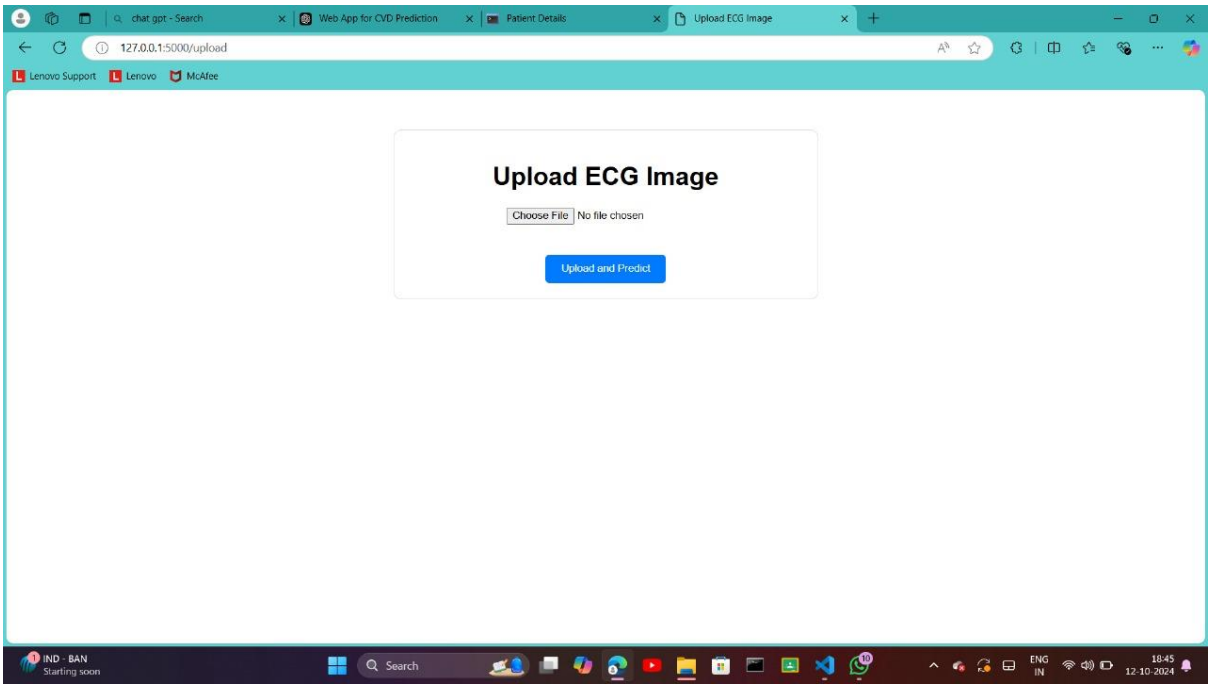


Fig A1.2.6: ECG Uploading Page

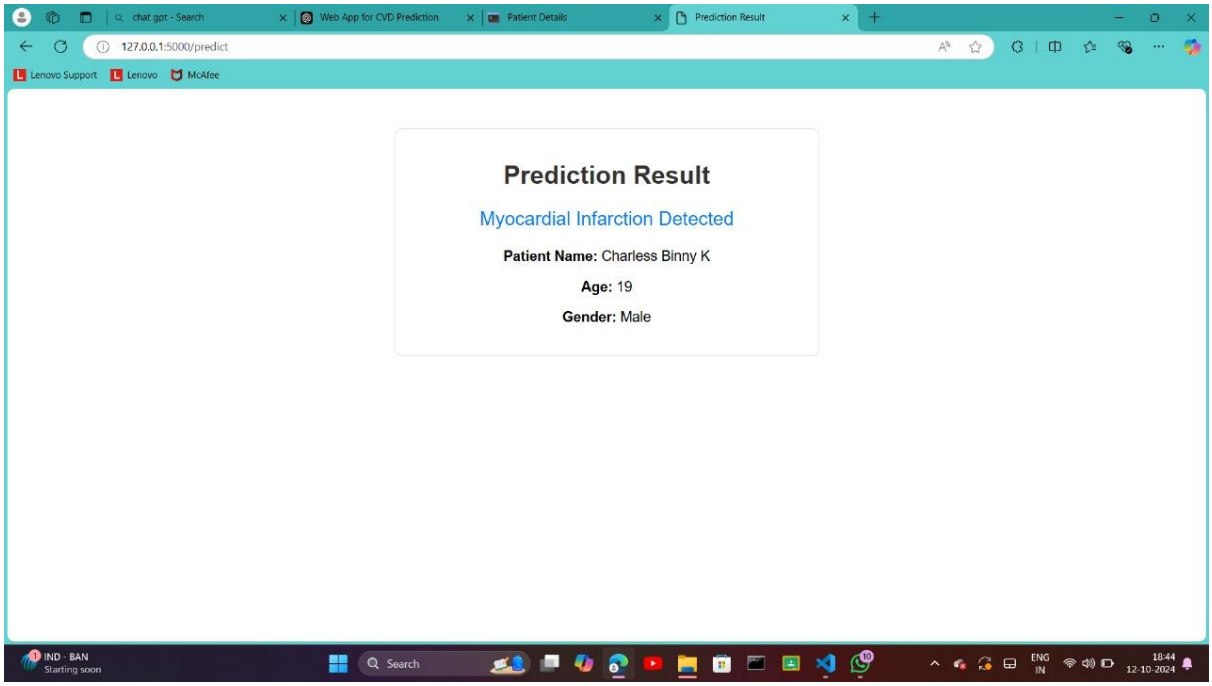


Fig A1.2.7: Predicted Result Page

REFERENCES

1. A.T. Das, S. Sarkar, and S. Chakraborty, "ECG Signal Analysis Using Convolutional Neural Networks for the Detection of Myocardial Infarction," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, no. 1, pp. 1-10, Jan. 2021.
2. H. Rajendra Acharya, U. Rajendra Acharya, and N. Kannathal, "Automated Identification of Myocardial Infarction Using Heart Rate Signals," *IEEE Engineering in Medicine and Biology Magazine*, vol. 21, no. 4, pp. 37-47, July-Aug. 2020.
3. K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Proc. of International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
4. F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 2017, pp. 1251–1258.
5. M. Abdar, F. Khosravi, and M. Nahavandi, "A Review of Deep Learning Approaches for ECG Signal Analysis," *IEEE Access*, vol. 9, pp. 86015-86038, 2021.
6. S. Yadav, K. Rawal, and S. Narayan, "Application of Deep Neural Networks in Biomedical Signal Processing," *IEEE Transactions on Biomedical Engineering*, vol. 67, no. 6, pp. 1573-1585, June 2020.
7. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA: MIT Press, 2016, pp. 324-340.