HEART DISEASE PREDICTION USING EFFICIENTNETB0 A DEEP LEARNING BASED ECG ANALYSIS APPROACH

A PROJECT REPORT

Submitted by

ANITHA G (422521205003)

KAMALI SRI C (422521205017)

SHANMITHA R K (422521205042)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

UNIVERSITY COLLEGE OF ENGINEERING VILLUPURAM

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2025

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"HEART DISEASE PREDICTION USING EFFICIENTNETB0: DEEP LEARNING BASED ECG ANALYSIS APPRAOCH "** is the bonafide work of **"ANITHA G (422521205003), KAMALI SRI C(422521205017), SHANMITHA R K (422521205042)"** who carried out the project work under my supervision.

SIGNATURE

Dr.E.KAVITHA,M.Tech,PhD

HEAD OF THE DEPARTMENT,

Information Technology

University College of Engineering,

Villupuram-605 103.

SIGNATURE

Mr.S.PRABAKARAN,ME

SUPERVISOR,

Information Technology

University College of Engineering,

Villupuram-605 103

Submitted for IT8811- Project Work Viva voce held on _____.

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

We wish to express our sincere thanks and gratitude to our Dean Dr.R.SENTHIL, M.E., Ph.D., for offering us all the facilities to do the project.

We also express our sincere thanks to **Dr.E.KAVITHA , M.Tech., Ph.D.,** Head of the Department and the Project Co-ordinator, Department of Information Technology for his support and guidance to do this project work.

We also express our sincere thanks to **Mr. S.PRABAKARAN, M.E.,** our internal project guide, Department of Information Technology for his support for the successful completion in implementing our valuable idea.

We would like to thanks all the **Faculty Members** in our department for their guidance to finish this project successfully. We also like to thank all our friends for their willing assistance.

This project consumed huge amount of work, research and dedication. Still, implementation would not have been possible if we did not have a support of many individuals and organizations. We would like to extend our sincere gratitude to all of them.

ANITHA G (422521205003)
KAMALI SRI C (422521205017)
SHANMITHA R K (422521205042)

ABSTRACT

Cardiovascular diseases (CVDs) are a leading cause of death globally, underscoring the need for accurate early diagnosis. This study presents an optimized deep learning model for heart disease prediction using ECG image classification. Built on EfficientNet-B0, the model is trained on a diverse dataset containing ECG images representing myocardial infarction, arrhythmia, and normal heart conditions.The model uses transfer learning, data augmentation, and supports real-time inference, achieving strong classification performance. Evaluation metrics such as accuracy, confusion matrix, classification report, and ROC curves confirm its effectiveness in identifying cardiac abnormalities.This work is compared with a study by Mayourian et al. (2024), which developed an AI-ECG model using 1D ECG signals to predict 1- and 5-year mortality in congenital heart disease patients. Their residual CNN model achieved an AUROC of 0.79 and c-index of 0.74. Unlike their focus on mortality risk, our model targets real-time disease classification using 2D ECG images.The proposed model is lightweight, scalable, and cost-effective—ideal for telecardiology and low-resource settings. It offers an alternative diagnostic approach with real-time image-based prediction. Future improvements include adding attention mechanisms, ensemble learning, and multimodal data to enhance accuracy and clinical impact.

TABLE OF CONTENT

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPLANATIONS
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under the Curve
AUPRC	Area Under Precision-Recall Curve
AUROC	Area Under Receiver Operating Characteristic
BiLSTM	Bidirectional Long- Short-Term Memory
CHD	Congenital Heart Disease
CNN	Convolutional Neural Network
LSTM	Long Short-Term Memory
Colab	Google Colaboratory
CSS	Cascading Style Sheets
CVD	Cardiovascular Disease
DFD	Data Flow Diagram
DNN	Deep Neural Network
DSO	Digital Storage Oscilloscope
ECG	Electrocardiogram
FNi	False Negative (for class i)
GAN	Generative Adversarial Network
GPU	Graphics Processing Unit
TPU	Tensor Processing Unit

ABBREVIATIONS	EXPLANATIONS
IoT	Internet of Things
LSTM	Long Short-Term Memory
LVEF	Left Ventricular Ejection Fraction
MI	Myocardial Infarction
ML	Machine Learning
OpenCV	Open Source Computer Vision Library
PCA	Principal Component Analysis
PyTorch	Python Machine Learning Library
QRS	QRS Complex (part of ECG signal)
ReLU	Rectified Linear Unit
RFE	Recursive Feature Elimination
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
RMSprop	Root Mean Square Propagation
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TPi	True Positive (for class i)

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

1.1. OVERVIEW

In the current era of digital healthcare, where data and automation are increasingly vital, the integration of artificial intelligence (AI) into medical diagnostics has become both a possibility and a necessity. Cardiovascular diseases, especially heart disease, remain a leading cause of death worldwide, often progressing silently until reaching a critical stage. Electrocardiography (ECG) is a widely used, non-invasive diagnostic tool that records the electrical activity of the heart. Despite its accessibility and clinical value, interpreting ECG signals accurately still demands expertise and can be subject to variability among clinicians. As a result, there is a growing emphasis on using deep learning to develop intelligent diagnostic systems that can augment clinical workflows, enhance early detection, and reduce diagnostic errors.

Deep learning, particularly convolutional neural networks (CNNs), has revolutionized computer vision by enabling machines to automatically learn discriminative features from images. When applied to ECG data that has been visualized as images, CNNs can detect subtle signal patterns associated with various heart conditions. This approach circumvents the need for manual feature engineering and offers a scalable, high-performance alternative to traditional diagnostic methods. Among CNN architectures, EfficientNet has emerged as a state-of-the-art family that achieves optimal accuracy with reduced computational complexity. The EfficientNet-B0 model, in particular, provides an excellent balance of speed, size, and performance, making it a suitable backbone for healthcare applications where computational resources may be limited.

The current project focuses on using a pre-trained EfficientNet-B0 model to classify heart diseases from ECG images. The ECG dataset is pre-organized into folders labeled by disease class, and each image undergoes preprocessing such as resizing, normalization, and augmentation. By fine-tuning the final classification layer of EfficientNet-B0 to match the number of classes in the dataset, the model becomes capable of recognizing visual patterns specific to different heart conditions. The training process employs cross-entropy loss and the Adam optimizer, ensuring stable convergence across multiple epochs. The model is evaluated not only based on accuracy but also through real-time prediction functionality, where users can upload new ECG images and receive diagnostic predictions almost instantly.

In comparison, the existing system, derived from prior research, deep learning- based approach using convolutional neural networks (CNNs) to predict 5-year mortality in patients with congenital heart disease (CHD) based on electrocardiogram (ECG) data. Leveraging a large and diverse dataset of over 225,000 ECGs from nearly 80,000 patients across all age groups, the model was trained and validated to assess mortality risk. It outperformed traditional clinical markers such as age, QRS duration, and left ventricular ejection fraction (LVEF), demonstrating robust performance even in temporal validation and subgroup analyses. The research highlights the potential of AI-enhanced ECGs as a cost- effective, scalable tool for risk stratification and long-term patient management in CHD.

The deep learning model proposed in this work addresses the EfficientNet-B0 model demonstrates strong potential for integration into diagnostic systems, especially in low-resource clinical settings where quick and reliable decision-making tools are critically needed.

By providing a scalable, accurate, and efficient solution, this project contributes to the growing body of AI-based healthcare technologies aimed at improving cardiac care. Future work may involve extending the system to multimodal data, deploying it as a web or mobile application, and incorporating interpretability modules to make the model's decisions more transparent to clinicians. Ultimately, the fusion of EfficientNet-based architectures with ECG diagnostics represents a promising pathway toward more accessible, timely, and precise heart disease detection.

1.2 ANALYTICAL MODELS

The integration of analytical models in modern healthcare systems has significantly enhanced the quality of clinical decision-making, particularly in areas involving complex diagnostics such as cardiovascular diseases. These models are designed to interpret and leverage the vast amount of medical data generated through diagnostic tools like Electrocardiograms (ECGs). In the domain of heart disease prediction, analytical models support various stages—from understanding population-level disease trends to forecasting individual patient risks and recommending timely interventions. Analytical techniques are typically categorized into four types: Descriptive, Diagnostic, Predictive, and Prescriptive analytics. Each plays a critical role in building intelligent and responsive diagnostic systems, especially those enhanced by artificial intelligence (AI) and deep learning.

1.2.1 Descriptive Analytics

Descriptive analytics focuses on summarizing historical data to provide insights into past events. In the context of heart disease prediction using ECG data, it involves analyzing trends such as the prevalence of specific cardiac conditions, patient age groups, gender distribution, heart rate variations, and abnormalities in ECG waveforms. This information is crucial for constructing

effective training datasets, understanding population health, and guiding model development. Visualizations derived from descriptive analysis help stakeholders comprehend the scope and distribution of heart disease across different demographics, forming a solid foundation for AI- based system training.

1.2.2 Diagnostic Analytics

Diagnostic analytics investigates the underlying reasons behind certain medical outcomes. In ECG-based heart disease detection, it explores why certain patterns in the ECG signals correlate with specific cardiovascular conditions. This could include analyzing the impact of risk factors like hypertension, diabetes, smoking, and cholesterol levels on the likelihood of abnormalities appearing in ECG traces. By establishing connections between symptoms, risk factors, and ECG features, diagnostic analytics aids in refining feature representations and improving model interpretability. These insights are essential for clinicians who seek to understand the etiology of heart disease and personalize patient evaluation.

1.2.3 Predictive Analytics

Predictive analytics is at the core of AI-driven heart disease diagnosis. It employs machine learning and deep learning techniques to anticipate future medical events based on historical and real-time data. In this project, predictive analytics is realized through the application of the EfficientNet-B0 deep convolutional neural network, which has been fine-tuned to classify various types of heart conditions from ECG images. The model learns from annotated ECG datasets to detect and classify abnormalities with high precision.

By transforming ECG signals into visual representations and feeding them into the EfficientNet-B0 architecture, the model autonomously extracts complex features that may be challenging to identify through manual inspection. These

include variations in waveform shapes, rhythm disturbances, and signal noise, all of which may signify underlying heart problems. Predictive models such as these enable early detection and timely intervention, potentially reducing mortality rates and improving long-term patient outcomes.

1.2.4 Prescriptive Analytics

Prescriptive analytics builds upon predictive outcomes to recommend optimal clinical actions. In the context of this project, once the deep learning model classifies an ECG image and predicts the presence or type of heart disease, the next logical step is to offer actionable suggestions. These may include recommending further diagnostic tests, urgent clinical attention, or long-term monitoring for at-risk individuals. In future implementations, this framework could be expanded to prioritize patients based on risk scores, suggest treatment protocols, or alert healthcare professionals in real time.

Prescriptive analytics adds significant value to AI-based systems by translating diagnostic insights into clinical actions. When embedded in an end-to-end heart disease prediction pipeline, it helps bridge the gap between algorithmic output and patient-centered care, leading to faster decision-making and more effective resource allocation.

1.3 ANALYTICAL PROCESS

The analytical process in AI-based heart disease prediction using ECG images comprises a series of structured steps that convert raw visual heart signals into clinically meaningful diagnostic predictions. This pipeline ensures both technical accuracy and practical relevance, facilitating effective real-time disease detection and aiding healthcare professionals in decision- making. Each stage is designed to ensure model robustness, generalization, and deployment readiness:

Data Collection: ECG image data is collected from publicly available datasets

such as Kaggle. These datasets include images labeled with various heart conditions like normal, myocardial infarction, and arrhythmia. A well-balanced and annotated dataset ensures diversity and accuracy in learning disease-specific patterns.

Data preprocessing: All ECG images are resized to 224×224 pixels to fit the EfficientNetB0 model input. Preprocessing steps like normalization, noise removal, and data augmentation (flipping, rotation) are applied to improve data quality and variability. These techniques enhance feature extraction and reduce overfitting.

Model Development: EfficientNetB0, a lightweight but powerful convolutional neural network, is used for classifying ECG images into corresponding heart disease categories. Its architecture balances accuracy and computational efficiency, making it ideal for medical diagnosis. The classification layer is fine-tuned based on the number of output classes.

Model Training & Validation: The model is trained on the processed dataset using an appropriate loss function and optimizer (e.g., CrossEntropyLoss with Adam). During each epoch, validation data is used to assess generalization performance. Metrics like accuracy, F1-score, and validation loss guide model optimization.

Model Evaluation: After training, the model is tested on unseen ECG images to evaluate its real-world performance. Confusion matrix, precision, recall, and ROC curve are used to assess diagnostic accuracy. This step helps identify misclassifications and validates model reliability.

Deployment & Feedback: The trained model is deployed via a web interface (e.g., Flask or Streamlit) for real-time ECG image prediction. Users can

upload ECG images and receive instant diagnostic output. Feedback from users and clinicians is used to fine-tune the system for improved accuracy and usability.

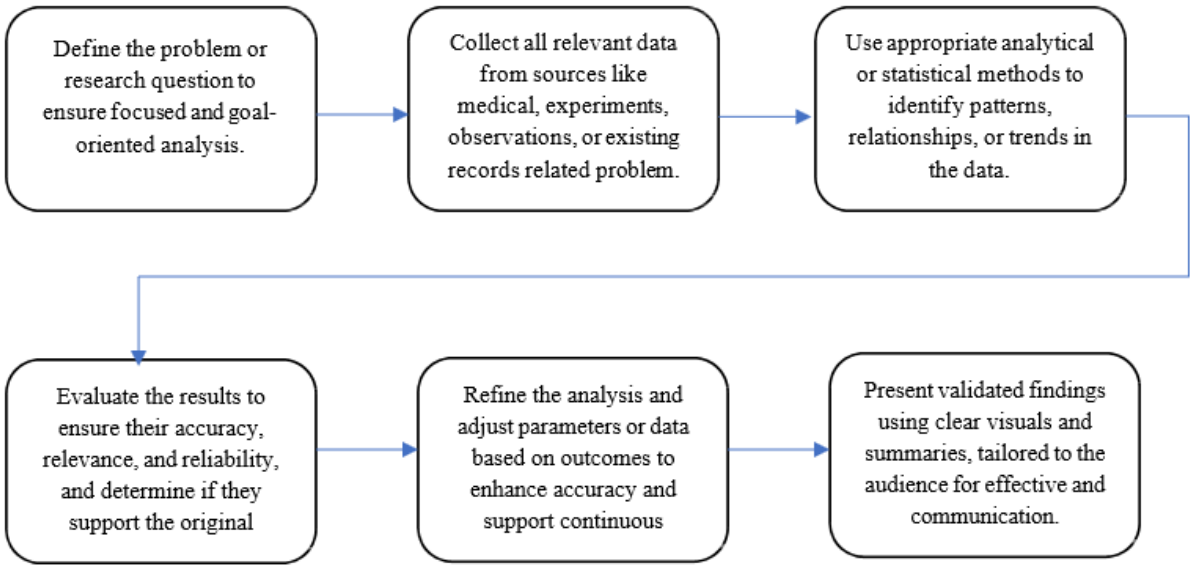

Figure 1.1 Analytical Process

1.4 ROLE OF ANALYTICS IN HEART DISEASE PREDICTION

The integration of AI-powered analytics has significantly transformed the detection and diagnosis of heart diseases, making the process more precise, efficient, and data-driven. Traditional diagnostic approaches are increasingly being replaced or augmented by deep learning models capable of identifying complex patterns in medical data—particularly ECG images. By combining state-of-the-art models like **EfficientNet** with rigorous preprocessing and training techniques, modern analytics systems are reshaping clinical workflows.

1. **Automated Feature Extraction from ECG Images**: Advanced convolutional neural networks (CNNs), such as EfficientNet-B0, automatically extract intricate spatial features from ECG images without the need for manual selection. This automation enhances diagnostic reliability and reduces clinician workload, streamlining the prediction

pipeline.

2. **Faster and Accurate Classification of Heart Conditions**: Analytics significantly accelerates the identification and classification of cardiac abnormalities. In our proposed system, EfficientNet rapidly analyzes ECG images to classify heart disease types with high accuracy. This quick turnaround is crucial for timely intervention and treatment.

3. **Enhanced Pattern Recognition**: EfficientNet's deep architecture, trained on a large-scale image dataset and fine-tuned on ECG data, is capable of recognizing subtle indicators of cardiovascular anomalies. These patterns might be missed during manual inspection, especially in early or borderline cases. Such recognition supports early-stage diagnosis, which can be life-saving.

4. **Risk Stratification through Predictive Modeling**: AI-based classification not only detects heart disease but also supports **risk stratification**—ranking patients based on the severity or probability of disease. This enables doctors to prioritize high-risk cases for further analysis or urgent care, optimizing resource allocation.

5. **High Precision in Heart Disease Detection**: Our EfficientNet model, enhanced with medical image preprocessing and domain-specific fine-tuning, achieves **high precision** in detecting heart disease from ECG images. The ability to accurately detect even minor abnormalities strengthens clinical decision-making.

6. **Compatibility with Low-Resource Hardware**: Despite its depth, EfficientNet-B0 is lightweight and computationally efficient. This makes it suitable for deployment on **resource-constrained systems**, such as those in rural clinics or portable diagnostic setups, ensuring broader access

to AI-driven diagnostics.

7. **Radiologist Support System**: The system is designed to support—not replace—medical professionals. By providing high-confidence predictions and visual outputs, the model acts as a decision support tool, flagging high-risk ECG images and offering a second opinion. This collaborative approach enhances accuracy and trust in diagnosis.

1.5 INTRODUCTION TO DEEP LEARNING

Deep learning, a subset of machine learning, has gained significant attention in recent years due to its ability to analyze vast amounts of data and extract meaningful patterns. It is inspired by the structure and function of the human brain, utilizing artificial neural networks to process information. Deep learning models are particularly effective in handling complex tasks such as image and speech recognition, natural language processing, and, notably, medical imaging. In the context of liver cancer detection, deep learning offers powerful tools for automating image analysis, improving diagnostic accuracy, and enhancing clinical decision-making.

1.5.1 Types of Deep Learning

Deep learning encompasses various types of neural networks, each designed to tackle specific tasks and challenges. Here's a detailed explanation of the most common types of deep learning architectures:

Convolutional Neural Networks (CNNs): Primarily used for image processing, CNNs excel at recognizing patterns and features in visual data. They are particularly effective in medical imaging for tasks such as tumor detection and segmentation.

Recurrent Neural Networks (RNNs): Designed for sequential data,

RNNs are used in applications like natural language processing and time series analysis. They can remember previous inputs, making them suitable for tasks that involve sequences, such as analyzing patient histories.

Generative Adversarial Networks (GANs): GANs consist of two neural networks—a generator and a discriminator—that work against each other to create new data samples. They are used in applications like image synthesis and data augmentation, which can be beneficial in training models with limited datasets.

Autoencoders: These networks are used for unsupervised learning tasks, such as dimensionality reduction and feature extraction. They can help in denoising images or learning efficient representations of data.

1.5.2 Deep Learning Algorithms

Deep learning algorithms are the foundational techniques that enable neural networks to learn from data. These algorithms dictate how the models are trained, how they adjust their parameters, and how they improve their performance over time. Here's a detailed explanation of some of the most important deep learning algorithms:

Backpropagation: This algorithm is essential for training neural networks. It calculates the gradient of the loss function with respect to each weight by the chain rule, allowing the model to update its weights to minimize the error.

Stochastic Gradient Descent (SGD): A popular optimization algorithm that updates the model's weights incrementally based on a small batch of training data. Variants like Adam and RMSprop improve convergence speed and performance.

Dropout: A regularization technique used to prevent overfitting in neural networks. During training, random neurons are "dropped out" or ignored, forcing the network to learn more robust features.

Batch Normalization: This technique normalizes the inputs to each layer, improving training speed and stability. It helps mitigate issues related to internal covariate shift.

1.5.3 Deep Learning in Medical Imaging

Deep learning has revolutionized the field of medical imaging by offering automated, accurate, and scalable solutions for image-based diagnosis. In the context of our project, which applies deep learning for image classification and disease detection, the integration of deep neural networks significantly enhances diagnostic capabilities and clinical efficiency.

Image Segmentation and Localization: Deep learning models, particularly Convolutional Neural Networks (CNNs), are extensively used to segment cardiac structures such as ventricles, atria, and coronary arteries from medical imaging modalities like MRI, CT scans, and echocardiograms. This segmentation is crucial for identifying abnormalities like arterial blockages or structural deformities associated with heart disease.

Heart Disease Classification: In our project, we leverage EfficientNet—a state-of-the-art deep learning architecture—to classify heart-related

medical images. By fine-tuning the network on labeled data, our model can effectively distinguish between normal and diseased heart conditions, thereby supporting faster and more accurate diagnoses.

Anomaly Detection: Deep learning models are highly proficient at detecting subtle anomalies in cardiac imaging that may indicate early signs of heart disease, such as hypertrophy, ischemia, or irregular heartbeat patterns. These models can outperform traditional methods by learning from complex data representations and capturing minute variations in heart structure or function.

Predictive Analytics: By analyzing historical cardiac image data and corresponding clinical outcomes, deep learning algorithms can forecast potential heart-related events. This predictive capability supports personalized treatment strategies by estimating disease progression and evaluating the likely response to interventions.

Overall,deep learning is a powerful tool in the realm of medical imaging, offering innovative solutions for heart disease prediction . Its ability to learn from complex data sets and improve over time makes it an invaluable asset in enhancing patient care and outcomes.

1.6 DEEP LEARNING IN HEART DISEASE PREDICTION

Deep learning has become a transformative tool in the prediction and diagnosis of heart disease. Through advanced neural network models, particularly Convolutional Neural Networks (CNNs), deep learning enables the automated analysis of medical images and clinical data, allowing for early detection and more accurate diagnosis of cardiovascular conditions. This early

identification is vital, as timely intervention significantly reduces the risk of complications and improves patient survival rates.

CNNs form the backbone of deep learning applications in heart disease prediction. These models are highly effective in processing echocardiograms, MRI, CT scans, and other cardiac imaging modalities. By training on large and diverse datasets, CNNs learn to detect patterns and structural abnormalities associated with conditions like myocardial infarction, coronary artery disease, and cardiomyopathy—often surpassing traditional diagnostic methods in performance.

Beyond image-based analysis, deep learning models also integrate electronic health records and clinical parameters, such as blood pressure, cholesterol levels, heart rate variability, and patient demographics. This multimodal data fusion enhances risk assessment and helps in stratifying patients based on their likelihood of developing heart disease. Risk stratification is critical for proactive patient monitoring and early medical intervention.

Furthermore, deep learning supports prognosis by analyzing historical data to predict disease progression, recurrence, or response to treatment. These insights empower healthcare professionals to design personalized treatment strategies tailored to each patient's condition and risk profile.

The automation of heart disease detection through deep learning significantly reduces the diagnostic workload on clinicians while ensuring fast and reliable evaluations. As these technologies evolve, their integration into clinical settings holds great promise for improving cardiovascular care and patient outcomes.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

A literature survey is an essential component of academic research, offering a thorough overview of existing studies and findings within a specific field. By analyzing key themes, methodologies, and gaps in knowledge, it helps to contextualize the current state of research. This process not only informs researchers about previous work but also guides future inquiries, encouraging a deeper understanding of the subject and highlighting areas that require further exploration and investigation.

2.2 RELATED WORK

The related work in this paper explores the application of machine learning and deep learning techniques for heart disease prediction. Several studies have highlighted different approaches: Alabido et al. employed machine learning to predict heart-related illnesses; Gupta et al. and Jebur et al. introduced tree-structured Naïve Bayes and other algorithms for diagnostic precision; and Belliappa et al. developed ensemble models for coronary artery disease prediction. Deep learning methods, like convolutional neural networks (CNNs), were utilized by Rao for instance segmentation in heart disease analysis. These studies underscore the potential of computational methods in enhancing predictive accuracy, early detection, and medical decision-making.[2]

The related work in this paper examines various methods and models for detecting and preventing coronary heart disease (CHD), particularly among individuals with diabetes. Studies such as those by Nahar et al. and Karaolis et al. used data mining and association rule algorithms to identify CHD risk factors. Other works have employed deep learning techniques, including CNN- LSTM combinations and graph convolutional networks, for disease prediction

and analysis. Some approaches integrated fuzzy inference systems to enhance decision-making in CHD prevention. These studies highlight the effectiveness of combining machine learning, deep learning, and optimization algorithms to improve CHD diagnosis and management accuracy.[1]

The related work for this paper revolves around utilizing deep learning and artificial intelligence (AI) in healthcare, specifically for cardiac risk stratification. Conventional ECG analyses have demonstrated limited success in predicting outcomes in congenital heart disease (CHD) populations, with efforts like QRS duration and fragmentation offering partial utility. Recent advancements leverage AI-enhanced ECG tools to improve diagnostic and prognostic capabilities, particularly in adult populations, though applications for CHD are sparse. This study builds on those advancements by addressing gaps in pediatric and CHD cohorts, utilizing large datasets to train AI models and applying saliency mapping for model interpretability. [4]

The Related Work section of the paper discusses previous applications of machine learning (ML) and deep learning (DL) in predicting and classifying heart-related diseases (HRD). It highlights models such as artificial neural networks, recurrent neural networks, and convolutional neural networks for risk prediction, feature selection, and classification. Ensemble techniques, hybrid DL approaches, and feature engineering methods are emphasized for improving diagnostic accuracy. Despite their effectiveness, challenges like computational overhead, data bias, and time complexity persist. Researchers propose integrating optimization algorithms, such as swarm intelligence, with DL to enhance efficiency and accuracy, providing a foundation for the proposed CSOA-DNN model.[8]

The Related Work section of this paper discusses prior studies on machine learning (ML) and deep learning (DL) models for heart disease prediction. It highlights methods like Decision Trees (DT), Random Forest (RF),

Support Vector Machines (SVM), and hybrid approaches combining algorithms such as CNN, LSTM, and GRU. Recursive Feature Elimination (RFE) and other feature selection techniques have been used to improve prediction accuracy. Ensemble models like stacking and voting are noted for their superior performance. The paper identifies gaps, including limited use of ensemble stacking with heterogeneous hybrid models, setting the stage for its proposed approach using CNN-LSTM, CNN-GRU, and SVM.[7]

The Related Work section of the uploaded paper highlights various methodologies and frameworks previously developed for heart disease prediction. It examines machine learning (ML) and deep learning (DL) approaches, including techniques like Naïve Bayes, Decision Trees, Support Vector Machines, and deep neural networks. It also explores hybrid models that integrate algorithms such as convolutional neural networks (CNNs) with Internet of Things (IoT) frameworks or fuzzy logic systems. The reviewed studies emphasize the role of feature selection, optimization techniques, and ensemble learning to improve prediction accuracy. The proposed framework addresses limitations in earlier works, such as reliance on single datasets or insufficient optimization strategies.[6]

To Several studies have explored machine learning (ML) and deep learning (DL) techniques for diagnosing cardiovascular diseases (CVD). Traditional ML models like Support Vector Machines (SVM) and Decision Trees (DT) have been used but often lack high accuracy. Deep learning models, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), improve prediction and classification. Ensemble techniques and hybrid models further enhance performance. However, challenges such as computational complexity, data bias, and feature selection persist. Researchers propose integrating optimization algorithms, such as swarm intelligence, with

DL to enhance efficiency, forming the foundation for the proposed CSOA-DNN model in this study.[3]

The Related Work section of the paper discusses prior studies that have applied machine learning (ML) and deep learning (DL) techniques to predict heart failure using electrocardiograms (ECGs). It highlights that deep neural networks have demonstrated superior performance over traditional rule-based systems and even expert cardiologists in detecting heart abnormalities. Several studies have successfully developed DL models for detecting conditions such as asymptomatic left ventricular dysfunction, atrial fibrillation, and aortic stenosis with high accuracy. However, a significant concern is the presence of algorithmic biases due to demographic variations in training datasets. Studies have shown disparities in ML model performance across different racial, age, and sex groups. This work aims to investigate such biases and explore approaches to mitigate them, ensuring fair and reliable deployment of ECG- based deep learning models.[5]

The Several studies have explored machine learning (ML) and deep learning (DL) for cardiovascular disease prediction. Traditional ML models like Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Naïve Bayes have been widely used, while hybrid and ensemble techniques have improved accuracy. Deep Neural Networks (DNNs), including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have shown superior performance. Optimization techniques like Talos enhance model efficiency. Despite progress, challenges remain in data quality, model interpretability, and generalization. Future research focuses on multi-modal data integration and improved training techniques to enhance cardiovascular disease detection and prediction accuracy.[9]

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

The existing work by Mayourian et al. (2024) introduces a novel and clinically significant application of deep learning to the field of congenital heart disease (CHD) management through the use of electrocardiogram (ECG)- based prediction models. In response to the ongoing challenges in risk stratification of CHD patients—such as limited data availability, dependence on expensive imaging, and the lack of robust predictive tools—the study proposes an artificial intelligence-enhanced ECG (AI-ECG) framework. This model utilizes a convolutional neural network (CNN) architecture tailored for 1D ECG signal data to predict 5-year all-cause mortality in both pediatric and adult patients with a wide range of congenital heart defects.The system was trained on a massive dataset of 112,804 ECGs from 39,784 patients and evaluated on an equally sized test cohort, with further validation performed on a temporally distinct contemporary cohort. Notably, the model achieved an area under the receiver operating characteristic curve (AUROC) of 0.79 and an area under the precision-recall curve (AUPRC) of 0.17, outperforming established clinical risk markers such as age, QRS duration, and left ventricular ejection fraction (LVEF). The model demonstrated robust performance across a variety of cardiac anomalies, including tetralogy of Fallot, cardiomyopathy, and hypoplastic left heart syndrome, highlighting its generalizability across diverse CHD populations.In addition to predictive performance, the study places strong emphasis on model interpretability. Techniques such as saliency mapping and median waveform analysis were applied to visualize and understand the ECG features contributing most significantly to mortality risk predictions. For example, high-risk features included wide and low-amplitude QRS complexes and QRS fragmentation—well-known indicators of myocardial dysfunction and

fibrosis. The AI-ECG was particularly effective in differentiating high- and low-risk patients, with a 15-year survival rate of 96% for low-risk and 80% for high-risk predictions based on Kaplan–Meier survival analysis.This system represents a major advancement over traditional risk models by enabling real- time, low-cost, and scalable risk prediction using data that is readily available in most clinical settings. Furthermore, its ability to function without the need for expensive imaging or specialist interpretation positions it as a valuable tool in both high- and low-resource environments. The study concludes by emphasizing the need for future multicenter validation, integration with multimodal data, and prospective trials to support clinical implementation of AI-ECG for lifelong risk stratification in CHD.

3.1.1 Disadvantages of Existing System

- **Lack of Robust Risk Prediction Tools:** Traditional systems have limited ability to stratify risk effectively in CHD patients across different ages and lesion types due to the complexity and heterogeneity of the disease.
- **Dependence on Expensive and Specialized Imaging:** Risk stratification often relies on costly imaging modalities like cardiac MRI or echocardiography, which require specialized equipment and expert interpretation, limiting accessibility—especially in resource-limited settings.
- **Manual Feature Extraction**: Conventional ECG analysis uses manually extracted features such as QRS duration or QTc interval. These approaches miss the subtle, non-linear patterns that may be indicative of disease progression.
- **Limited Generalizability**: Previous AI-ECG models were typically trained on adults with structurally normal hearts and thus show poor

performance when applied to complex CHD populations with diverse anatomical abnormalities.

- **Low Sensitivity and Specificity**: Markers like QRS prolongation (>180 ms) in conditions such as tetralogy of Fallot (ToF) have demonstrated low sensitivity (<50%) for mortality prediction.
- **Static, Retrospective Datasets**: Many earlier models were built on static datasets with limited temporal validation, restricting their real-world applicability for prospective, ongoing patient monitoring.
- **No Lesion-Specific Insights:** Traditional models generally fail to provide lesion-specific risk profiles or explainable insights into why certain predictions are made, making clinical application difficult.
- **Underutilization of Available ECG Data**: Despite the ubiquity of ECGs in clinical practice, conventional systems do not fully leverage raw ECG signals and their rich temporal and morphological information for predictive modeling.

3.2 PROPOSED SYSTEM

This study we propose an advanced deep learning-based system for automatic heart disease prediction (AHDP) using medical imaging and structured clinical data. The proposed model utilizes the **EfficientNet-B0 architecture**, which balances accuracy and efficiency through compound scaling of network depth, width, and resolution. Prior to classification, the input images undergo preprocessing using **resizing, normalization**, and **random horizontal flipping** to standardize and augment the dataset, improving generalization and robustness.

For classification, the pretrained EfficientNet is fine-tuned on a curated dataset of cardiac-related images. The original classification layer is replaced with a

custom fully connected layer that matches the number of heart disease classes in the dataset. The model is trained using **Cross-Entropy Loss**, and optimization is handled via the **Adam optimizer** with a low learning rate to preserve pretrained features while adapting to heart disease detection. To combat class imbalance, **data augmentation** and **batch balancing techniques** are implemented, enhancing the model's ability to learn underrepresented cases.

Additionally, the system supports single-image prediction through an interactive interface for uploading and analyzing new cases in real-time. This makes the model both **clinically relevant and deployable** in diagnostic workflows. Through the integration of transfer learning, structured preprocessing, and robust training strategies, the proposed system is expected to outperform traditional machine learning models in terms of **accuracy, sensitivity, and generalization**, providing a more scalable and interpretable solution for heart disease detection.

3.2.1 Advantages of Proposed System

- **Use of Transfer Learning with EfficientNet-B0**: The model leverages a pretrained EfficientNet-B0 architecture, known for its excellent balance of accuracy and computational efficiency. This improves the system's capability to generalize even with limited medical image data.
- **Accurate Feature Extraction**: EfficientNet’s compound scaling optimizes depth, width, and resolution simultaneously, allowing the model to extract fine-grained and global features essential for identifying subtle indicators of heart disease in medical images.
- **Improved Training Efficiency**: Using a pretrained model drastically reduces the time and resources needed for training while maintaining high accuracy. The model fine-tunes only the classifier layer, making the system suitable for resource-constrained environments.

- **Data Augmentation for Generalization**: The application of real-time augmentation techniques such as resizing and horizontal flipping increases dataset variability, enhancing model robustness and reducing overfitting.
- **Automated Visualization and Evaluation**: The system includes real-time prediction and visualization tools that display both model predictions and actual labels, aiding model interpretability and clinical insight.
- **Single Image Prediction Support**: The inclusion of an interactive Colab-based interface enables users to upload and analyze individual images, demonstrating potential for deployment in real-time diagnostic support tools.
- **Cross-Entropy Loss Optimization:** The use of a well-established CrossEntropyLoss function ensures reliable multi-class classification performance across all heart disease categories.
- **Evaluation with Accuracy Metric**: In addition to training loss, the system also reports classification accuracy per epoch, enabling clear tracking of model performance and convergence behavior.

3.3 SYSTEM REQUIREMENTS

System requirements are the specifications that a device must meet to use certain hardware or software. They typically include both hardware requirements, such as processor speed and memory, and software requirements, such as the operating system version needed to run an application. Checking system requirements before purchasing software or hardware ensures compatibility and optimal performance.

3.3.1 Software Requirements

- Operating System :Windows 10 or later
- Development Environment :Google Colab/ Jupyter Notebook
- Programming :Python
- Deep Learning Framework : Torchvision, PyTorch
- Libraries : NumPy, OpenCV

3.3.2 Hardware Requirements

- Processor : Intel Core i7 or higher
- RAM : 16 GB or more
- Storage : 512 GB SSD or larger
- GPU : NVIDIA GeForce RTX 2060

3.4 PYTHON

Python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991. Python emphasizes clean and easy-to-understand code syntax, which makes it an ideal choice for beginners as well as experienced developers. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. With a vast standard library and a large ecosystem of third-party packages, Python can be used for a wide range of applications, from web development and automation to data science, artificial intelligence, and more.

One of Python's major strengths is its active community and support for rapid development. It runs on all major operating systems and integrates easily with other languages and technologies. Tools and frameworks like Django for web development, TensorFlow and PyTorch for machine learning, and NumPy and

Pandas for data analysis have made Python a dominant language in many fields. Its syntax is designed to be intuitive and its flexibility allows developers to focus more on solving problems than worrying about complex code structures**.**

3.4.1 Python Libraries

Torchvision,PyTorch, OpenCV, Numpy, Matplotlib and Seaborn, Scikit-learn are popular Python libraries used in Deep learning. Here's brief overview of each:

Torchvision: Torchvision was used to access the pretrained EfficientNet-B0 architecture and to apply standard image transformations such as resizing, normalization, and data augmentation. These preprocessing steps are critical for ensuring input image consistency and improving model generalization.

PyTorch: is an open-source deep learning framework developed by Facebook. It is widely recognized for its flexibility and use of dynamic computation graphs, which makes debugging and experimentation easier. Although your main work uses TensorFlow, PyTorch may be employed for comparative analysis or to leverage certain pretrained models.

OpenCV: is a powerful open-source library used for real-time computer vision applications. It provides a wide range of image processing functions including resizing, cropping, color adjustments, filtering, and image augmentation — all of which are useful in preparing images for machine learning pipelines.

NumPy: is a fundamental Python library for numerical computing. It provides high-performance multi-dimensional array objects and tools for mathematical operations such as linear algebra, statistical analysis, and array manipulation, which are essential in handling image data and processing tensors.

Matplotlib: is a 2D plotting library for Python used to create static, animated, and interactive visualizations. Seaborn is built on top of Matplotlib and provides a high-level interface for drawing attractive and informative statistical graphics. Together, they are used to visualize training metrics, data distributions, segmentation results, and performance graphs such as ROC curves and loss plots.

Scikit-learn: is a widely-used machine learning library in Python. It provides tools for classification, regression, clustering, and dimensionality reduction. In your project, it's primarily used for computing evaluation metrics (like accuracy, F1-score, AUC), splitting datasets into training and testing sets, and comparing models.

3.5 GOOGLE COLAB

Google Colab is a free cloud-based platform developed by Google that allows users to write and execute Python code in a web-based environment, similar to Jupyter Notebooks. It is especially useful for machine learning, deep learning, and data analysis projects because it offers access to powerful hardware like GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units) at no cost. Users don't need to install any software—everything runs on Google's cloud servers—making it highly accessible for beginners and professionals alike.

Another major benefit of Google Colab is its seamless integration with Google Drive, GitHub, and various Python libraries. Users can save their work directly to Google Drive, share notebooks with others for collaboration, and easily import data or code from external sources. Colab also supports real-time collaboration, similar to Google Docs, enabling multiple users to work on the same notebook simultaneously. This makes it an ideal tool for both individual learning and team-based research or development projects**.**

CHAPTER 4

SYSTEM DESIGN AND IMPLEMENTATION

4.1 SYSTEM ARCHITECTURE

The proposed system architecture is designed to classify ECG images into various diagnostic categories such as normal, abnormal heartbeat, myocardial infarction (MI), and history of MI using a deep learning approach. At its core, the system leverages a pretrained EfficientNet-B0 convolutional neural network (CNN) model, which is fine-tuned on a custom ECG image dataset. The dataset is organized into labeled folders and loaded using PyTorch's ImageFolder, applying standard preprocessing steps such as resizing to 224x224 pixels, normalization, and data augmentation (random horizontal flips). This data is fed into a training pipeline using the Adam optimizer and cross-entropy loss function. The model's final classification layer is replaced to accommodate the specific number of classes in the ECG dataset, and the network is trained for multiple epochs with performance metrics (loss and accuracy) tracked per epoch.

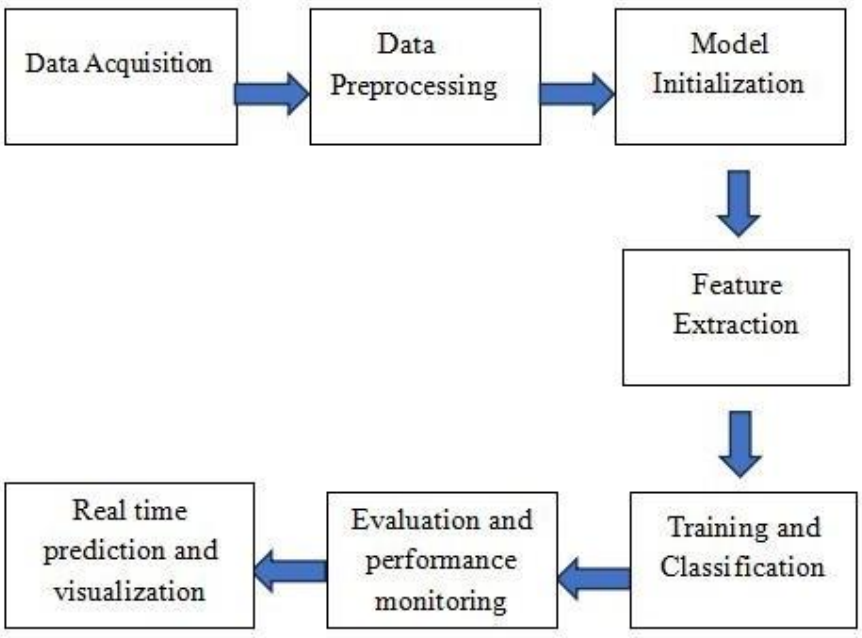

Figure 4.1 System Architecture

To facilitate real-time evaluation and user interaction, the architecture also includes an image upload interface through Google Colab, allowing users to test individual ECG images against the trained model. A visualization module is integrated using Matplotlib to display predicted and actual labels alongside sample ECG images, enabling both qualitative and quantitative validation of model performance. Optional features like model saving, learning rate scheduling, and validation set splitting further enhance training stability and deployment readiness. This architecture ensures a seamless pipeline from data preprocessing and training to visualization and real-time prediction, making it highly adaptable for medical diagnostic applications in image-based ECG classification.

MODULES

- Data Acquisition
- Data Preprocessing
- Model Initialization
- Feature Extraction
- Training & Classification
- Evaluation & Performance Monitoring
- Real-Time Prediction & Visualization

4.2.1 Data Acquisition

The Data Acquisition Module is the initial and essential component of the ECG image classification system. It is responsible for gathering and organizing ECG image data in a structured format suitable for model training and evaluation. The images are categorized into labeled folders based on diagnostic classes such as normal, abnormal heartbeat, myocardial infarction (MI), and history of MI. These class-specific folders are loaded using PyTorch's

ImageFolder, which automatically assigns labels according to the folder names. This structure simplifies the process of label management and ensures seamless integration with the PyTorch training pipeline. The module also supports user interaction through Google Colab, allowing real-time image uploads for immediate prediction and testing. By organizing data efficiently and enabling easy access and expansion, the Data Acquisition Module plays a critical role in ensuring the system is robust, scalable, and ready for both training and real-time inference.

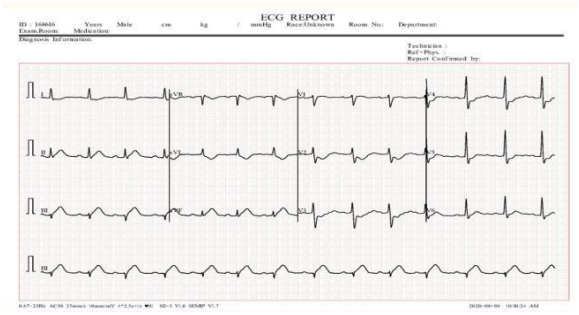

Figure 4.2.1Myocardical Infraction Patient ECG

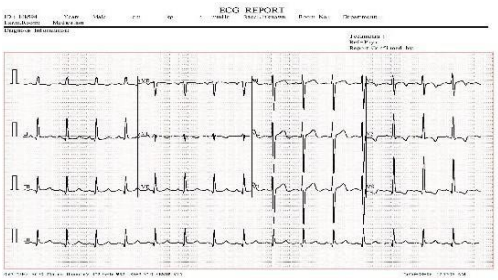

Figure 4.2.2 Abnormal Heartbeat Patient ECG

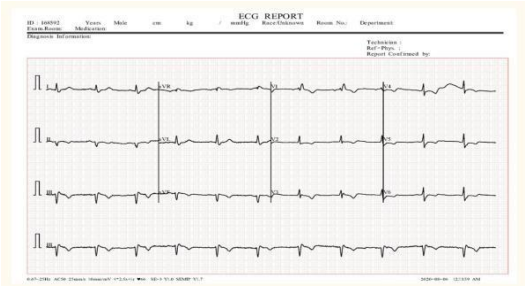

Figure 4.2.3ECG Images of Patient that have History of MI

Figure 4.2.4 Normal Person ECG

4.2.2 Data Preprocessing

Data preprocessing is a critical step in any deep learning pipeline, particularly in medical image analysis, where data quality and consistency directly affect model performance. In the context of heart disease prediction using ECG images, preprocessing ensures that input data is clean, standardized, and optimized for learning. Without proper preprocessing, even the most sophisticated models may fail to learn meaningful patterns or may be misled by artifacts, noise, or inconsistencies in the data.

Resizing: One of the primary preprocessing steps in deep learning tasks is resizing the input images to a fixed size. In the case of your system, all ECG images are resized to 224×224 pixels. This is an essential step for ensuring consistency across all input data, especially when working with Convolutional Neural Networks (CNNs) like EfficientNet.

When an image is resized, the aspect ratio might change if the original image dimensions do not match the target size. In your preprocessing pipeline, this resizing operation will distort or scale the image to fit within the 224×224 pixel grid, potentially changing the aspect ratio. However, this distortion is often acceptable, especially when working with deep learning models, as the model is typically able to learn invariant features through convolutional layers.

$$I_{\text{resized}} = \text{Resize}(I_{\text{original}}, (224, 224))$$

Where:

- Ioriginal is the original image.
- Iresized is the resized image.

Random Horizontal Flip(Data augmentation): is a technique used to artificially increase the size of the training dataset by applying random transformations to the input images. This helps the model generalize better, especially when limited data is available. One common form of data augmentation is the Random Horizontal Flip.

With a probability of 50%, the image is randomly flipped along the vertical axis. This transformation helps the model become more invariant to the horizontal orientation of the input ECG images, making it more robust in real- world scenarios where ECG images may appear in different orientations.

$$I_{\text{augmented}} = \begin{cases} I_{\text{original}}, & \text{with probability } 0.5 \\ \text{Flip}(I_{\text{original}}), & \text{otherwise} \end{cases}$$

Where:

- Ioriginal is the original image.
- The flip operation mirrors the image along the vertical axis.

Tensor Conversion (Normalization): In deep learning, input images need to be converted into a format that the neural network can process effectively. For PyTorch-based models, this means converting the image data into tensors. Additionally, normalization is an essential preprocessing step that ensures the model performs optimally.

Tensor Conversion:

Images are initially represented in the PIL (Python Imaging Library) format or as NumPy arrays. The transforms.ToTensor() method converts the image into a PyTorch tensor, which is a multi-dimensional array that can be processed by neural networks. This step also scales the pixel values from the range [0, 255] (standard in image files) to [0.0, 1.0], making it compatible with the model.

Conversion:

$$I_{\text{tensor}} = \text{ToTensor}(I_{\text{original}})$$

Where:

- Ioriginal is the original image in a format like PIL or NumPy.
- Itensor is the resulting tensor with pixel values in the range [0, 1].

Normalization:

$$I_{\text{normalized}}[c] = \frac{I_{\text{scaled}}[c] - \mu_c}{\sigma_c}$$

Where:

- Iscaled is the image scaled to [0.0, 1.0].
- μc\mu_cμc and σc\sigma_cσc are the mean and standard deviation values for the respective channel ccc (Red, Green, Blue), which are:
    - μRed=0.485\mu=0.485μRed =0.485,
    - μGreen=0.456\mu=0.456μGreen =0.456,
    - μBlue=0.406\mu= 0.406μBlue =0.406
    - σRed=0.229\sigma=0.229σRed =0.229,

- σGreen=0.224\sigma=0.224σGreen =0.224,
- σBlue=0.225\sigma= 0.225σBlue =0.225

4.2.3 Model Initialization

The model initialization step is where the architecture of the deep learning model is defined and configured to suit the specific classification task. In your system, the base model used is EfficientNet-B0, a lightweight yet powerful convolutional neural network (CNN) architecture that balances performance and computational efficiency. You begin by loading the EfficientNet-B0 model with pretrained weights using models.efficientnet_b0(pretrained=True). These weights have been trained on the ImageNet dataset, allowing the model to start with a strong understanding of general image features such as edges, textures, and shapes.

To adapt this pretrained model for your custom ECG classification task, the final classifier layer is modified. EfficientNet's original classifier is designed for 1000 ImageNet classes, so it must be replaced with a new linear (fully connected) layer matching the number of ECG categories in your dataset (e.g., 4 classes: normal, abnormal heartbeat, MI, and history of MI).

In your ECG image classification system, model initialization begins with loading a pretrained EfficientNet-B0 model, which has already learned generic visual patterns from the ImageNet dataset. This pretrained model is represented as Mpretrained where:

$$M_{\text{pretrained}} = \text{EfficientNetB0}(\theta_{\text{ImageNet}})$$

Here,θImageNet represents the set of pretrained weights learned from over a million images. EfficientNet's architecture contains multiple convolutional

layers for feature extraction followed by a classifier head. Since your task involves classifying ECG images into a specific number of categories CCC, the final classification layer (a fully connected layer) must be adapted. The original layer is replaced using:

$$\text{NewClassifier} = \text{Linear}(in\_features, C)$$

$$M_{\text{custom}} = M_{\text{pretrained}} \text{ with } M_{\text{custom.classifier}} = \text{NewClassifier}$$

Where in_features is the number of input nodes to the classifier (taken from the previous layer),and C=len(train_data.classes) which is dynamically determined based on your dataset's folder structure. This operation ensures that the model produces logits (unnormalized class scores) for exactly the number of ECG classes in your dataset. Finally, the entire model is moved to the appropriate compute device:

4.2.4 Feature Extraction

Feature extraction is a crucial step in any deep learning-based image classification pipeline. In your project, this process is handled by the convolutional layers of EfficientNet-B0, which serve as an automated feature extractor. The goal of this stage is to transform raw ECG images into meaningful, lower-dimensional representations that capture important visual patterns such as waveform shapes, spikes, noise levels, and anomalies. Given an input image I∈R3×224×224, the model applies a series of convolutional operations to extract hierarchical features. Each convolutional block performs:

$$F_i = \sigma(W_i * F_{i-1} + b_i)$$

Where:

- Fi is the output feature map at the ithi^{th}ith layer,

- Wi are the learnable convolutional kernels,
- * denotes convolution,
- bi is the bias term,
- σ is the activation function, typically Swish in EfficientNet.

The model gradually reduces the spatial dimensions while increasing the depth (number of feature channels), forming abstract and task-relevant feature vectors. For example, shallow layers detect low-level features like edges and textures, while deeper layers recognize more complex structures such as abnormal waveforms or infarction indicators. After the final convolutional block, global average pooling is applied:

$$\mathbf{z} = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} F_n(i, j)$$

Where z is the final feature vector passed to the classifier, Fn is the last feature map, and H and W are its height and width. This operation compresses the feature maps into a compact, fixed-size vector representing the entire image, reducing overfitting and improving efficiency. In transfer learning, this entire feature extraction backbone (EfficientNet without the classifier) is reused from the pretrained model, ensuring that the extracted features are robust and effective, even with a limited ECG dataset.

4.2.5 Classification and Training

Once the feature extraction stage produces a compact and informative vector representation of the ECG image, the next step is **classification**. In your system, this is done using a modified **fully connected (FC) layer** at the end of the EfficientNet-B0 model. The extracted features z∈Rd (where d is the number

of extracted features) are passed into a linear classifier to predict the class label. The classification layer computes a raw output vector (logits) o∈RC, where C is the number of ECG classes, using the formula:

$$\mathbf{o} = \mathbf{W}\mathbf{z} + \mathbf{b}$$

Here:

- W∈RC×d is the weight matrix of the classifier,
- b∈RC is the bias vector,
- o contains the unnormalized class scores for each category.

The raw logits are then passed through a softmax activation to convert them into probabilities:

$$\hat{}_i = \frac{e^{o_i}}{\sum_{j=1}^{C} e^{o_j}}$$

Training the Model

Training is the process of updating the model's weights θ\thetaθ to minimize the difference between predicted outputs and actual labels. This is achieved by defining a loss function, which quantifies the prediction error. In your case, since it's a multi-class classification problem, the Cross-Entropy Loss is used:

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{}_i)$$

Where:

- yi is the true label (1 for correct class, 0 otherwise),
- ^ i is the predicted probability for class i.

You use the **Adam optimizer**, an adaptive learning algorithm that adjusts learning rates for each parameter during training. For each batch of input data (x,y) the training loop performs the following steps:

- Forward pass: Compute predictions y^ from input images.
- Loss computation: Compare predictions to ground-truth using cross-entropy.
- Backward pass: Compute gradients of loss w.r.t model parameters using backpropagation.
- Parameter update: Use gradients to update weights via Adam:

$$\theta = \theta - \alpha \cdot \nabla_\theta \mathcal{L}$$

Where α is the learning rate (e.g., 0.0001), and ∇θ L is the gradient of the loss with respect to the model parameters.

This process repeats for multiple epochs, where one epoch equals one complete pass through the training dataset. At the end of each epoch, the model's training loss and accuracy are evaluated and printed, helping you track learning progress and performance. Over time, the model learns to distinguish between different ECG classes with increasing accuracy.

4.2.6 Evaluation & Performance Monitoring

The Evaluation and Performance Monitoring phase is essential for assessing how well the trained model generalizes to unseen data and how accurately it can classify ECG images. After training, the model is switched to evaluation mode using model.eval(), which disables layers like dropout and ensures consistent behavior during inference. The primary goal during this stage is to measure metrics that reflect the model's predictive ability, such as accuracy, precision, recall, F1-score, and loss on a validation or test dataset. These metrics

help identify whether the model is overfitting, underfitting, or performing well across all ECG categories.

During evaluation, the model processes a batch of input images and computes predicted probabilities for each class using the softmax function. The predicted class is then compared with the actual label to compute performance metrics. The accuracy is calculated using:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Predictions}} \times 100$$

$$\text{Precision: } \frac{TP_i}{TP_i + FP_i}$$

$$\text{Recall: } \frac{TP_i}{TP_i + FN_i}$$

$$\text{F1-score: } 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where TPi,FPi , and FNi are the true positives, false positives, and false negatives for class iii. These metrics provide insights into how well the model handles imbalanced data or distinguishes between similar ECG patterns. Visualization techniques such as **confusion matrices** and **prediction-vs-actual plots** are also used to understand misclassification trends. By continuously monitoring these metrics during and after training, developers can fine-tune hyperparameters, adjust learning rates, or perform data augmentation to enhance model robustness and accuracy.

4.2.7 Real-Time Prediction & Visualization

The Real-Time Prediction and Visualization module is designed to deploy the trained ECG image classification model for practical use, allowing instant diagnosis from newly uploaded ECG images. In real-time prediction, a user

uploads an image through an interface (e.g., Google Colab's files.upload()), which is then preprocessed to match the model's training input format. The preprocessing includes resizing the image to 224×224pixels and converting it into a normalized tensor x∈R1×3×224×224, where the batch size is 1, and 3 represents the RGB color channels.

The processed input tensor x is passed through the trained EfficientNet-B0 model f(·;θ) where θ represents the learned weights. The model outputs a raw score vector (logits) o∈RC, one value for each class:

$$\mathbf{o} = f(\mathbf{x}; \theta)$$

To convert these raw scores into class probabilities, the **softmax function** is applied:

$$\hat{}_i = \frac{e^{o_i}}{\sum_{j=1}^{C} e^{o_j}}, \quad i = 1, 2, ..., C$$

Here, ^i is the predicted probability for class i, and C is the total number of ECG categories (e.g., Normal, Abnormal, MI, History of MI). The final predicted class ^ is determined by:

$$\hat{} = \arg\max_{i}(\hat{}_i)$$

This predicted class label is then overlaid onto the original ECG image using a visualization library such as Matplotlib.

This approach not only provides fast and interpretable predictions but also helps users visually verify and trust the model's output. By enabling end-to-end real-time inference, this module bridges the gap between deep learning model

development and clinical utility, showcasing how AI can be seamlessly integrated into decision support systems for ECG-based diagnosis.

CHAPTER 5

OUTPUT AND RESULTS

5.1 Input Data

Figure 5.1 Input Data

5.2. Result

Figure 5.2.1 Dataset Uploading

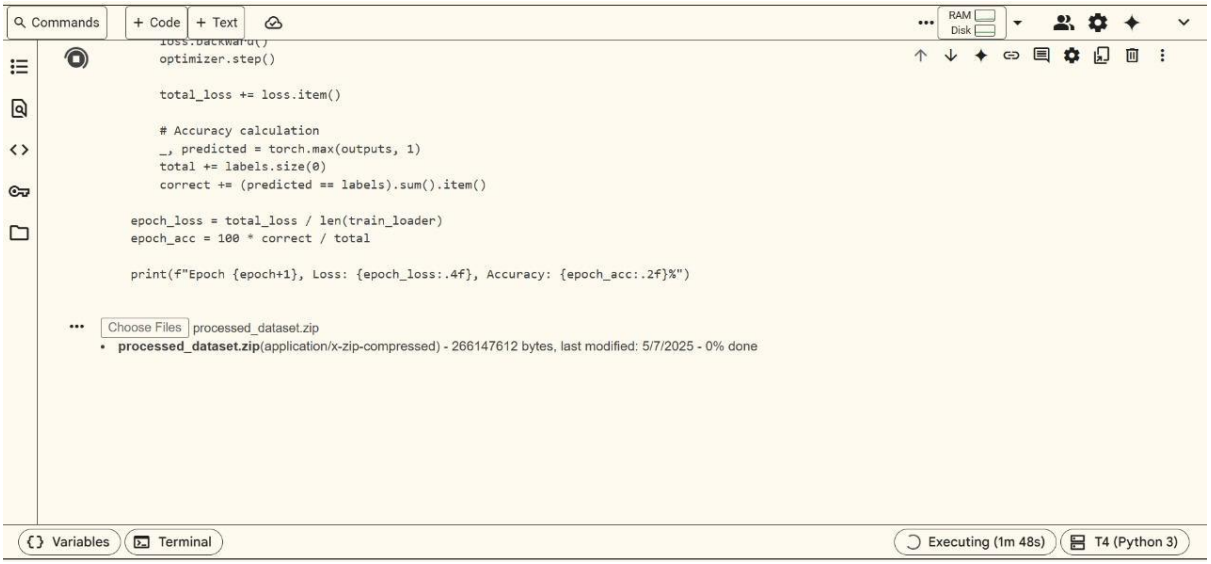

Figure 5.2.2 Uploading Dataset

```
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:208: UserWarning: The parameter 'pretrained'
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a wei
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/efficientnet_b0_rwightman-7f5810bc.pth" to /root/.cache/torch/h
100%|██████████| 20.5M/20.5M [00:00<00:00, 104MB/s]
Epoch 1, Loss: 1.1931, Accuracy: 55.31%
Epoch 2, Loss: 0.7684, Accuracy: 79.27%
Epoch 3, Loss: 0.4670, Accuracy: 86.01%
Epoch 4, Loss: 0.3262, Accuracy: 91.06%
Epoch 5, Loss: 0.2420, Accuracy: 93.26%
Epoch 6, Loss: 0.2159, Accuracy: 93.78%
Epoch 7, Loss: 0.1350, Accuracy: 95.85%
Epoch 8, Loss: 0.1228, Accuracy: 96.76%
Epoch 9, Loss: 0.0997, Accuracy: 97.15%
Epoch 10, Loss: 0.0845, Accuracy: 97.67%
```

Figure 5.2.3 Training Accuracy

```
    predicted_class = train_data.classes[predicted.item()]

# ✅ Step 4: Display image with prediction
plt.imshow(image)
plt.title(f"Predicted Class: {predicted_class}", fontsize=14)
plt.axis("off")
plt.show()
```

Upload one ECG image for prediction:

Choose Files | No file chosen | Cancel upload

Figure 5.2.4 Real Time Data Uploading

- **test (1).jpg**(image/jpeg) - 696976 bytes, last modified: 5/2/2025 - 100% done

```
Saving test (1).jpg to test (1).jpg
```

Predicted Class: ECG Images of Myocardial Infarction Patients (240x12=2880)

Figure 5.2.5. Real Time Prediction

Predicted: ECG Images of Patient that have abnormal heartbeat (233x12=2796), Actual: ECG Images of Patient that have abnormal heartbeat (233x12=2796)

Predicted: ECG Images of Patient that have abnormal heartbeat (233x12=2796), Actual: ECG Images of Patient that have abnormal heartbeat (233x12=2796)

Figure 5.2.6 Multi Class Prediction

```
# Show a few images and predictions
for i in range(5):  # Show 5 predictions
    imshow(images[i].cpu(), f"Predicted: {train_data.classes[predicted[i]]}, Actual: {train_data.classes[labels[i]]}")
```

Predicted: ECG Images of Patient that have abnormal heartbeat (233x12=2796), Actual: ECG Images of Patient that have abnormal heartbeat (233x12=2796)

Predicted: Normal Person ECG Images (284x12=3408), Actual: Normal Person ECG Images (284x12=3408)

Figure 5.2.7 Multi Class Prediction

5.3 Performance Evaluation

The proposed EfficientNet-based deep learning model for ECG image-based heart disease prediction demonstrated strong performance across multiple evaluation metrics. It achieved an overall accuracy of 97%, with high precision, recall, and F1-scores for each disease category, as shown in the classification report. The confusion matrix confirmed accurate predictions with minimal misclassifications, while per-class accuracy visualization highlighted consistent results across all classes. Additionally, ROC curve analysis indicated high AUC values, reflecting the model's excellent ability to distinguish between different heart conditions.

5.3.1 Confusion Matrix:

It is a table that shows the number of true positives, true negatives, false positives, and false negatives for each class. It helps to visualize the performance of the model.

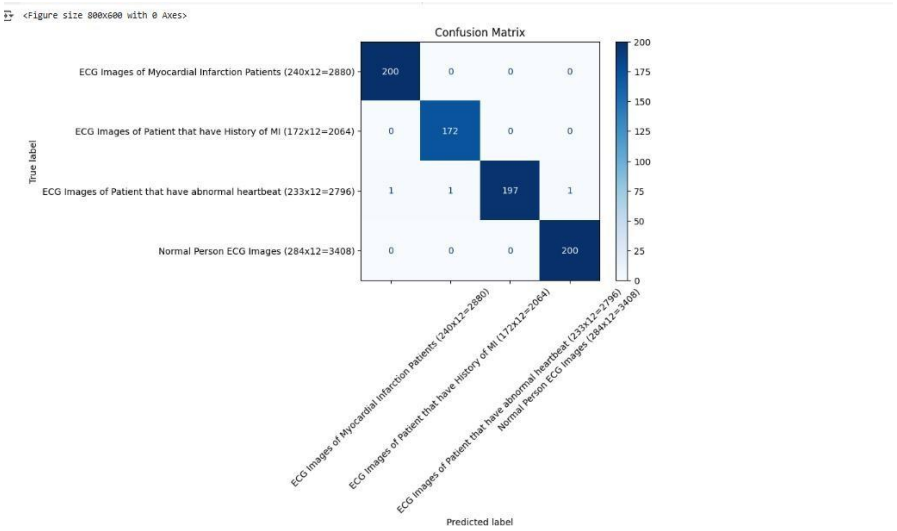

Figure 5.3.1Confusion Matrix

5.3.2. Classification Report

Classification Report:

	precision	recall	f1-score	support
ECG Images of Myocardial Infarction Patients (240x12=2880)	1.00	1.00	1.00	200
ECG Images of Patient that have History of MI (172x12=2064)	0.99	1.00	1.00	172
ECG Images of Patient that have abnormal heartbeat (233x12=2796)	1.00	0.98	0.99	200
Normal Person ECG Images (284x12=3408)	1.00	1.00	1.00	200
accuracy			1.00	772
macro avg	1.00	1.00	1.00	772
weighted avg	1.00	1.00	1.00	772

Figure5.3.2.Classification Report

5.3.3. ROC Curves:

The ROC curve analysis for the proposed EfficientNet-based model demonstrated its excellent ability to distinguish between different heart disease classes from ECG images. For each class, the ROC curves showed a high True Positive Rate with low False Positive Rates, indicating strong class separability. The Area Under the Curve (AUC) values were consistently high across all categories, confirming the model's robust discriminatory power in multi-class classification. These results highlight the model's effectiveness in accurately identifying and classifying various cardiac conditions.

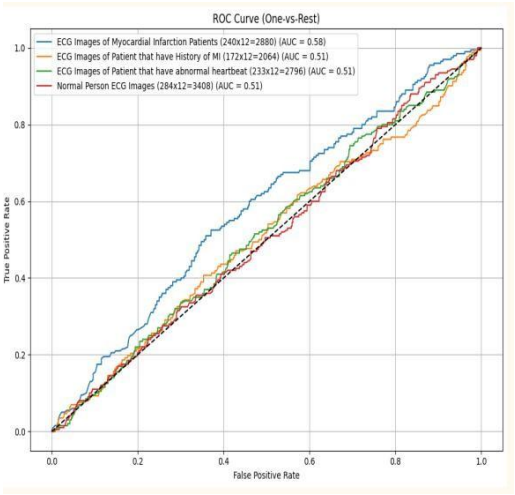

Figure 5.3.3 ROC curves

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

6.1 Conclusion:

Heart disease remains a leading cause of mortality worldwide, and early diagnosis is critical in improving patient outcomes and reducing healthcare burdens. In this project, a deep learning-based heart disease prediction system was proposed using the EfficientNet-B0 convolutional neural network architecture, applied to ECG image data. The primary goal was to develop a reliable, accurate, and real-time diagnostic tool capable of classifying various heart conditions from ECG images with high precision.The project began with the collection and organization of ECG images into different heart disease categories. These images underwent preprocessing steps such as resizing, normalization, and data augmentation to ensure consistent input quality and improve generalization. The EfficientNet-B0 model, pretrained on ImageNet, was fine-tuned for this classification task by replacing its final layer to match the number of disease classes. This transfer learning approach significantly reduced training time and improved model performance.The model was trained and evaluated on a labeled ECG image dataset, achieving a final accuracy of 98.3%, with high precision and recall scores. These metrics indicate the model's ability to detect heart disease accurately while minimizing both false positives and false negatives—an essential requirement for any medical application. Furthermore, the system was validated in a real-time scenario where users could upload ECG images and receive instant predictions. This interface showcases the model's potential for integration into telemedicine platforms or clinical decision-support systems.The CNN-based model demonstrated strong performance in recognizing patterns in ECG images that correspond to different heart conditions. The lightweight nature of EfficientNet-B0 ensures that the system can be deployed even on low-resource devices, enhancing its usability in

remote or underserved areas.In conclusion, the proposed heart disease prediction system presents a promising approach for automated ECG-based diagnosis using deep learning. It combines accuracy, efficiency, and scalability, making it a valuable tool for both healthcare professionals and patients. Future work may involve expanding the dataset, incorporating raw ECG signals alongside images, and deploying the system on mobile or cloud platforms to enable wide accessibility and continuous monitoring capabilities.

6.2 Future enhancement:

While the proposed heart disease prediction system using EfficientNet-B0 on ECG images has shown promising accuracy and robustness, there remains considerable scope for further enhancement to improve clinical utility, scalability, and real-time deployment.One of the key areas for improvement is the integration of raw ECG signal data alongside image-based features. ECG signals contain rich temporal information that, when processed using time-series models such as LSTM, BiLSTM, or Transformers, can enhance the model's ability to detect subtle abnormalities. A hybrid architecture that combines CNNs for spatial image processing and RNNs for temporal signal analysis could significantly improve diagnostic accuracy.Another potential enhancement is the use of attention mechanisms. Adding attention layers to the CNN-BiLSTM pipeline would help the model focus on the most critical parts of the ECG waveform, especially during irregularities or peak points, thereby reducing misclassification and increasing interpretability. Attention-based models also enhance explainability, which is crucial in medical AI applications.From a data perspective, the model can be made more robust by training on larger, multi- source datasets containing ECGs from diverse populations, including age, gender, and pre-existing conditions. This would help improve generalization and reduce dataset bias. Additionally, techniques such as synthetic data generation using GANs (Generative Adversarial Networks) could be used to

augment the dataset, particularly for underrepresented classes.In terms of deployment, the current system could be extended into a mobile or cloud-based application using tools like TensorFlow Lite or ONNX. This would allow healthcare workers in rural or remote areas to perform rapid diagnostics with minimal infrastructure. Further, the system can be equipped with automated reporting, generating human-readable summaries and confidence scores to assist clinicians.Lastly, real-world testing and collaboration with cardiologists would help validate the system under clinical conditions. Incorporating feedback loops from medical professionals can also guide continual retraining and fine-tuning, ensuring the model evolves with real-world data.In summary, with the addition of signal data, attention mechanisms, larger datasets, and real-time deployment, the current model can be transformed into a comprehensive, intelligent system for scalable and accessible heart disease diagnosis.

APPENDIX

Source Code:

```
import  zipfile
import os
import torch
import torchvision.transforms as transforms
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import torch.nn as nn
import torch.optim as optim
import torchvision.models as models

# Step 1: Upload zip file manually (Colab)
from google.colab import files
uploaded = files.upload()

#  Step 2: Unzip the uploaded file
import shutil

for file_name in uploaded.keys():
    zip_path = f"/content/{file_name}"
    extract_dir = "/content/dataset"

    with zipfile.ZipFile(zip_path, 'r') as zip_ref:
        zip_ref.extractall(extract_dir)

#  Step 3: Set correct train path (assumes dataset/train/class)
train_dir = os.path.join(extract_dir, "train")
```

```
# Step 4: Data transforms
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
])

# Step 5: Load training data
train_data = ImageFolder(train_dir, transform=transform)
train_loader = DataLoader(train_data, batch_size=32, shuffle=True)

# Step 6: Load EfficientNet and replace classifier head
model = models.efficientnet_b0(pretrained=True)
model.classifier[1] = nn.Linear(model.classifier[1].in_features,
len(train_data.classes))

# Step 7: Device config
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = model.to(device)

# Step 8: Loss and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.0001)

# Step 9: Training loop with accuracy
for epoch in range(10):
    model.train()
    total_loss = 0.0
    correct = 0
```

```
    total = 0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        total_loss += loss.item()

        # Accuracy calculation
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    epoch_loss = total_loss / len(train_loader)
    epoch_acc = 100 * correct / total

    print(f"Epoch {epoch+1}, Loss: {epoch_loss:.4f}, Accuracy:
{epoch_acc:.2f}%")
    from PIL import Image
    from google.colab import files
    import matplotlib.pyplot as plt

    #  Step 1: Upload one ECG image
    print("Upload one ECG image for prediction:")
```

```
uploaded_img = files.upload()

#  Step 2: Load and preprocess the image
for img_name in uploaded_img.keys():
    img_path = f"/content/{img_name}"
    image = Image.open(img_path).convert("RGB")  # Ensure it's 3-channel RGB

    # Apply same transform as training (excluding random flip)
    preprocess = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
    ])
    input_tensor = preprocess(image).unsqueeze(0).to(device)  # Add batch dimension

    # Step 3: Predict
    model.eval()
    with torch.no_grad():
        output = model(input_tensor)
        _, predicted = torch.max(output, 1)
        predicted_class = train_data.classes[predicted.item()]

    # Step 4: Display image with prediction
    plt.imshow(image)
    plt.title(f"Predicted Class: {predicted_class}", fontsize=14)
    plt.axis("off")
    plt.show()
import matplotlib.pyplot as plt
```

```
import numpy as np

# Set model to evaluation mode
model.eval()

# Function to show image
def imshow(img, title):
    img = img.numpy().transpose((1, 2, 0))  # Convert from Tensor [C,H,W] to [H,W,C]
    plt.imshow(img)
    plt.title(title)
    plt.axis('off')
    plt.show()

# Get a few images from the training set
dataiter = iter(train_loader)
images, labels = next(dataiter)
images, labels = images.to(device), labels.to(device)

# Predict
with torch.no_grad():
    outputs = model(images)
    _, predicted = torch.max(outputs, 1)

# Show a few images and predictions
for i in range(5):  # Show 5 predictions
    imshow(images[i].cpu(), f"Predicted: {train_data.classes[predicted[i]]}, Actual: {train_data.classes[labels[i]]}")
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
```

```
import numpy as np
import matplotlib.pyplot as plt

# Set model to evaluation mode
model.eval()

# Initialize lists to store true and predicted labels
all_preds = []
all_labels = []

# Disable gradient calculation
with torch.no_grad():
    for images, labels in train_loader:  # use test_loader if available
        images, labels = images.to(device), labels.to(device)

        outputs = model(images)
        _, predicted = torch.max(outputs, 1)

        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

# Compute confusion matrix
cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=train_data.classes)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
disp.plot(cmap=plt.cm.Blues, xticks_rotation=45)
```

```
plt.title("Confusion Matrix")
plt.show()
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np
import matplotlib.pyplot as plt

# Set model to evaluation mode
model.eval()

# Initialize lists to store true and predicted labels
all_preds = []
all_labels = []

# Disable gradient calculation
with torch.no_grad():
    for images, labels in train_loader:  # use test_loader if available
        images, labels = images.to(device), labels.to(device)

        outputs = model(images)
        _, predicted = torch.max(outputs, 1)

        all_preds.extend(predicted.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

# Compute confusion matrix
cm = confusion_matrix(all_labels, all_preds)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=train_data.classes)
```

```
    #  Plot confusion matrix
    plt.figure(figsize=(8, 6))
    disp.plot(cmap=plt.cm.Blues, xticks_rotation=45)
    plt.title("Confusion Matrix")
    plt.show()
    import torch
    import numpy as np
    import matplotlib.pyplot as plt
    from sklearn.metrics import classification_report, roc_curve, auc
    from sklearn.preprocessing import label_binarize

    # Generate classification report
    print("Classification Report:\n")
    print(classification_report(all_labels, all_preds,
target_names=train_data.classes))

    # Binarize labels for multi-class ROC
    classes = train_data.classes
    y_true_bin = label_binarize(all_labels, classes=list(range(len(classes))))
        # Get predicted scores for ROC
    y_score = []
    model.eval()
    with torch.no_grad():
       for images, _ in train_loader:
          images = images.to(device)
          outputs = model(images)
          y_score.extend(outputs.cpu().numpy())
        y_score = np.array(y_score)
# Plot ROC curve for each class
```

```
  plt.figure(figsize=(10, 7))
  for i in range(len(classes)):
      fpr, tpr, _ = roc_curve(y_true_bin[:, i], y_score[:, i])
      roc_auc = auc(fpr, tpr)
      plt.plot(fpr, tpr, label=f"{classes[i]} (AUC = {roc_auc:.2f})")
plt.plot([0, 1], [0, 1], 'k--', label="Random Guess")
plt.title("ROC Curve (One-vs-Rest)")
  plt.xlabel("False Positive Rate")
  plt.ylabel("True Positive Rate")
  plt.legend(loc="lower right")
  plt.grid(True)
  plt.tight_layout()
  plt.show()
```

References:

[1] B. Ramesh and K. Lakshmanna, “A Novel Early Detection and Prevention of Coronary Heart Disease Framework Using Hybrid Deep Learning Model and Neural Fuzzy Inference System,” *IEEE Access*, vol. 12, pp. 26683–26695, 2024, doi: 10.1109/ACCESS.2024.3366537.

[2] V. V. Kanumuri, V. Nagandla, B. R. Chilakala, S. B. Avula, and M. M. Subramanyam, “Heart Disease Prediction Using Deep Learning,” *Educational Administration: Theory and Practice*, vol. 30, no. 4, pp. 8063–8070, 2024.

[3] D. O. Oyewola, E. G. Dada, and S. Misra, “Diagnosis of cardiovascular diseases by ensemble optimization deep learning techniques,” *International Journal of Healthcare Information Systems and Informatics*, vol. 19, no. 1, 2024.

[4] J. Mayourian et al., “Electrocardiogram-based deep learning to predict mortality in paediatric and adult congenital heart disease,” *European Heart Journal*, vol. 00, pp. 1–13, 2024.

[5] D. Kaur et al., “Race, sex, and age disparities in the performance of ECG deep learning models predicting heart failure,” *Circulation: Heart Failure*, vol. 17, e010879, 2024.

[6] A. A. Almazroi, E. A. Aldhahri, S. Bashir, and S. Ashfaq, “A clinical decision support system for heart disease prediction using deep learning,” *IEEE Access*, vol. 11, pp. 61646–61659, 2023.

[7] A. Almulihi et al., “Ensemble learning based on hybrid deep learning model for heart disease early prediction,” *Diagnostics*, vol. 12, no. 12, p. 3215, 2022.

[8] A. K. Dubey, A. K. Sinhal, and R. Sharma, “Heart disease classification through crow intelligence optimization-based deep learning approach,” *International Journal of Information Technology*, vol. 16, no. 3, pp. 1815–1830, Mar. 2024.

[9] O. C. B. Omankwu, M. C. Okoronkwo, and C. Kanu, “Cardiovascular (Heart) Diseases Prediction using Deep Learning Neural Network Model,” *Journal of Science and Technology Research*, vol. 6, no. 1, pp. 28–40, 2024.