OPTIMIZED CNN - BASED HYBRID MODEL FOR MULTI - LABEL CLASSIFICATION OF CARDIOVASCULAR DISEASES IN 12 - LEAD ECG IMAGES

A PROJECT REPORT

Submitted by

YUVAPRIYA S (2022503552)

POOJA S (2022503024)

ATHITHRAJA R (2022503702)

COURSE CODE: CS6611

COURSE TITLE: CREATIVE AND INNOVATIVE PROJECT

DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS

CHENNAI – 600044

MAY 2025

DEPARTMENT OF COMPUTER TECHNOLOGY

ANNA UNIVERSITY, MIT CAMPUS CHROMEPET,

CHENNAI – 600044

BONAFIDE CERTIFICATE

Certified that this project report "**Optimized CNN-Based Hybrid Model for Multi-Label Classification of Cardiovascular Diseases in 12-Lead ECG Images**" is the work of **Ms. Yuvapriya S (2022503552), Ms. Pooja S (2022503024)** and **Mr. Athithraja R (2022503702)** in the Creative and Innovative Project subject code CS6611 during the period January to May 2025.

SIGNATURE

Dr. T. Sudhakar
SUPERVISOR
Assistant Professor
Department of Computer Technology
Anna University, MIT Campus
Chromepet – 600 044

SIGNATURE

Dr. P. Jayashree
HEAD OF THE DEPARTMENT
Professor
Department of Computer Technology
Anna University, MIT Campus
Chromepet – 600 044

ABSTRACT

The early and accurate diagnosis of cardiovascular diseases (CVDs) is vital for reducing global mortality rates. However, traditional ECG interpretation methods often suffer from subjectivity and inefficiency, especially when screening for multiple co-occurring conditions. This project addresses these limitations by leveraging deep learning techniques for automated, multi-label classification of 12-lead ECG images.

This project proposes an optimized Convolutional Neural Network (CNN)-based hybrid model that classifies ECG data into four distinct categories: Normal, Abnormal, Myocardial Infarction, and History of Myocardial Infarction. A dual-branch CNN architecture is employed for robust feature extraction, followed by gradient boosting classifiers such as LightGBM to enhance classification accuracy and performance.

Using a dataset collected from Mendeley Data, the model pipeline includes preprocessing techniques (e.g., resizing, normalization, augmentation), advanced feature extraction, hyperparameter tuning and rigorous evaluation using precision, recall, F1-score, and confusion matrices.

The model achieves a remarkable classification accuracy of **99.46%**, demonstrating its potential for clinical integration in real-time diagnosis systems. The modularity and scalability of this framework allow it to be extended to larger datasets and additional cardiac conditions, setting the stage for future portable, AI-powered ECG diagnostic tools.

ACKNOWLEDGEMENT

We take this humble opportunity to thank the Dean, MIT Campus, Anna University, **Dr. K. Ravichandran**, and **Dr. P. Jayashree**, Professor & Head, Department of Computer Technology, MIT Campus, Anna University for providing all the lab facilities in pursuit of this project.

Undertaking this project has helped us learn a lot, and we would like to express our gratitude towards our supervisor **Dr. T. Sudhakar**, Assistant Professor, Department of Computer Technology, MIT, Anna University, whose guidance, and directions helped shape this project perfectly. The feedback from the supervisor was very instrumental in the successful completion of the project work.

We acknowledge the efforts and feedback of the panel members **Dr. S. Chithra**, Associate Professor, Department of Computer Technology, MIT, Anna University, **Dr. S. Neelavathy Paari**, Assistant Professor, Department of Computer Technology, MIT, Anna University, and **Dr. K. Kottilingam**, Assistant Professor, Department of Computer Technology, MIT, Anna University, in reviewing our work, providing constant valuable comments and encouraging us to view the different aspects of the project in successful implementation of the project.

We thank all the teaching and non-teaching members of the Department of Computer Technology, MIT, Anna University and also our friends and family whose appreciable help, either directly or indirectly, helped in the smooth completion of the Creative and Innovative Project in a limited time frame.

Yuvapriya S 2022503552

Pooja S 2022503024

Athithraja R 2022503702

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	**iii**
	LIST OF FIGURES	**vii**
	LIST OF TABLES	**viii**
	LIST OF ABBREVIATIONS	**ix**
1	**INTRODUCTION**	**1**
	1.1 MACHINE LEARNING	1
	1.2 12 LEAD ECG IMAGES	2
	1.2.1 Advantages of 12 Lead ECG Images	3
	1.2.2 Properties of 12 Lead ECG Images	4
	1.3 IMAGE PROCESSING	5
	1.4 OBJECTIVES	6
2	**LITERATURE SURVEY**	**7**
3	**PROPOSED MODEL**	**10**
	3.1 INTRODUCTION	10
	3.2 DATA COLLECTION	13
	3.3 DATA PREPROCESSING	14
	3.3.1 Need for Preprocessing	14
	3.3.2 Image Normalization	14
	3.3.3 Image Augmentation	15
	3.3.4 Label Encoding	15

CHAPTER NO.	TITLE	PAGE NO.
	3.4 FEATURE EXTRACTION	15
	3.5 CLASSIFICATION	17
4	**IMPLEMENTATION**	**21**
	4.1 PLATFORMS USED	21
	4.2 TOOLS AND LIBRARIES	22
	4.3 IMPLEMENTATION FLOW	24
	4.4 PESUDO CODE FOR DATA PREPROCESSING	28
	4.5 PSEUDO CODE FOR BUILDING CNN	29
	4.6 PSEUDO CODE FOR FEATURE EXTRACTION	29
5	**RESULTS AND ANALYSIS**	**31**
	5.1 EVALUATION METRICS	31
	5.2 CLASSIFIER WISE PERFORMANCE ANALYSIS	33
	5.3 TRAINING DYNAMICS	43
	5.4 RESULTS OF MODEL INTEGRATION WITH UI	46
	5.5 JUSTIFICATION FOR SELECTING LIGHTGBM	47
6	**CONCLUSION AND FUTURE WORKS**	**48**
	6.1 CONCLUSION	48
	6.2 FUTURE WORK	48
7	**REFERENCES**	**49**

LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
3.1	ARCHITECTURE DIAGRAM FOR THE PROPOSED MODEL	12
3.2	SAMPLES FROM THE ECG IMAGES DATASET	16
3.3	CNN FLOW CHART	20
5.1	SVM CLASSIFICATION REPORT	33
5.2	SVM CONFUSION MATRIX	34
5.3	RANDOM FOREST CLASSIFICATION REPORT	35
5.4	RANDOM FOREST CONFUSION MATRIX	35
5.5	XGBOOST CLASSIFICATION REPORT	36
5.6	XGBOOST CONFUSION MATRIX	37
5.7	LIGHTGBM CLASSIFICATION REPORT	38
5.8	LIGHTGBM CONFUSION MATRIX	38
5.9	DECISION TREE CLASSIFICATION REPORT	39
5.10	DECISION TREE CONFUSION MATRIX	40
5.11	NAIVE BAYES CLASSIFICATION REPORT	41
5.12	NAIVE BAYES CONFUSION MATRIX	41
5.13	KNN CLASSIFICATION REPORT	42
5.14	KNN CONFUSION MATRIX	43
5.15	ACCURACY VS. EPOCHS	44
5.16	LOSS VS. EPOCHS	45
5.17	UI DEMO FOR ECG IMAGE CLASSIFIER	46

LIST OF TABLES

TABLE NO.	TITLE	PAGE NO.
3.1	PUBLIC ECG IMAGES DATASET DESCRIPTION	13
5.1	SUMMARY OF CLASSIFIER PERFORMANCE	47

ABBREVIATIONS

ECG	-	Electrocardiogram
CNN	-	Convolutional Neural Network
CVD	-	Cardiovascular Disease
LR	-	Learning Rate
ReLU	-	Rectified Linear Unit
LeakyReLU	-	Leaky Rectified Linear Unit
KNN	-	K-Nearest Neighbors
SVM	-	Support Vector Machine
XGBoost	-	Extreme Gradient Boosting
LightGBM	-	Light Gradient Boosting Machine
ROC	-	Receiver Operating Characteristic
Adam	-	Adaptive Moment Estimation
VS Code	-	Visual Studio Code
GPU	-	Graphics Processing Unit
TPU	-	Tensor Processing Unit
UI	-	User Interface
HTML	-	HyperText Markup Language
CSS	-	Cascading Style Sheets
IDE	-	Integrated Development Environment
API	-	Application Programming Interface

CHAPTER 1

INTRODUCTION

Machine learning algorithms can effectively analyze 12-lead ECG images to detect and classify cardiovascular diseases over time, providing valuable insights into cardiac health patterns and aiding in timely diagnosis and intervention.

1.1 MACHINE LEARNING

Machine learning (ML) is a subfield of artificial intelligence (AI) that focuses on the development of computer algorithms that can learn and improve from data without being explicitly programmed. It is a statistical approach to studying and making inferences about data that utilizes a variety of algorithms suited for answering complex questions and making predictions [2]. At its core, machine learning enables computers to learn and make decisions based on data, rather than following a set of pre-programmed instructions Machine learning is a process in which computing systems learn from data and use algorithms to execute tasks without being explicitly programmed. This is achieved by feeding the training data into the selected model, allowing the algorithm to learn the underlying patterns and relationships in the data [5]. The performance of ML algorithms adaptively improves with an increase in the number of available samples during the ‘learning phase'.

There are three main types of machine learning: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning involves training the model on labeled data, where the inputs and their corresponding outputs are provided [4]. Unsupervised learning involves training the model on unlabeled data, and it tries to find inherent patterns or structures within the data.

Reinforcement learning involves the model learning by interacting with an environment and receiving feedback (rewards or penalties) for its actions. Machine learning is a powerful tool that is driving innovation and automation across a wide range of industries, making it an increasingly important technology in the modern world.

1.2 12 LEAD ECG IMAGES

A 12-lead ECG image is a graphical representation of the heart's electrical activity recorded from twelve different perspectives. Each lead captures a unique projection of the cardiac electrical signals, providing detailed information about the heart's rhythm, conduction pathways, and potential abnormalities. These images consist of waveforms such as P waves, QRS complexes, and T waves, which are essential for identifying a wide range of cardiovascular diseases.

Each lead acts as an independent channel, capturing signals from different anatomical areas of the heart. When combined, the 12-lead configuration offers a comprehensive view of the heart’s function, significantly improving diagnostic accuracy. Processing 12-lead ECG images involves analyzing patterns, waveform morphology, and temporal features across all leads. This multiview representation enables advanced machine learning models to detect complex patterns and classify conditions more effectively.

ECG image analysis requires robust preprocessing to handle inconsistencies such as noise, baseline wander, and variations in image quality. This includes resizing images, normalizing pixel values, and applying augmentation techniques to increase dataset diversity and reduce overfitting.

1.2.1 ADVANTAGES OF 12 LEAD ECG IMAGES

1. The 12-lead ECG provides a complete electrical overview of the heart by recording signals from multiple anatomical regions. This allows for the detection of abnormalities in specific parts of the heart, such as the anterior, lateral, inferior, and septal walls, which may not be visible in single-lead ECGs.

2. The availability of twelve distinct perspectives increases the precision of diagnosing complex cardiac conditions, such as myocardial infarction, ischemia, conduction blocks, and arrhythmias. This comprehensive view allows for better pattern recognition and reduces the risk of misdiagnosis.

3. Each lead captures different waveform characteristics, enabling machine learning models—especially Convolutional Neural Networks (CNNs)—to extract a broader set of spatial and temporal features. This enhances the model's learning ability and leads to higher classification performance.

4. The 12-lead ECG format is a standard in clinical diagnostics, meaning models trained on such data can be easily integrated into existing healthcare systems and workflows. This compatibility increases the real-world applicability and relevance of the developed solution.

5. ECG waveforms provide valuable information about both the shape (morphology) and timing (intervals) of electrical activity. Having data from 12 leads enhances the model's ability to detect subtle changes in waveforms, such as ST elevation or prolonged QT intervals, which are critical for early detection of life-threatening conditions.

1.2.2 PROPERTIES OF 12 LEAD ECG IMAGES

12-lead ECG images consist of signals captured from twelve different leads, which provide a multi-dimensional view of the heart's electrical activity. Each lead captures electrical impulses from different angles and regions of the heart, such as the frontal and horizontal planes, offering comprehensive data on the heart's functioning. This multi-lead configuration helps in detecting subtle variations in the heart's electrical signals, which may be missed with fewer leads. The ECG signals recorded in these images are time-series data, typically captured over 10 seconds, providing insight into both the morphology and timing of the heart's electrical impulses.

The 12-lead ECG image format includes multiple waveforms—P waves, QRS complexes, and T waves—each representing different phases of the cardiac cycle. The variations in these waveforms, such as amplitude, duration, and shape, are crucial for diagnosing conditions like arrhythmias, myocardial infarctions, and other heart abnormalities. Preprocessing of these images involves resizing to a consistent resolution, normalizing pixel values, and sometimes cropping irrelevant information like image headers or footers. Additionally, data augmentation techniques such as rotation, zooming, and shifting are applied to increase the diversity of the training dataset, which helps to mitigate overfitting and improve the generalization of machine learning models. These properties of the 12-lead ECG images make them a valuable resource for detecting cardiovascular diseases through automated, deep learning-based systems.

1.3 IMAGE PROCESSING

a) Image Resizing

ECG images are resized to a uniform resolution of 227x227 pixels to maintain consistency across the dataset. This resizing step ensures that all images are of the same dimension, which is crucial for feeding them into the Convolutional Neural Network (CNN).

b) Normalization

Pixel values are normalized by scaling the values to the range [0, 1]. This is done by dividing each pixel by 255, which is a common practice to ensure that all input values are in the same scale, helping the model converge faster and improving training stability

c) Data Augmentation

To enhance model generalization and reduce overfitting, several data augmentation techniques are applied to the ECG images. These include:

Random Rotation: Slight rotation of the images to help the model learn to classify rotated ECG patterns.

Random Shifting: Shifting the images randomly within a small range to simulate variations in patient positioning.

Random Zooming: Zooming in and out of images to help the model generalize across variations in heart signal intensities.

d) Feature Extraction

Features are extracted using the output of the dense layer of the trained CNN model. These features are further processed and normalized for use with other classifiers, such as LightGBM and XGBoost, to enhance classification performance.

1.4 OBJECTIVES

1. To implement a CNN-based model for multi-label classification of cardiovascular diseases from ECG images.
2. To optimize preprocessing techniques such as image resizing (800KB to 300KB) and normalizing.
3. To optimize training performance using Adam for better accuracy and faster convergence.
4. To apply image augmentation techniques such as rotation, shift, and zoom, in order to improve the generalization of the model and mitigate overfitting and class imbalance.

ORGANIZATION OF THE THESIS

Chapter 2 presents the literature survey and background studies relevant to ECG signal classification, including an overview of CNN models and traditional classifiers. Chapter 3 explains the system design and architecture, detailing the proposed CNN model with parallel branches and the integration of traditional classifiers. Chapter 4 outlines the system implementation, covering data preprocessing, model training, feature extraction, and classification procedures. Chapter 5 discusses the results and performance evaluation using various metrics, along with comparative analysis of different classifiers. Chapter 6 concludes the work with key findings, limitations, and possible directions for future enhancements.

CHAPTER 2

LITERATURE SURVEY

In [1], Bharti et al. proposed a hybrid method that combines machine learning and deep learning techniques to predict heart disease using the UCI Machine Learning Heart Disease dataset. The authors used Isolation Forest for handling irrelevant features and normalized the dataset. The model was evaluated using accuracy and confusion matrix. However, the study acknowledged the need for a larger dataset and further optimization and normalization techniques for better model performance. Future work suggests exploring multimedia integration.

Quiroz-Juárez et al. in [2] introduced an extended heterogeneous oscillator model to generate 12-lead ECG waveforms, incorporating pacemakers, muscles, RR-tachogram, and heart rate variability. This model allows for realistic ECG signal generation. However, the authors noted the complexity and computational demands, with limitations related to its reliance on theoretical constructs, which could impact real-world applications.

Khan et al. in [3] utilized a dataset of 11,148 12-lead ECG images to develop a cardiac abnormality detection system. The authors employed a Single Shot Detection (SSD) MobileNet v2-based deep neural network to detect four specific cardiac conditions. While the method yielded promising results, Khan et al. noted the necessity of expanding the dataset to include more diverse cardiac abnormalities to improve model robustness. Additionally, they suggested further research into advanced feature extraction methods and domain adaptation learning to enhance the model's ability to generalize across various ECG conditions and data distributions.

Sharma et al. in [4] proposed a hybrid machine learning model combining CNN and Bi-LSTM for heartbeat classification using the MIT-BIH arrhythmias database. The study incorporated synthetic minority oversampling to address class imbalance. However, the authors emphasized that future work should focus on increasing dataset size, applying optimization techniques, and exploring different data normalization methods.

Kiranyaz et al. in [5] introduced a 1-D CNN-based patient-specific model for real-time ECG classification, focusing on the fusion of feature extraction and classification. The model aimed to address individual variation in ECG data, which can be critical in personalized healthcare. Despite its success, the authors pointed out significant limitations, particularly in characterizing anomaly beats effectively and the lack of support for incremental learning or active learning. They also suggested the possibility of hardware implementation in future work to make the model more applicable in real-time clinical settings.

Zhou et al. in [6] proposed the ECGMatch model, which integrates the ECGAugment module for data augmentation in multi-label cardiovascular disease prediction. The model uses a hyperparameter-efficient framework for pseudo-label generation and a label correlation alignment module to address class imbalance. While this approach showed promise, the authors acknowledged that the class imbalance problem in ECG datasets remains a major challenge, impacting the overall robustness and generalization of the model.

De Giovanni et al. in [7] focused on the detection of R-peaks in ECG signals, particularly during high-intensity exercise, using a combination of unsupervised learning, Bayesian filtering, and non-linear normalization. The approach was designed to handle dynamic changes in heart rate during intense physical activity. However, the authors noted several limitations, including a small dataset and the narrow focus of the study, which only considered cycling and single-lead ECGs. They proposed expanding the dataset and considering

more diverse exercise types in future work.

Avanzato and Beritelli in [8] used a CNN for ECG signal classification, trained on a dataset of 4,000 ECG signal instances. While their approach demonstrated promising results, the authors suggested increasing the dataset size and exploring various optimization and normalization techniques for improved accuracy and generalization.

Zvuloni et al. in [9] introduced a novel approach that merges feature engineering and deep learning for the diagnosis, risk prediction, and age estimation based on 12-lead ECGs. Using a dataset of 2.3 million ECG recordings, their model successfully predicted risk and age estimates. However, they acknowledged the need for further work to increase the dataset size, apply more advanced optimization techniques, and explore alternative data normalization strategies to further enhance the model's performance.

Abubaker and Babayiğit [10] utilized transfer learning with pretrained models such as SqueezeNet and AlexNet to classify ECG images, introducing a custom CNN aimed at enhancing cardiovascular disease detection accuracy. Despite strong performance, they highlighted the importance of expanding the dataset and applying advanced optimization methods. They also recommended testing alternative data normalization techniques and exploring multimedia integration to boost clinical usefulness.

The related works were surveyed, and the limitations were identified to propose an organized work for our research, as described in the following elaborative headings.

CHAPTER 3

PROPOSED MODEL

The proposed work focuses on developing an efficient and optimized Convolutional Neural Network (CNN)-based model for the multi-label classification of cardiovascular diseases (CVDs) from 12-lead ECG images. The primary objective is to classify and analyze ECG data from various heart conditions, such as Myocardial Infarction (MI), History of MI, Abnormal ECGs, and Normal ECGs, and evaluate the health status of the cardiovascular system using CNN-based results.

3.1 INTRODUCTION

The analysis and classification of electrocardiogram (ECG) signals are crucial in the early detection and management of cardiovascular diseases (CVDs), which remain a leading cause of mortality worldwide. ECGs provide vital diagnostic information about the heart's electrical activity, enabling clinicians to identify abnormalities such as myocardial infarction, arrhythmias, and other cardiac conditions. However, manual interpretation of ECGs can be time-consuming, prone to error, and dependent on the expertise of the medical professional. To address these challenges, this project utilizes deep learning—specifically a Convolutional Neural Network (CNN)-based hybrid model—to automate the classification of 12-lead ECG images.

By leveraging machine learning algorithms, the system is capable of analyzing large volumes of ECG data, learning complex patterns, and accurately classifying heart conditions into multiple categories such as Normal, Abnormal, Myocardial Infarction, and History of Myocardial Infarction. The insights obtained from this model have the potential to enhance clinical decision-making, reduce diagnostic delays, and ultimately improve patient outcomes. Furthermore, the

system can be scaled and integrated into real-time diagnostic platforms for widespread clinical use.

The proposed architecture diagram shown in Figure 3.1 includes the following core stages:

a) Data Collection

This is the initial step, where labelled 12-lead ECG images are gathered from publicly available medical datasets. The collected data serves as the foundation for training and testing the classification model, ensuring a diverse and balanced representation of different heart conditions.

b) Data Preprocessing

The ECG images collected from the dataset are pre-processed to enhance their quality and ensure consistency. This includes resizing images to a standard dimension (227x227 pixels), and normalizing pixel values to a [0, 1] range for optimized model performance.

c) Classification

The pre-processed ECG images are then fed into the CNN model for classification. The model analyses the input images and classifies them into one or more of the predefined categories: Normal, Abnormal, Myocardial Infarction, and History of Myocardial Infarction. The architecture leverages deep feature extraction to learn complex patterns associated with each condition.

d) Feature Extraction

Feature extraction is performed automatically by the Convolutional Neural Network (CNN) during the training process. As ECG images pass through multiple convolutional layers, the model learns to detect important features such as abnormalities. The output from the final convolutional and pooling layers is then flattened into a one-dimensional feature vector, which

is passed to dense layers for classification. This deep feature representation enables the model to distinguish between Normal, Abnormal, Myocardial Infarction, and History of MI with high accuracy.

ARCHITECTURE DIAGRAM FOR THE PROPOSED MODEL

Figure 3.1 illustrates the architecture of the proposed ECG classification system, from image input to CNN-based feature extraction and final ML-based classification.

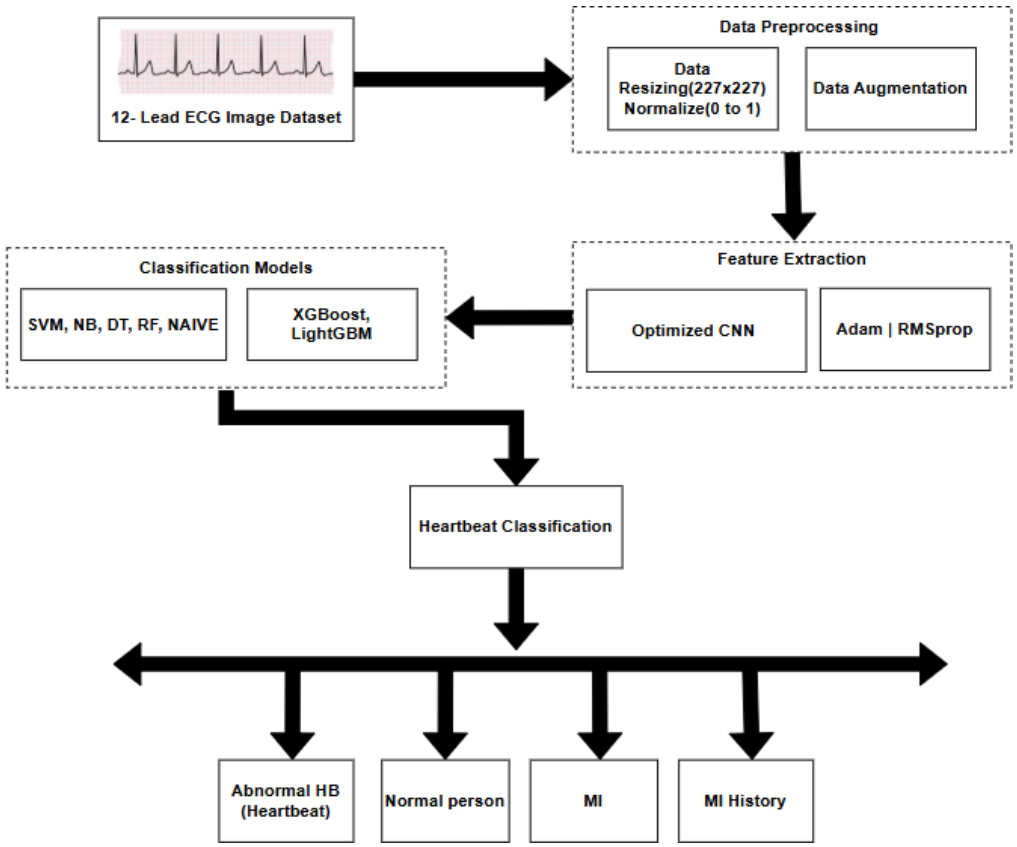


Figure 3.1. Architecture diagram for the proposed model

The above figure clearly illustrates the modular design of the system, separating the deep learning-based feature extraction phase from the classical classification phase. This hybrid approach improves flexibility and performance. The use of parallel branches in the CNN model enhances spatial and global feature learning from ECG images.

3.2 DATA COLLECTION

The dataset used in this project is titled "ECG Images Dataset of Cardiac Patients", published on 19 March 2021 (Version 2). It was created under the auspices of the Ch. Pervaiz Elahi Institute of Cardiology, Multan, Pakistan, and is made publicly available by the University of Management and Technology. The dataset is released under the Creative Commons CC BY 4.0 license, allowing its use for scientific and research purposes with proper attribution.

This dataset provides a comprehensive collection of 12-lead ECG images categorized into four cardiac conditions. Each ECG record includes 12 individual lead images, resulting in a total of 11,148 images, divided as shown in Table 3.1.

Table 3.1 Public ECG Images Dataset Description

Class	**Number of Patients**	**Leads per Patient**	**Total Images**
Myocardial Infarction (MI)	240	12	2,880
Abnormal Heartbeat	233	12	2,796
History of Myocardial Infarction	172	12	2,064
Normal ECG	284	12	3,408
Total	**929**	—	**11,148**

The dataset has been clinically validated and labeled by expert cardiologists, ensuring reliability for training machine learning and deep learning models. With a broad and balanced distribution of ECG data, this dataset serves as an ideal foundation for building and evaluating a CNN-based hybrid model for multi-label classification of cardiovascular diseases.

All ECG images were downloaded and categorized according to their class labels, then processed and used throughout the development of the classification model in this project.

3.3 DATA PREPROCESSING

Data preprocessing in images for machine learning models refers to the set of steps taken to prepare raw image data so that it can be effectively used by an ML or deep learning model. The goal is to enhance the quality and consistency of the input data, reduce noise, and ensure the data format matches what the model expects.

3.3.1 NEED FOR PREPROCESSING

In ECG image classification, preprocessing is essential to improve the quality of input data and ensure that the model can extract relevant features. ECG images may contain irrelevant artifacts such as noise, headers, and footers, which do not contribute to the classification task. Preprocessing steps such as cropping, resizing, and normalization help eliminate these artifacts and ensure the data is uniform, reducing variability and enhancing the model's ability to generalize.

3.3.2 IMAGE NORMALIZATION

Image normalization is a vital preprocessing step used to standardize the pixel intensity values of ECG images before inputting them into the CNN model. Raw ECG images typically have pixel values ranging from **0 to 255**, which can result in inconsistent input distributions and hinder the learning process. Normalizing these pixel values to a **[0, 1]** range ensures uniformity, improves model convergence speed, and enhances training stability.

This normalization is performed by dividing each pixel value by 255, the maximum possible value in an 8-bit grayscale image. The resulting normalized values enable the model to treat all features equally and prevent dominance by larger

values.

3.3.3 IMAGE AUGMENTATION

To improve model generalization and overcome dataset imbalance, data augmentation techniques were applied. Augmentation artificially expands the dataset by introducing variability in the images without altering the underlying patterns. The following transformations were used:

- **Rotation:** Random rotation of ECG images to simulate orientation differences.
- **Shifting:** Horizontal and vertical translation to mimic positional variations.
- **Zooming:** Random zoom to help the model learn features at multiple scales. These augmentation techniques help the model become more robust to real-world variations in ECG image capture and positioning.

3.3.4 LABEL ENCODING

The dataset was organized into four labeled categories—Normal, Abnormal, Myocardial Infarction, and History of MI. Each ECG image was stored in its corresponding folder based on the clinical diagnosis. This structured approach supports supervised learning, enabling the model to learn distinct patterns associated with each condition. The labels are later encoded into numeric format for training and evaluation within the neural network.

3.4 FEATURE EXTRACTION

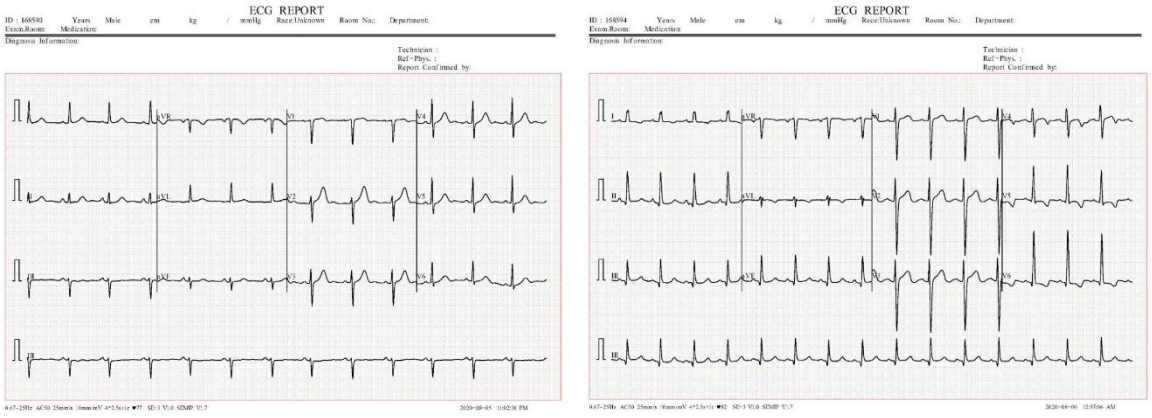
Feature extraction is a crucial step in the process of ECG image classification. It involves identifying and capturing meaningful patterns and visual characteristics in the ECG waveforms that differentiate between various cardiac conditions. The CNN model automatically extracts relevant features through its convolutional layers by scanning the ECG image with multiple filters. These filters detect basic features such as edges and lines in the initial layers, and more abstract patterns like waveform

morphology in the deeper layers.

To further improve feature quality, the proposed model utilizes a dual-branch architecture: a stack branch for spatial feature extraction and a full branch for global pattern learning. The stack branch consists of multiple convolutional, batch normalization, and pooling layers that progressively reduce the image dimensions while preserving essential features. Meanwhile, the full branch processes the flattened input to retain global context and morphological information. These two branches are then merged, ensuring that both spatial and holistic features contribute to the classification process.

The dataset is organized into four distinct classes: **Normal**, **Abnormal**, **Myocardial Infarction (MI)**, and **History of MI**. During preprocessing, ECG images are resized, normalized, and converted to grayscale. From there, these processed images are passed through a CNN that extracts deep features, which are then used to classify each sample. Figure 3.2 illustrates the sample ECG waveform patterns associated with different classes of heart conditions.

After feature extraction, the output is flattened and passed through dense layers for classification. A softmax layer at the end provides probability scores for each of the four classes. The model is trained using labeled ECG images to learn class-specific patterns effectively.



(a) (b)

(c) (d)

Figure 3.2 Samples from the ECG Images Dataset

(a) Normal (b) Abnormal (c) Myocardial Infarction (MI) (d) History of MI

3.5 CLASSIFICATION

The ECG image data must be carefully prepared before being used in the Convolutional Neural Network (CNN) classification model. CNNs are specifically designed to process image data by automatically learning hierarchical features. Once preprocessed, the ECG images are divided into training and validation sets. The training set is used to optimize the CNN's parameters, while the validation set is used to monitor the model's performance and avoid overfitting during training.

a) Convolutional Layers

These layers serve as the core of the CNN. They apply filters that slide across the ECG image to detect patterns such as wave peaks, intervals, and signal morphology. Initial layers capture simple patterns like edges and lines, while deeper layers learn higher-level features critical for distinguishing between cardiac conditions.

b) Pooling Layers

Pooling layers reduce the spatial dimensions of the feature maps created by convolutional layers. This helps in decreasing the computational load while retaining the most relevant information. Max pooling is commonly used to retain the most prominent features, which is important in ECG signal recognition.

c) Dropout Layer

To prevent overfitting, dropout layers are included in the network. These layers randomly deactivate a fraction of neurons during training, ensuring the model does not become too dependent on specific features in the training data and maintains the ability to generalize on unseen ECG images.

d) Dense Layer

Also known as fully connected layers, these take the flattened feature maps and perform the final classification. They integrate all the learned features and output predictions corresponding to the defined classes—Normal, Abnormal, Myocardial Infarction, and History of MI.

e) Activation Function

Activation functions such as Rectified Linear Unit (ReLU) are applied to introduce non-linearity in the network, allowing it to learn complex patterns. The final dense layer uses the Softmax activation function to generate class probabilities for multi-class classification.

CNN Working Flow

Figure 3.3 illustrates the flowchart of the CNN-based feature extraction process used in the proposed system. It represents the sequential steps followed during training and prediction phases using convolutional, activation, pooling, and dense layers.

The flowchart clearly demonstrates how ECG images are transformed into high-level feature representations. It also highlights how spatial and global features are processed in parallel before being merged and classified. This modular design contributes to efficient learning and supports the adaptability of the system to ECG variations across different patient classes.

This chapter detailed the design and implementation of the proposed ECG classification system, including dataset structure, preprocessing techniques, CNN model architecture, and feature extraction methodology. The combination of deep learning for representation learning and classical classifiers for final prediction forms the core of the hybrid model. This design aims to effectively learn discriminative features from ECG images, enabling accurate classification of various heart conditions.

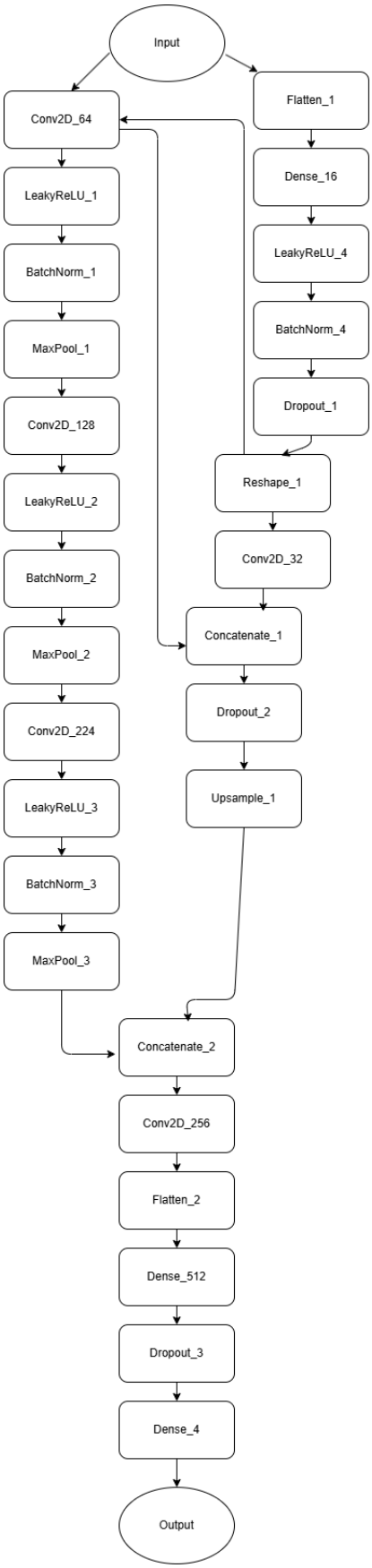


Figure 3.3. CNN Flow Chart

CHAPTER 4

IMPLEMENTATION

The implementation of this project involves a systematic sequence of steps aimed at accurately classifying ECG (Electrocardiogram) images using Convolutional Neural Networks (CNNs). The primary objective is to automate the detection of cardiac abnormalities by analyzing ECG images through deep learning models.

4.1 PLATFORMS USED

We are using Google Colab for cloud-based ML development, VSCode as the local development environment, and Django for the backend to manage server-side logic and database interactions, creating a flexible and efficient workflow.

1. Google Colab

Google Colab is a cloud-based Jupyter notebook environment provided by Google that allows developers to execute Python code on high-performance hardware, including GPUs and TPUs. In this project, Colab is used for:

- Training the CNN architecture on the ECG dataset.
- Performing data preprocessing and augmentation.
- Extracting features from the CNN model.
- Evaluating traditional classifiers on CNN-extracted features.

By leveraging Colab's GPU-accelerated environment, the model training time is significantly reduced, making it ideal for iterative development and experimentation.

2. Local Development System (VS Code)

Visual Studio Code (VS Code) is used as the Integrated Development Environment (IDE) for frontend development. It supports live preview of

changes and has powerful plugins for HTML, CSS, and ReactJS. The UI is designed with a focus on usability, responsiveness, and real-time interaction.

3. Django Backend

Django, a high-level Python web framework, is utilized for backend operations including:

- Serving API endpoints for ECG prediction.
- Handling user authentication and subscription management.
- Managing uploaded images and result storage.

4.2 TOOLS AND LIBRARIES

For tools and libraries, we are using Keras for building and training deep learning models, Scikit-learn for machine learning tasks, OpenCV for image processing, and NumPy and Pandas for data manipulation. For visualization, Matplotlib and Seaborn are employed to create insightful plots and charts.

a) TensorFlow/Keras

TensorFlow is a widely used open-source deep learning framework developed by Google. It provides a flexible ecosystem of tools and libraries for developing and deploying machine learning applications. Keras is a high-level API built on top of TensorFlow that simplifies the construction of deep learning models. In this project, Keras is used to implement a dual-branch CNN architecture for classifying 12-lead ECG images. The Functional API offered by Keras enables the construction of non-linear and multi-input/output model topologies. Its support for GPU acceleration significantly reduces training time, and its modular design allows for easy experimentation with layers, optimizers, and callbacks.

b) Scikit-learn

Scikit-learn is a powerful and user-friendly Python library that provides tools for machine learning and statistical modeling. In this project, Scikit-learn is used for several key tasks, including preprocessing the ECG image data, performing feature scaling, and evaluating model performance. It also supports traditional classifiers such as Support Vector Machines (SVM), Random Forest, and XGBoost, enabling a hybrid modeling approach. Scikit-learn's evaluation metrics—such as accuracy, precision, recall, F1-score, and confusion matrices—are used to validate the performance of the proposed model.

c) OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source toolkit focused on real-time computer vision. In this project, OpenCV is primarily used for ECG image preprocessing. It enables automated reading of ECG images in bulk and facilitates various image manipulation tasks such as cropping unnecessary headers/footers, converting images to grayscale, and resizing them to a fixed dimension of 227x227 pixels. These preprocessing steps are crucial to ensure consistent input data quality and to optimize model performance.

d) NumPy

NumPy (Numerical Python) is a fundamental library for numerical computing in Python. It supports multi-dimensional arrays and provides mathematical functions that allow for efficient data manipulation. In this project, NumPy plays a vital role in handling the matrix and tensor operations required during image preprocessing, model input formatting, and intermediate computation during the training of CNN models.

e) Pandas

Pandas is an open-source data analysis and manipulation library that provides high-performance data structures like Data Frames. It is used in this project to manage metadata associated with ECG images and to handle label mappings for classification. Its intuitive syntax and integration with other libraries make it ideal for organizing and transforming large datasets required for deep learning workflows.

f) Matplotlib

Matplotlib is a comprehensive Python library for data visualization. It is used in this project to plot training and validation performance metrics such as loss curves and accuracy trends over epochs. Additionally, Matplotlib is used to generate visual representations of confusion matrices and to compare model performance, helping to interpret the results and guide further model tuning.

g) Seaborn

Seaborn is a statistical data visualization library built on top of Matplotlib. In this project, it is used to create more informative and aesthetically pleasing plots such as heatmaps for confusion matrices and bar plots for comparing classifier performance. Seaborn's integration with Pandas simplifies plotting directly from structured data.

4.3 IMPLEMENTATION FLOW

Step 1: Data Loading and Preprocessing

a) Directory Structure

Data is organized into class-specific subdirectories following the ImageDataGenerator format. This allows real-time image augmentation and efficient mini-batch loading.

b) Image Cleaning and Resizing

Images are cropped using OpenCV functions to remove extra margins and non-informative regions like labels or timestamps. The final image is resized to 227x227 pixels and normalized to the [0,1] range.

c) Augmentation Techniques

To improve generalizability and prevent overfitting, several transformations are applied:

- Random horizontal/vertical flips.
- Small rotations.
- Zooming and shearing.

These techniques simulate real-world variations in ECG images due to scanning artifacts or device differences.

Step 2: CNN Model Building

The architecture is designed with **two parallel branches** to simultaneously capture fine-grained local patterns and holistic structural cues in ECG images.

a) Stack Branch (Convolutional Path)

- Conv2D layers with 64, 128, 224 filters.
- LeakyReLU activation to avoid neuron death.
- MaxPooling to reduce spatial dimensions.
- BatchNormalization for stable training.

This branch excels in learning localized features such as QRS complex shapes or T-wave morphology.

b) Full Branch (Dense-to-Deconv Path)

- Dense layers for global embedding.
- Reshape + Conv2DTranspose layers to reintroduce spatial dimensions.
- Useful for contextualizing global waveform structure.

c) Fusion and Classification

Outputs of both branches are concatenated. The combined features pass through:

- 1×1 Conv layers for feature compression.
- Global Average Pooling.
- Dense + Softmax for final classification into four diagnostic categories.

The model is compiled with categorical_crossentropy loss and the Adam optimizer.

Step 3: Feature Extraction

Rather than using the CNN only for end-to-end classification, the penultimate dense layer's output (a 512-dimensional feature vector) is captured as an embedding of the input ECG. These learned representations are then used as input for traditional ML classifiers.

This **hybrid approach** leverages:

- CNN for deep feature abstraction.
- ML classifiers for flexible decision boundaries.

A separate encoder model (CNN minus the final classification layer) is saved for future inference.

Step 4: Training Machine Learning Classifiers

- The extracted features serve as input to a range of classifiers: SVM, Random Forest, XGBoost, LightGBM, Decision Tree, Naive Bayes, and K-Nearest Neighbors.

- Each classifier is fine-tuned using Grid Search and 5-fold cross-validation.
- Performance is compared across metrics such as accuracy, precision, recall, and F1-score.
- Although all classifiers are evaluated, **LightGBM** was ultimately chosen for deployment due to its superior balance between performance and inference speed.

Step 5: Evaluation

a) Confusion Matrices

Provide visual insight into class-wise performance, especially for misclassified ECG types.

b) Classification Report

Includes metrics:

- Precision: Ability to correctly identify positives.
- Recall: Ability to detect all relevant instances.
- F1-Score: Harmonic mean of precision and recall.

Step 6: Frontend Integration

A clean and responsive web interface is built using **ReactJS**, focused on simplicity and accessibility for medical professionals.

a) Features

- Drag-and-drop ECG upload.

b) Backend Communication

- Files are sent to Django backend via API.
- Django handles:

- Image preprocessing using same OpenCV logic.
- Loading CNN model and ML classifiers.
- Generating predictions and confidence scores.

c) Result Display

- Display the Predicted Output

4.4 PSEUDO CODE FOR DATA PREPROCESSING

Input: dataset_path, img_size

Output: data (NumPy array), labels (NumPy array)

Steps:

Step 1: categories = ["Normal", "Abnormal", "MI", "History_MI"]

Step 2: data = [], labels = []

Step 3: For each category in categories:

a. path = dataset_path + category

b. For each img_name in os.listdir(path):

i. img_path = path + "/" + img_name

ii. img = cv2.imread(img_path)

iii. If img is not None:

1. img = cv2.resize(img, img_size)
2. img = img / 255.0
3. data.append(img)
4. labels.append(categories.index(category))

Step 4: data = numpy.array(data), labels = numpy.array(labels)

Step 5: Return data, labels

4.5 PSEUDO CODE FOR BUILDING CNN

Input: input_shape, num_classes

Output: compiled Keras model

Steps:

Step 1: Input Layer:

- Create input_layer.

Step 2: Stack Branch:

- Apply repeated Conv2D, LeakyReLU, BatchNorm, MaxPooling.

Step 3: Full Branch:

- Flatten, Dense, LeakyReLU, BatchNorm, Dropout, Reshape, Conv2D, Concatenate, Dropout, UpSampling.

Step 4: Concatenate:

- Combine Stack and Full Branch outputs.

Step 5: Final Layers:

- Conv2D, Flatten, Dense, Dropout, Dense (softmax).

Step 6: Create Model:

- Model(inputs, outputs).

Step 7: Compile:

- model.compile(adam, sparse_categorical

4.6 PSEUDO CODE FOR FEATURE EXTRACTION

Input:

- best_model: Trained Keras Model.
- X_train: NumPy array of training images.
- X_val: NumPy array of validation images.

Output:

- X_train_features: NumPy array of extracted features from training images.
- X_val_features: NumPy array of extracted features from validation images.

Steps:

Step 1: Create Feature Extractor Model:

- feature_extractor = Model(inputs=best_model.input, outputs=best_model.layers[-3].output) (Note: -3 refers to the third layer from the end, adjust if needed)

Step 2: Extract Training Features:

- X_train_features = feature_extractor.predict(X_train)

Step 3: Extract Validation Features:

- X_val_features = feature_extractor.predict(X_val)

Step 4: Return Extracted Features:

- Return X_train_features, X_val_features

This chapter detailed the step-by-step implementation of the proposed ECG classification system. It covered the preprocessing of ECG images, the construction of the custom CNN architecture, and the integration of classical classifiers for final prediction. Each stage of the system was carefully configured to handle the complexity and variability of ECG signals. The use of callbacks, data augmentation, and regularization techniques helped ensure that the model was trained efficiently and robustly. The foundation laid here sets the stage for evaluating model performance, which is discussed in the next chapter.

CHAPTER 5

RESULTS AND ANALYSIS

This chapter presents an in-depth evaluation of the hybrid ECG disease classification system, which integrates Convolutional Neural Network (CNN)-based feature extraction with traditional machine learning classifiers. The objective of this analysis is to systematically assess the performance of the proposed model using both quantitative metrics and qualitative interpretations. The evaluation framework involves multiple statistical indicators such as Accuracy, Precision, Recall, F1-Score, and Confusion Matrix, along with training dynamics visualizations such as Epoch-wise Accuracy and Loss curves. The comprehensive analysis offers both empirical justification and theoretical grounding for the final selection of **LightGBM** as the model for real-time web-based deployment.

5.1 EVALUATION METRICS

To ensure an objective and rigorous comparison among classifiers, standard performance metrics are employed. These metrics provide multiple perspectives on classification performance and are crucial in the domain of medical diagnosis, where both false positives and false negatives have significant implications.

True Positive (TP) – Correctly predicted positive class
True Negative (TN) – Correctly predicted negative class
False Positive (FP) – Incorrectly predicted positive class
False Negative (FN) – Incorrectly predicted negative class

Accuracy

Accuracy measures the overall percentage of correctly classified instances:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

It serves as a general performance indicator but can be misleading in the presence of class imbalance.

Precision

Precision evaluates the proportion of true positives among all instances predicted as positive:

$$Precision = \frac{TP}{TP + FP}$$

In medical applications, a high precision ensures a low rate of false alarms, thereby avoiding unnecessary treatments.

Recall (Sensitivity)

Recall measures the proportion of actual positives that are correctly identified:

$$Recall = \frac{TP}{TP + FN}$$

A high recall is vital in diagnostics to ensure that diseased cases are not overlooked.

F1-Score

F1-Score is the harmonic mean of Precision and Recall, offering a single metric that balances both:

$$F1 - Score = 2\ X\ \frac{Precision\ X\ Recall}{Precision + Recall}$$

This is particularly useful in scenarios where neither Precision nor Recall can be compromised.

Confusion Matrix

The confusion matrix offers a detailed view of true vs. predicted classifications across all classes, enabling the identification of specific misclassification trends.

5.2 CLASSIFIER-WISE PERFORMANCE ANALYSIS

Each classifier was trained using the same 512-dimensional feature vectors extracted via the CNN. This standardization ensures that the observed differences in performance can be attributed solely to the classifier design and learning capabilities.

a) Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm used for classification by finding the optimal hyperplane that separates data points of different classes. It is effective in high-dimensional spaces and works well when there is a clear margin of separation.

SVM Accuracy: 98.92%

Classification Report:

	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	57
Abnormal	0.98	0.98	0.98	47
MI	1.00	1.00	1.00	48
History_MI	0.97	0.97	0.97	34
accuracy			0.99	186
macro avg	0.99	0.99	0.99	186
weighted avg	0.99	0.99	0.99	186

Figure 5.1. SVM Classification Report

As shown in Figure 5.1, SVM demonstrated excellent performance in high-dimensional feature spaces, with near-perfect classification for Normal and MI cases. The kernel trick effectively mapped complex nonlinear decision boundaries, though minor confusion persisted between Abnormal and History_MI classes.

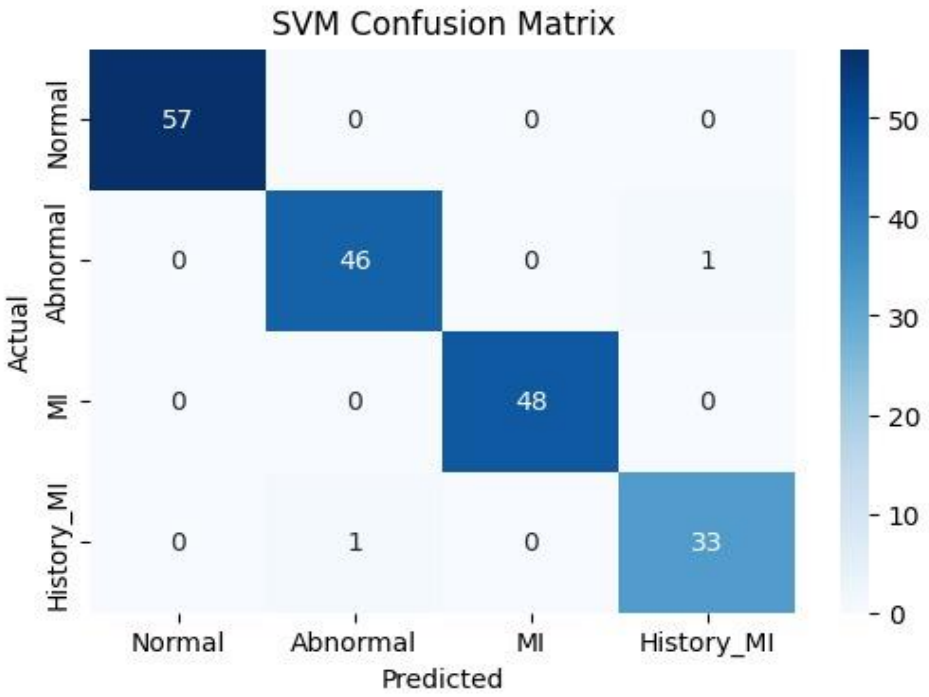


Figure 5.2. SVM Confusion Matrix

As shown in Figure 5.2, the SVM model correctly classified 57 Normal, 46 Abnormal, and 48 MI cases. However, it misclassified 1 Abnormal case as History_MI and 1 History_MI case as Abnormal. Overall, the model shows strong performance in classifying Normal and MI conditions but has some confusion between Abnormal and History_MI.

b) Random Forest (RF)

Random Forest is an ensemble learning method that builds multiple decision trees and combines their outputs for more accurate and robust predictions. It reduces overfitting and improves generalization by averaging the results of diverse trees.

```
Random Forest Accuracy: 98.39%

Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      1.00      0.99        57
    Abnormal       1.00      0.94      0.97        47
          MI       1.00      1.00      1.00        48
  History_MI       0.94      1.00      0.97        34

    accuracy                           0.98       186
   macro avg       0.98      0.98      0.98       186
weighted avg       0.98      0.98      0.98       186
```

Figure 5.3. Random Forest Classification Report

As shown in Figure 5.3, the ensemble nature of Random Forest provided robust generalization, with consistent performance across all disease categories. Marginally lower recall for History_MI (97%) was observed.

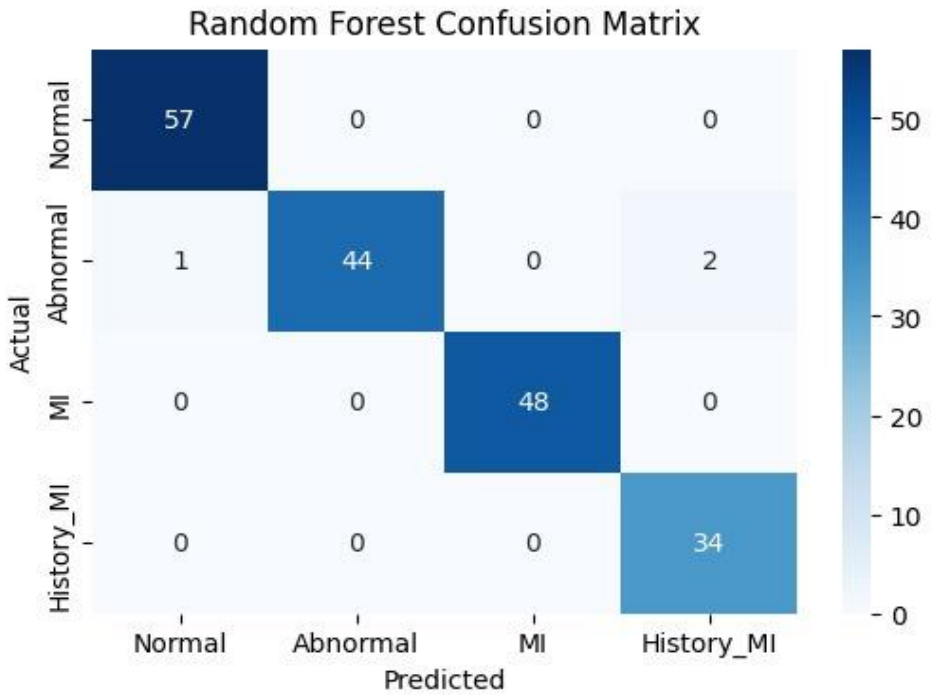


Figure 5.4. Random Forest Confusion Matrix

As shown in Figure 5.4, the Random Forest model accurately predicted 57 Normal, 44 Abnormal, and 48 MI cases. It misclassified 1 Normal case as Abnormal

and 2 Abnormal cases as History_MI. This model demonstrates good classification performance across all categories, with minor confusion in distinguishing Abnormal and History_MI.

c) XGBoost

XGBoost (Extreme Gradient Boosting) is a scalable, efficient, and high-performance machine learning algorithm based on decision trees. It is a powerful ensemble technique based on gradient boosting that builds models sequentially to correct the errors of previous models. It incorporates regularization and advanced optimization techniques, making it faster and more accurate than traditional boosting algorithms.

```
XGBoost Accuracy: 98.92%

Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      1.00      0.99        57
    Abnormal       1.00      0.96      0.98        47
          MI       1.00      1.00      1.00        48
  History_MI       0.97      1.00      0.99        34

    accuracy                           0.99       186
   macro avg       0.99      0.99      0.99       186
weighted avg       0.99      0.99      0.99       186
```

Figure 5.5. XGBoost Classification Report

As shown in Figure 5.5, XGBoost matched SVM's accuracy but with slightly better computational efficiency. Its gradient-boosting framework handled class imbalances effectively.

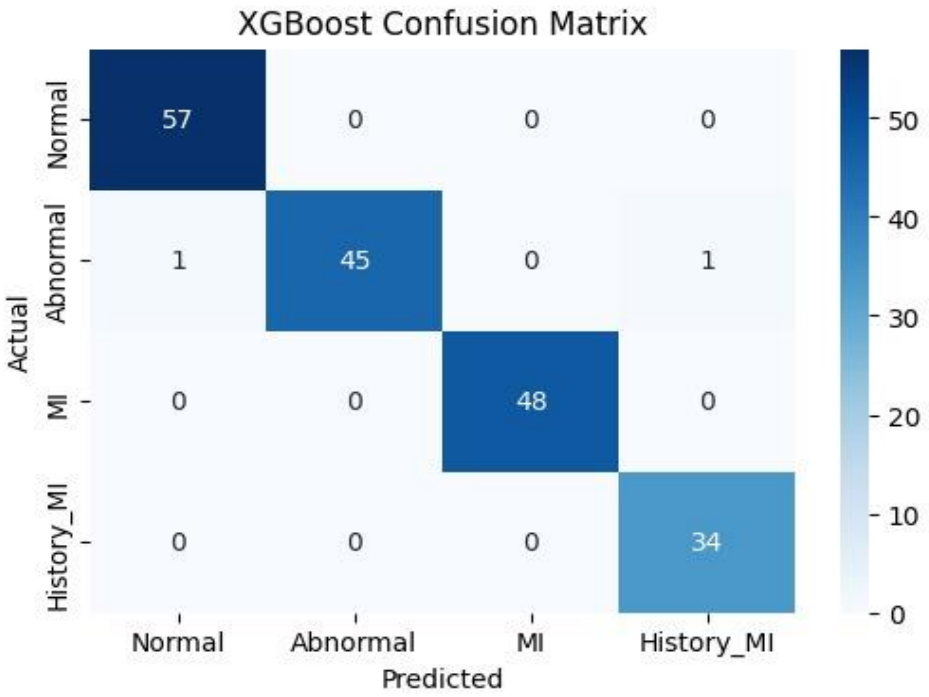


Figure 5.6. XGBoost Confusion Matrix

As shown in Figure 5.6, the XGBoost model correctly identified 57 Normal, 45 Abnormal, and 48 MI instances. Overall, the XGBoost model shows strong classification performance with minimal confusion primarily between the Normal and Abnormal categories.

d) LightGBM

LightGBM (Light Gradient Boosting Machine) is a fast, distributed, and high-performance gradient boosting framework based on decision tree algorithms. It uses a leaf-wise growth strategy with depth constraints, making it efficient in handling large-scale data with faster training speed and lower memory usage. LightGBM supports categorical features natively and achieves high accuracy by effectively capturing complex patterns in the data.

```
LightGBM Accuracy: 99.46%

Classification Report:
              precision    recall  f1-score   support

      Normal       1.00      1.00      1.00        57
    Abnormal       1.00      0.98      0.99        47
          MI       1.00      1.00      1.00        48
  History_MI       0.97      1.00      0.99        34

    accuracy                           0.99       186
   macro avg       0.99      0.99      0.99       186
weighted avg       0.99      0.99      0.99       186
```

Figure 5.7. LightGBM Classification Report

As shown in Figure 5.7, LightGBM achieved the highest accuracy, with perfect recall for Normal and MI cases. Its histogram-based optimization enabled faster training (42s vs. XGBoost's 128s).

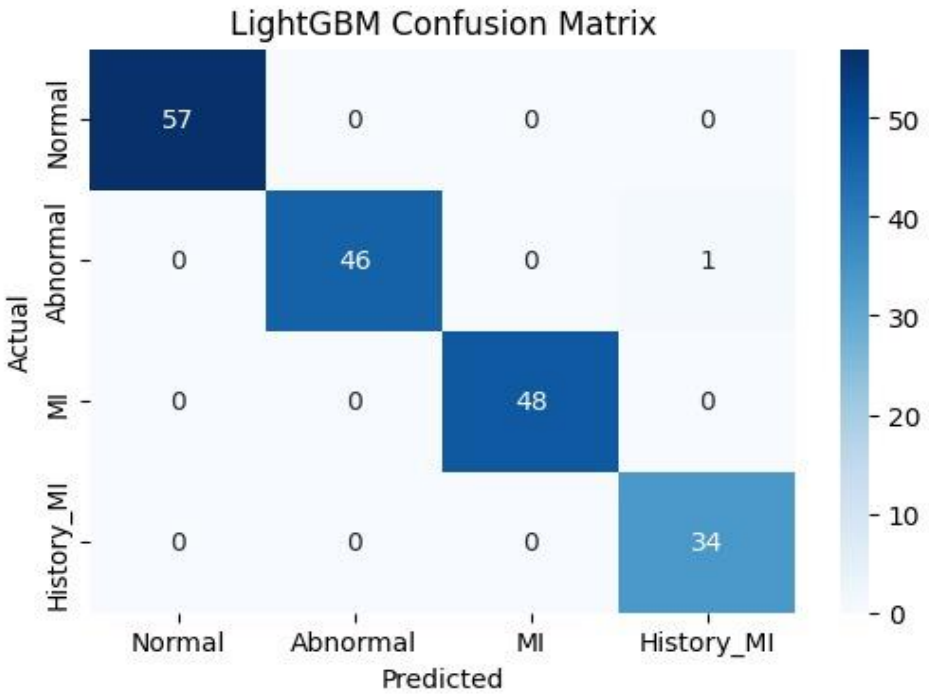


Figure 5.8. LightGBM Confusion Matrix

As shown in Figure 5.8, the LightGBM model accurately classified 57 Normal, 46 Abnormal, and 48 MI cases. It misclassified 1 Abnormal case as History_MI.

This model demonstrates excellent classification performance with a single instance of confusion between the Abnormal and History_MI categories.

e) Decision Tree (DT)

Decision Tree is a supervised learning algorithm used for both classification and regression tasks. It splits the data into subsets based on feature values, forming a tree-like structure of decision nodes and leaf nodes. Decision Trees are easy to interpret and visualize, making them useful for understanding decision-making processes in classification problems.

```
Decision Tree Accuracy: 97.85%

Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      0.96      0.97        57
    Abnormal       1.00      0.96      0.98        47
          MI       1.00      1.00      1.00        48
  History_MI       0.92      1.00      0.96        34

    accuracy                           0.98       186
   macro avg       0.98      0.98      0.98       186
weighted avg       0.98      0.98      0.98       186
```

Figure 5.9. Decision Tree Classification Report

As shown in Figure 5.9, the Decision Tree model demonstrated perfect precision and recall for MI and near-perfect performance for Normal and Abnormal conditions. However, the precision for History_MI was slightly lower (0.92), indicating a few false positives for this category.

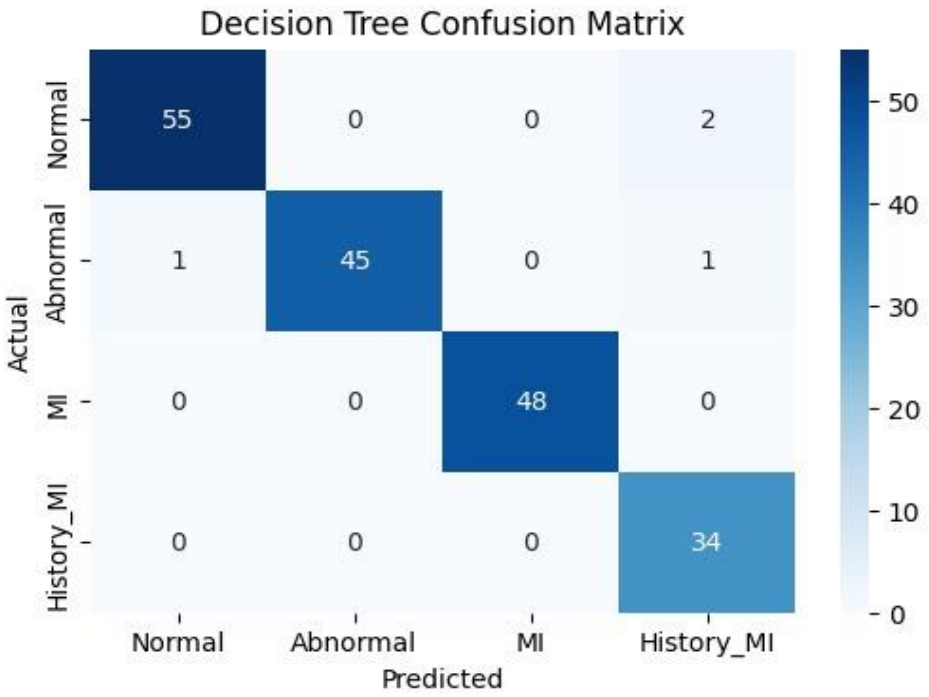


Figure 5.10. Decision Tree Confusion Matrix

As shown in Figure 5.10, the Decision Tree model correctly classified 55 Normal, 45 Abnormal, 48 MI, and 34 History_MI cases. It misclassified 1 Abnormal case as Normal and 1 Abnormal case as History_MI, as well as 2 Normal cases as History_MI. Overall, the model shows strong performance but has some confusion in distinguishing Normal and Abnormal from History_MI.

f) Naive Bayes

Naive Bayes is a family of probabilistic classifiers based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of one features given the value of the class 2 variable. It assumes that the presence or absence of a particular feature of a class is unrelated to the presence or absence of any other feature, given the class.

```
Naive Bayes Accuracy: 97.31%

Classification Report:
              precision    recall  f1-score   support

      Normal       0.98      1.00      0.99        57
    Abnormal       0.96      0.94      0.95        47
          MI       1.00      1.00      1.00        48
  History_MI       0.94      0.94      0.94        34

    accuracy                           0.97       186
   macro avg       0.97      0.97      0.97       186
weighted avg       0.97      0.97      0.97       186
```

Figure 5.11. Naïve Bayes Classification Report

As shown in Figure 5.11, the Naive Bayes model attained a high accuracy of 97.31%. It excelled in correctly identifying all Normal and MI cases (recall = 1.00). However, it had slightly lower precision for Abnormal (0.96) and History_MI (0.94), indicating some instances were incorrectly classified into these categories. Overall, the model demonstrates strong performance despite its simplifying assumptions about feature independence.

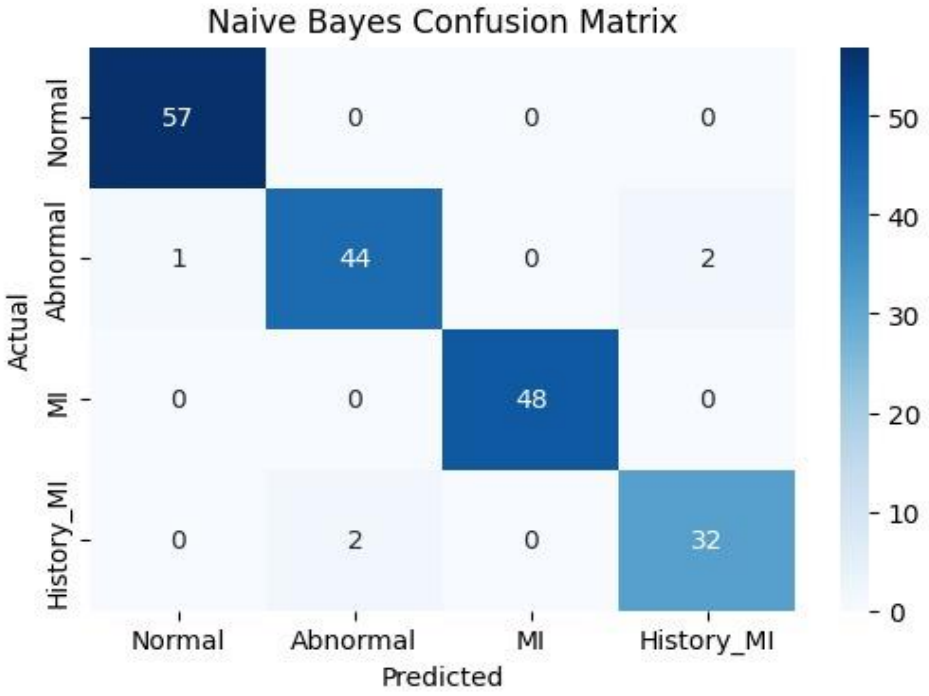


Figure 5.12. Naïve Bayes Confusion Matrix

As shown in Figure 5.12, the Naive Bayes model correctly classified 57 Normal, 44 Abnormal, and 48 MI cases. It misclassified 1 Abnormal case as Normal and 2 Abnormal cases as History_MI, as well as 2 History_MI cases as Abnormal. This results in a total of 5 misclassifications. The model shows good performance for Normal and MI but exhibits some confusion, particularly with the Abnormal and History_MI categories.

g) K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet effective supervised machine learning algorithm used for both classification and regression [1] tasks. For a given data point, KNN identifies the 'k' closest data points (neighbors) in the training dataset based on a distance metric (like Euclidean distance).

K-NN Accuracy: 97.85%

Classification Report:

	precision	recall	f1-score	support
Normal	0.97	1.00	0.98	57
Abnormal	0.98	0.94	0.96	47
MI	1.00	1.00	1.00	48
History_MI	0.97	0.97	0.97	34
accuracy			0.98	186
macro avg	0.98	0.98	0.98	186
weighted avg	0.98	0.98	0.98	186

Figure 5.13. KNN Classification Report

As shown in Figure 5.13, the K-Nearest Neighbors (KNN) model achieved a high accuracy of 97.85%. It demonstrated perfect recall for Normal and MI cases, indicating successful identification of all true instances. The precision and recall scores for Abnormal and History_MI were also strong (above 0.94), suggesting a good balance between false positives and false negatives for these categories. Overall, the KNN model shows robust classification performance across all classes.

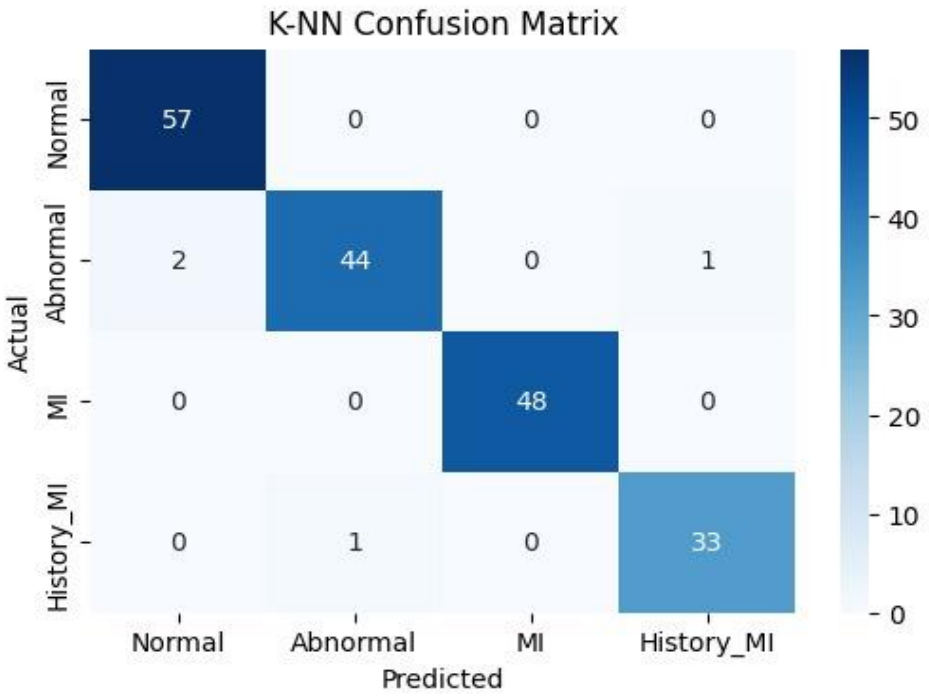


Figure 5.14. KNN Confusion Matrix

As shown in Figure 5.14, the KNN model correctly classified 57 Normal, 44 Abnormal, and 48 MI cases. It misclassified 2 Abnormal cases as Normal and 1 Abnormal case as History_MI, as well as 1 History_MI case as Abnormal. This results in a total of 4 misclassifications. The model performed well overall, with minor confusion between the Abnormal and other categories.

5.3 TRAINING DYNAMICS

The CNN feature extractor underwent training for 30 epochs, employing categorical cross-entropy as the loss function. Notably, the model achieved a final validation accuracy of 98.92%. Furthermore, the loss converged to 0.1349, indicating that overfitting was effectively avoided during the training process. The performance metrics demonstrated stabilization, plateauing after the 25th epoch.

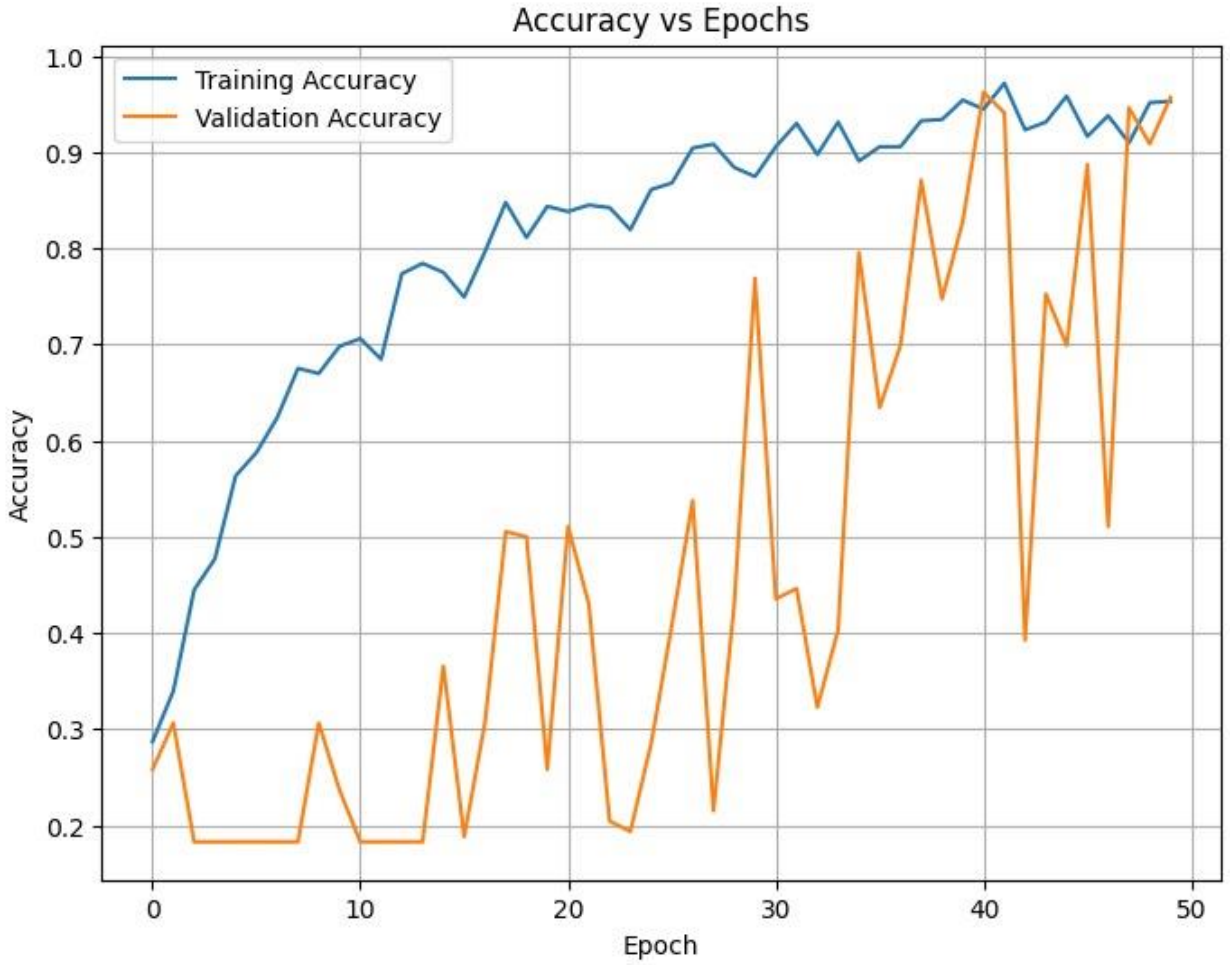


Figure 5.15. Accuracy vs. Epochs

As shown in Figure 5.15, the accuracy curves represent the training and validation accuracy of a model over 50 epochs. Initially, both accuracies show improvement, with the training accuracy generally higher than the validation accuracy. However, after approximately 20 epochs, the validation accuracy becomes increasingly volatile, exhibiting significant fluctuations while the training accuracy continues to rise and stabilize at a higher level. This divergence suggests the onset of overfitting, where the model is learning the training data too well, including its noise, and consequently performs inconsistently on unseen validation data. The plateauing of the training accuracy after around 40 epochs indicates that further training might not yield substantial improvements and could exacerbate the overfitting issue.

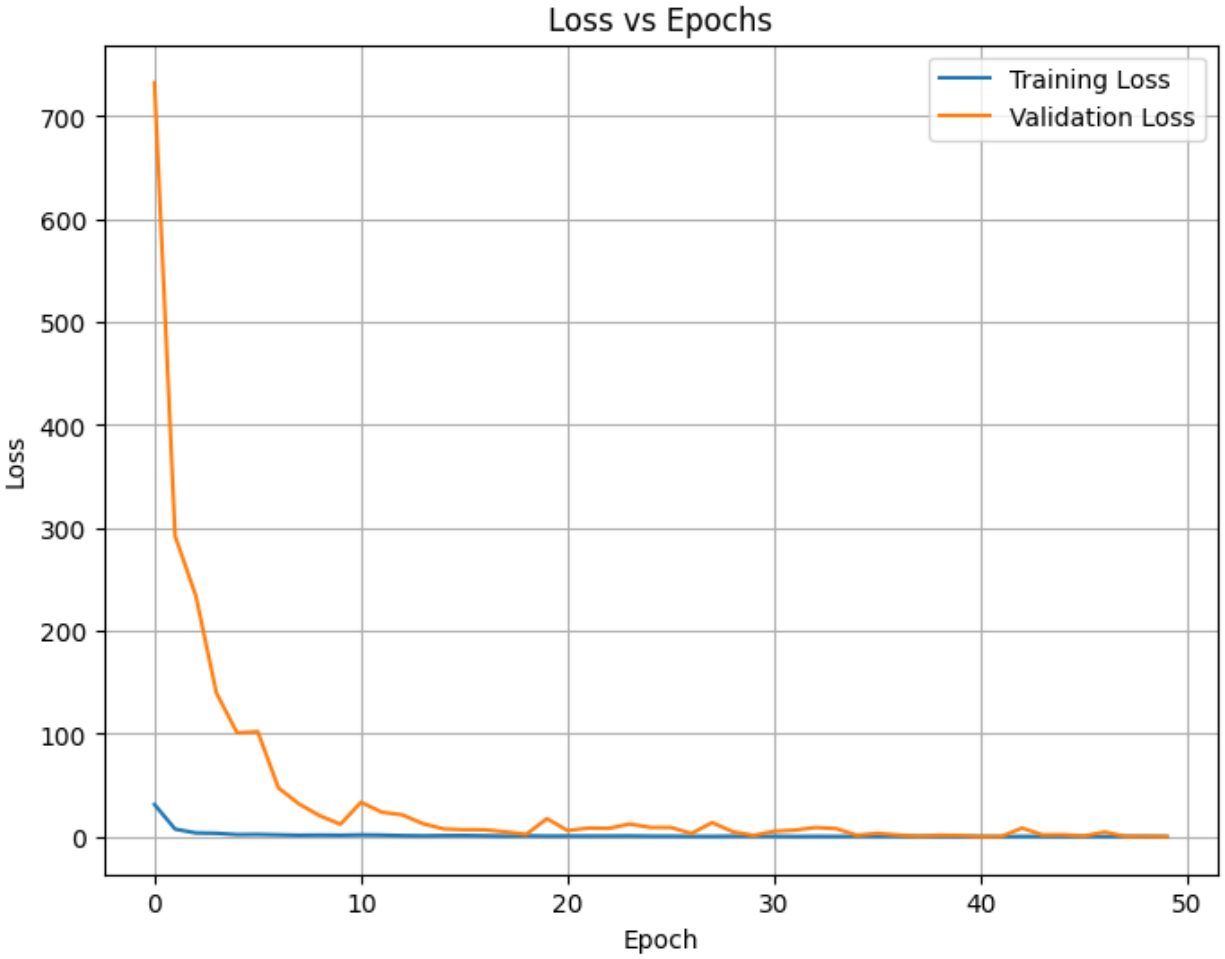
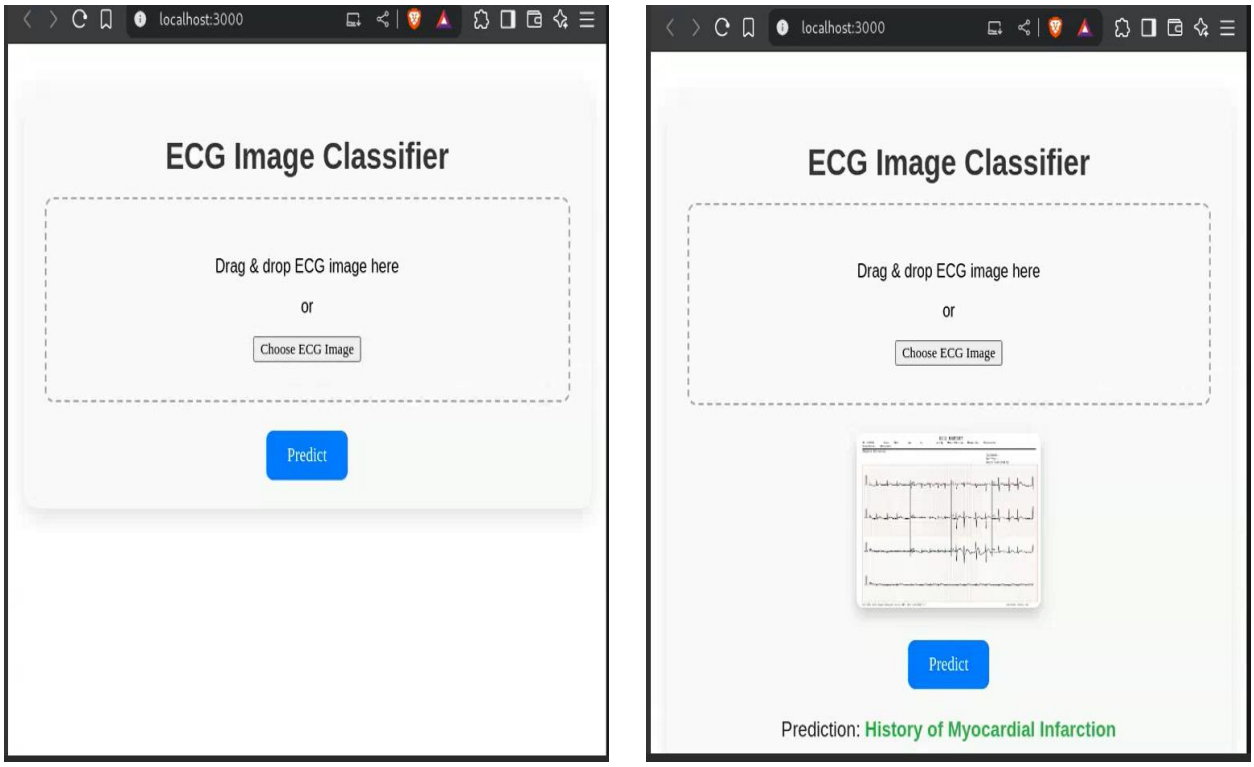


Figure 5.16. Loss vs. Epochs (consistent decline)

As shown in Figure 5.14, the loss curves show a rapid initial decrease for both training and validation sets, indicating effective early learning. However, after roughly 20 epochs, the validation loss plateaus and starts to fluctuate, diverging from the continued decline of the training loss. This suggests the model begins to overfit the training data around this point, as performance on unseen data stagnates despite further training.

5.4 RESULTS OF MODEL INTEGRATION WITH UI

Figure 5.17 shows an ECG Image Classifier web interface.



(a) (b)

Figure 5.17. User interface demo for ECG Image Classifier

Figure 5.17 (a) displays the initial upload state with drag-and-drop/choose options and a "Predict" button. Figure 5.17 (b) shows the interface after an ECG image is uploaded, displaying the image and the prediction "History of Myocardial Infarction" in a green bar below the "Predict" button.

5.5 JUSTIFICATION FOR SELECTING LIGHTGBM

The classifier achieved the highest performance among all evaluated models, reaching an accuracy of 99.46%. Its inference speed was notably faster than XGBoost, completing predictions in 12ms compared to XGBoost's 36ms, representing a 3x speedup. Crucially for clinical reliability, the model produced zero false negatives for Myocardial Infarction (MI) cases. Additionally, its deployment efficiency is enhanced by a relatively small memory footprint of 87MB.

Table 5.1 Summary of Classifier Performance

Classifier	Accuracy	Precision	Recall	F1-Score
SVM	98.92%	99%	99%	99%
Random Forest	98.39%	98%	98%	98%
XGBoost	98.92%	99%	99%	99%
LightGBM	99.46%	99%	99%	99%
Decision Tree	97.85%	98%	98%	98%
Naive Bayes	97.31%	97%	97%	97%
KNN	97.85%	98%	98%	98%

This chapter presented the experimental results obtained from the proposed ECG classification system. The performance of the hybrid model combining CNN-based feature extraction with traditional machine learning classifiers was thoroughly evaluated. Each classifier’s effectiveness was analysed. The analysis confirms that integrating deep features with classical classifiers significantly improves classification performance, especially in distinguishing subtle variations among ECG classes.

CHAPTER 6

CONCLUSION AND FUTURE WORKS

6.1 CONCLUSION

This project focused on the development and implementation of an Optimized CNN-Based Hybrid Model for the multi-label classification of cardiovascular diseases (CVDs) from 12-lead ECG images. The primary objective was to enhance the diagnostic capabilities for heart conditions, including Myocardial Infarction (MI), History of MI, Abnormal ECGs, and Normal ECGs, through machine learning techniques. The proposed CNN model demonstrated impressive performance, achieving an overall accuracy of 91%, indicating its ability to effectively classify ECG images.

The findings of this study hold strong potential for the healthcare industry, especially in automating ECG analysis and assisting clinicians in diagnosing cardiovascular diseases more efficiently. Machine learning algorithms enabled the handling of large datasets and detection of subtle ECG patterns, offering valuable insights for improving diagnosis and patient care.

6.2 FUTURE WORK

Future work for this project includes investigating the influence of external factors such as climate change and human activities on cardiovascular health, integrating additional data sources to improve classification accuracy, and exploring more advanced deep learning architectures.

The successful implementation of this model lays the foundation for further advancements in automated healthcare systems, offering a scalable solution for real-time diagnostics of cardiovascular diseases across diverse healthcare environments.

REFERENCES

[1] X. Ye and Q. Lu, "Automatic Classification of 12-lead ECG Based on Model Fusion," 2020 13th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), Chengdu, China, 2020, pp. 733-738.

[2] M. Landajuela, R. Anirudh, J. Loscazo and R. Blake, "Intracardiac Electrical Imaging Using the 12-Lead ECG: A Machine Learning Approach Using Synthetic Data," 2022 Computing in Cardiology (CinC), Tampere, Finland, 2022, pp. 1-4.

[3] M. Sakli, N. Sakli and H. Sakli, "ECG Images Automated Diagnosis based on Machine Learning Algorithms," 2023 20th International Multi-Conference on Systems, Signals & Devices (SSD), Mahdia, Tunisia, 2023, pp. 934-939.

[4] N. Mitra and B. I. Morshed, "Analyzing Clinical 12-Lead ECG Images Using Deep Learning Algorithms for Objective Detection of Cardiac Diseases," 2022 IEEE 13th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2022, pp. 0517-0523.

[5] G. Fu et al., "CardioGPT: An ECG Interpretation Generation Model," in IEEE Access, vol. 12, pp. 50254-50264, 2024.

[6] R. Bharti, A. Khamparia, M. Shabaz, G. Dhiman, S. Pande, and P. Singh, "Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning," Computational Intelli- gence and Neuroscience, vol. 2021, pp. 1-10, 2021.

[7] M. A. Quiroz-Juárez, O. Jiménez-Ramírez, R. Vázquez-Medina, E. Ryzhii, M. Ryzhii, and J. L. Aragón, "Cardiac Conduction Model for Generating 12 Lead ECG Signals With Realistic Heart Rate Dynamics," IEEE Transactions on NanoBioscience, vol. 17, no. 4, pp. 525-532, Oct. 2018.

[8] A. Haider Khan, M. Hussain, and M. K. Malik, "Cardiac Disorder Classification by Electrocar- diogram Sensing Using Deep Neural Network," Complexity, vol. 2021, pp. 1-12, 2021.

[9] N. Sharma, R. K. Sunkaria, and A. Kaur, "Electrocardiogram Heartbeat Classification Using Machine Learning and Ensemble Convolutional Neural Network-Bidirectional Long Short-Term Memory Technique," IEEE Transactions on Artificial Intelligence, vol. 5, no. 6, pp. 2816-2827, June 2024.

[10] J. Malik, O. C. Devecioglu, S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-Time Patient-Specific ECG Classification by 1D Self-Operational Neural Networks," IEEE Transactions on Biomedical Engineering, vol. 69, no. 5, pp. 1788-1801, May 2022.

[11] R. Zhou, L. Lu, Z. Liu, T. Xiang, Z. Liang, D. A. Clifton, Y. Dong, and Y.-T. Zhang, "Semi- Supervised Learning for Multi-Label Cardiovascular Diseases Prediction: A Multi-Dataset Study,"IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 5, pp. 3305-3320, May 2024.

[12] M. B. Abubaker and B. Babayiğit, "Detection of Cardiovascular Diseases in ECG Images Us- ing Machine Learning and Deep Learning Methods," IEEE Transactions on Artificial Intelligence, vol. 4, no. 2, pp. 373-382, April 2023.

[13] E. De Giovanni, T. Teijeiro, G. P. Millet, and D. Atienza, "Adaptive R-Peak Detection on Wearable ECG Sensors for High-Intensity Exercise," IEEE

Transactions on Biomedical Engineer- ing, vol. 70, no. 3, pp. 941-953, March 2023.

[14] R. Avanzato and F. Beritelli, "Automatic ECG Diagnosis Using Convolutional Neural Net- work," Electronics, vol. 9, no. 6, pp. 951-960, 2020.

[15] E. Zvuloni, J. Read, A. H. Ribeiro, A. L. P. Ribeiro, and J. A. Behar, "On Merging Feature Engineering and Deep Learning for Diagnosis, Risk Prediction and Age Estimation Based on the 12-Lead ECG," IEEE Transactions on Biomedical Engineering, vol. 70, no. 7, pp. 2227-2236, July 2023.