DEPARTMENT OF COMPUTER TECHNOLOGY ANNA UNIVERSITY, MIT CAMPUS

Creative and Innovative Project (CS6811), Jan 2025 – May 2025

Review III Report

Team ID:B23

Optimized CNN-Based Hybrid Model for Multi-Label Classification of Cardiovascular Diseases in 12-Lead ECG Images

Submitted by

YUVAPRIYA S 2022503552

POOJA S 2022503024

ATHITHRAJA R 2022503702

Under the guidance of

Dr. T. Sudhakar

Assistant Professor
Department of Computer Technology

Anna University, MIT Campus

TITLE

Optimized CNN-Based Hybrid Model for Multi-Label Classification of Cardiovascular Diseases in 12-Lead ECG Images

DOMAIN

Machine Learning, Deep Learning, Cardiovascular Disease Detection, Biomedical Signal Processing

INTRODUCTION

Cardiovascular diseases (CVDs) are a leading cause of mortality worldwide, responsible for an estimated 17.9 million deaths each year . Early and accurate diagnosis of CVDs is crucial for timely intervention and improved patient outcomes. The electrocardiogram (ECG) is a non-invasive and cost-effective tool widely used for the initial screening and diagnosis of various heart conditions.

With the advancements in deep learning, particularly Convolutional Neural Networks (CNNs), there is a growing interest in developing automated systems for ECG analysis. CNNs have shown remarkable success in various classification tasks, including medical image analysis . However, their application to ECG data, which is a one-dimensional time- series signal, presents unique challenges.

This paper proposes a supervised learning approach using a CNN to classify heart conditions into four categories: heart attack, history of heart attack, normal, and abnormal. The proposed model aims to provide a reliable and efficient tool for assisting clinicians in the interpretation of ECG data.

PROBLEM STATEMENT

The ECG is the most frequently used and reliable tool for the initial diagnosis and risk assessment of CVDs. However, the traditional ECG interpretation is time-consuming and has high inter-observer variability.

In recent years, deep learning has shown great promise in improving the accuracy and efficiency of ECG analysis. However, most of the current deep learning-based ECG analysis

models are designed for single CVD detection, which cannot meet the needs of clinical practice.

In this paper, we propose a novel deep learning framework for multi-label classification of CVDs from ECG images. The proposed framework can accurately identify multiple co- occurring CVDs from a single ECG image. The experimental results show that the proposed framework can achieve high accuracy in multi-label classification of CVDs.

OBJECTIVES

- To implement a CNN-based model for multi-label classification of cardiovascular diseases from ECG images.
- To optimize preprocessing techniques such as image resizing (800KB to 300KB) and normalizing.
- To optimize training performance using Adam for better accuracy and faster convergence.
- 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.

LITERATURE SURVEY

S. No.	Title	Author Name(s)	Methodology Used	Limitations
1	Prediction of Heart Disease Using a Combination of Machine Learning and Deep Learning	Rohit Bharti, Aditya Khamparia, Mohammad Shabaz, Gaurav Dhiman, Sagar Pande, and Parneet Singh	● Applied machine learning and deep learning on the UCI Machine Learning Heart Disease dataset. ● Handled irrelevant features using Isolation Forest. Normalized data. Evaluated models using accuracy and confusion matrix.	● Future work needed to increase dataset size. ● Explore optimization and normalizatio n techniques. ● Integrate models with multimedia.
2	Cardiac Conduction Model for	Mario A. Quiroz-Juárez, Omar Jiménez-	● Used an extended heterogeneous	● Potential for extension to

	Generating 12 Lead ECG Signals With Realistic Heart Rate Dynamics	Ramírez, Rubén Vázquez-Medina, Elena Ryzhii, Maxim Ryzhii, and José L. Aragón	oscillator model to generate 12-lead ECG waveforms. ● Model includes pacemakers, muscles, RR-tachogram, and heart rate variability. ● Calculated 12-lead ECG using a weighted linear combination.	simulate intracardiac signals and use as a testbed for implantable devices. ● Possible limitations due to complexity, computational demands, and reliance on theoretical constructs.
3	Cardiac Disorder Classification by Electrocardiogram Sensing Using Deep Neural Network	Ali Haider Khan, Muzammil Hussain, and Muhammad Kamran Malik	● Used a dataset of 11,148 12-lead ECG images. ● Applied Single Shot Detection (SSD) MobileNet v2-based Deep Neural Network to detect four cardiac abnormalities.	● Need for training on a larger dataset with more abnormalities. ● Exploration of advanced feature extraction. Research on domain adaptation learning.
4	Electrocardiogram Heartbeat Classification Using Machine Learning and Ensemble Convolutional Neural Network-Bidirectional Long Short-Term Memory Technique	Neenu Sharma, Ramesh Kumar Sunkaria, and Amandeep Kaur	● Used the MIT-BIH arrhythmias database. ● Applied machine learning ensemble techniques, including an ensemble of CNN and Bi-LSTM. Incorporated synthetic minority oversampling. ● Utilized time-series feature extraction.	● Future work to focus on increasing dataset size. ● Application of various optimization techniques. ● Exploration of different data normalization. ● Integration

				of models with multimedia.
5	Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks	Serkan Kiranyaz, Turker Ince, and Moncef Gabbouj	● Used adaptive 1-D CNNs to fuse feature extraction and classification in a patient-specific approach.	● Future work on hardware implementation. Limitations in characterizing anomaly beats. ● Lack of support for incremental or active learning.
6	Semi-Supervised Learning for Multi-Label Cardiovascular Diseases Prediction: A Multi-Dataset Study	Rushuang Zhou, Lei Lu, Zijun Liu, Ting Xiang, Zhen Liang, David A. Clifton, Yining Dong, and Yuan-Ting Zhang	● Proposed ECGMatch model with ECGAugment module for data augmentation. ● A hyperparameter-efficient framework for pseudo-label generation. ● A label correlation alignment module.	● Class imbalance problem in ECG datasets.
7	Adaptive R-Peak Detection on Wearable ECG Sensors for High-Intensity Exercise	Elisabetta De Giovanni, Tomas Teijeiro, Grégoire P. Millet, and David Atienza	● Based on unsupervised learning, Bayesian filtering, and non-linear normalization for adaptive detection of R-peaks during high-intensity exercise.	● Future work on hardware implementation. ● Dataset size is relatively small. ● Only considered cycling and single-lead

				ECG.
8	Automatic ECG Diagnosis Using Convolutional Neural Network	Roberta Avanzato and Francesco Beritelli	● Used a CNN to classify ECG signals, trained on a dataset of 4000 ECG signal instances.	● Future work on increasing dataset size. ● Exploration of various optimization techniques. ● Investigation of different data normalization. ● Integration of models with multimedia.
9	On Merging Feature Engineering and Deep Learning for Diagnosis, Risk Prediction and Age Estimation Based on the 12-Lead ECG	Eran Zvuloni, Jesse Read, Antônio H. Ribeiro, Antonio Luiz P. Ribeiro, and Joachim A. Behar	● Utilized a dataset of 2.3M 12-lead ECG recordings. ● Applied a combination of feature engineering and deep learning using a merged FE+DL model.	● Future work on increasing dataset size. ● Application of various optimization techniques. ● Exploration of different data normalization. ● Integration of models with multimedia.
10	Detection of Cardiovascular Diseases in ECG Images Using Machine	Mohammed B. Abubaker and Bilal Babayiğit	● Used a public ECG image dataset of cardiac patients. ● Investigated transfer learning using	● Future work could focus on increasing the size of the dataset

	Learning and Deep Learning Methods		pretrained deep neural networks SqueezeNet and AlexNet. Proposed a new CNN architecture.	and applying various optimization techniques. ● Different ways of normalizing the data could be explored. ● More ways to integrate models with multimedia.

ARCHITECTURE DIAGRAM OF THE PROPOSED MODEL

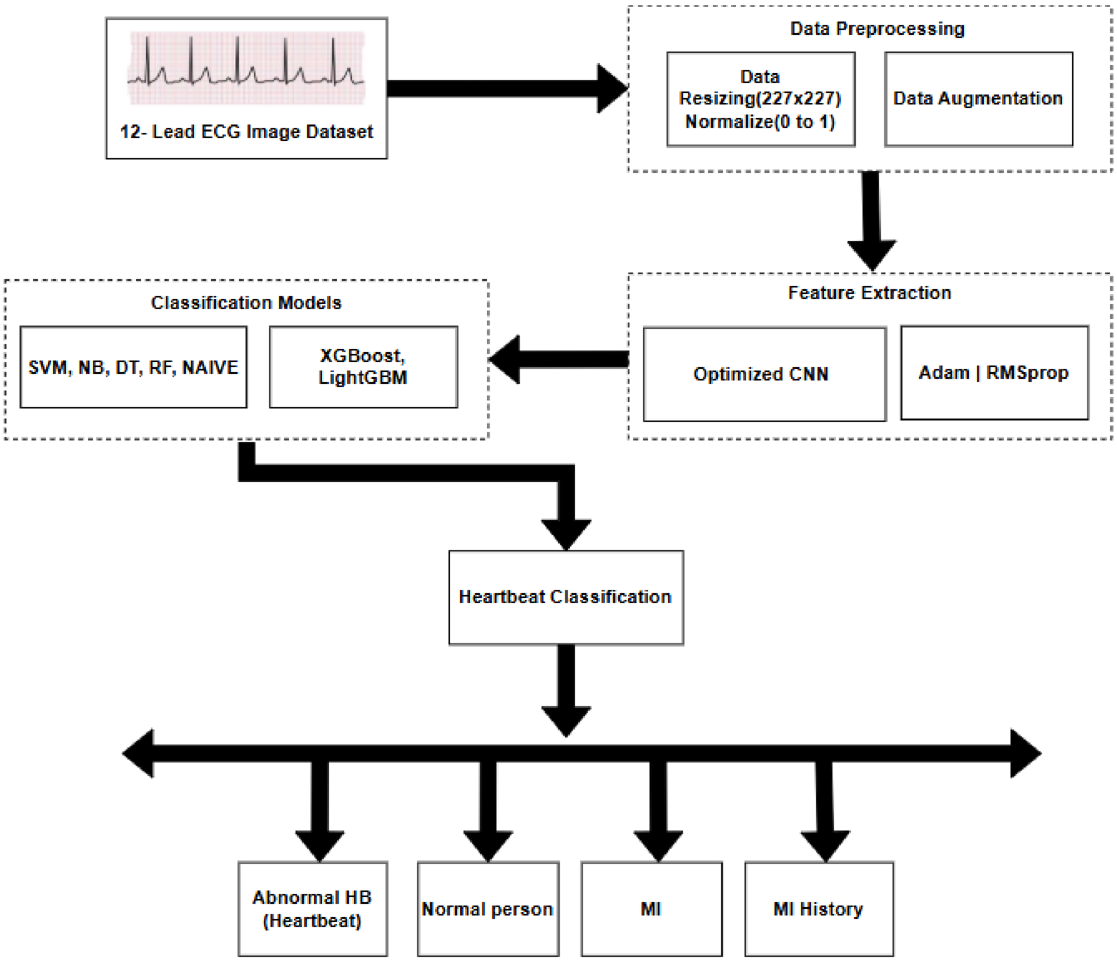


Fig. 1. Architecture diagram for the proposed model

MODULE DESCRIPTION

Data Pre-processing:

- **Dataset Structure & Sample Distribution:** The dataset is organized into subfolders, where each subfolder represents a different class of ECG images. It contains 929 images distributed across 4 classes, with folder names serving as class labels.

- **Image Processing:** ECG images often contain headers and footers that do not hold diag- nostic value, so they are cropped. The images are then resized to a uniform dimension of 227x227 pixels to maintain consistency.

- **Normalization:** Pixel values are normalized by dividing by 255.0, scaling them to the range [0,1]. This helps in faster convergence during training.
- **Label Encoding & Data Splitting:** Class labels are converted into a one-hot encoded for-mat for CNN training. The dataset is split into 80% training and 20% testing to evaluate model performance on unseen data.

- **Data Augmentation:** Techniques such as random rotation, shifts, shears, zooms, and flips are applied using ImageDataGenerator to improve model generalization and reduce overfit- ting.

Feature-Extraction:

- **CNN Model Definition:** A CNN model is defined using Keras. This model includes multiple convolutional layers, pooling layers, batch normalization, LeakyReLU activation, and dropout for regularization.
- **Hyperparameter Tuning:** Keras Tuner with Bayesian Optimization is used to find the optimal hyperparameters for the CNN model. This includes tuning the number of filters in convolutional layers, dropout rates, dense layer units, and learning rate.
- **Training with Early Stopping:** The CNN model is trained using the augmented training data, and early stopping is used to prevent overfitting.
- **Feature Extraction Layer:** The output of the dense layer before the final softmax layer is used as the feature vector. This is achieved by creating a new Keras model (encoder) that outputs the activations of this dense layer.
- **Feature Extraction:** The encoder model is used to extract features from both the training

and testing sets.

- **Feature Normalization:** The extracted features are normalized using StandardScaler. This scales the features to have zero mean and unit variance, which can improve the performance of the classifiers.
- **Saving the Feature Extractor:** The encoder model is saved for future use.

Model Selection:

- **Handles High Dimensionality:** ECG images have many features. Gradient boosting models effectively manage this, unlike KNN which struggles with numerous dimensions.
- **Captures Complex Patterns:** ECG signals have subtle variations. LightGBM and XGBoost excel at learning these intricate relationships, where simpler models may fail.
- **Reduces Overfitting:** Techniques in these models prevent overfitting, a risk with Decision Trees and Random Forests on complex data.
- **Feature Importance:** Provides insights into which ECG features are most indicative of specific heart conditions.
- **Efficiency and Speed:** Gradient boosting is generally faster, crucial for large medical datasets.

Why LightGBM Often Edges Out XGBoost?

1. **Speed:** LightGBM is known for its faster training speeds, especially on large datasets.
2. **Memory Efficiency:** It uses less memory, allowing for the handling of even larger ECG image datasets.
3. **Accuracy:** In many cases, LightGBM can achieve comparable or even better accuracy than XGBoost.
4. **Direct Categorical Feature Support:** LightGBM can handle categorical features directly, which can be useful in some ECG analysis scenarios.

Training and Evaluation:

- **Classifier Training:** Each classifier is trained using the extracted features from the training set and the corresponding labels.
- **Prediction:** The trained classifiers are used to predict the labels for the test set.

- **Performance Evaluation:** The performance of each classifier is evaluated using classification reports and confusion matrices.
- **Metrics:** The classification report includes precision, recall, F1-score, and support for each class, as well as macro and weighted averages. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives.

Visualization of Results:

- **Confusion Matrices:** Confusion matrices for each classifier are visualized using heatmaps. This provides a clear view of the classifier's performance for each class.
- **Performance Metrics Comparison:** Bar plots are used to compare the F1-scores, precision, and recall of different classifiers. This helps in identifying the best-performing classifier.
- **Training History Plots:** The training and validation accuracy and loss of the CNN model are plotted against the number of epochs. This helps in understanding the model's learning progress and identifying potential issues like overfitting or underfitting.

ALGORITHM:

Input : 12 lead ECG image

Output: Class of the ECG image in Myocardial infrarction, History of MI, Abnormal and Normal.

DATA PRE-PROCESSING

```
Input: dataset_path, img_size

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


1. categories = ["Normal", "Abnormal", "MI", "History_MI"]

2. data = [], labels = []

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))
4. data = numpy.array(data), labels = numpy.array(labels)
5. Return data, labels
```

BUILD CNN MODEL

Input: input_shape, num_classes
Output: compiled Keras model

```
1. Input Layer: Create input_layer.
2. Stack Branch: Apply repeated Conv2D, LeakyReLU, BatchNorm, MaxPooling.
3. Full Branch: Flatten, Dense, LeakyReLU, BatchNorm, Dropout, Reshape, Conv2D,
   Concate- nate, Dropout, UpSampling.
4. Concatenate: Combine Stack and Full Branch outputs.
5. Final Layers: Conv2D, Flatten, Dense, Dropout, Dense (softmax).
6. Create Model: Model(inputs, outputs).
7. Compile: model.compile(adam, sparse_categorical_crossentropy, accuracy).
8. Summary: model.summary().
9. Return: model.
```

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:
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)
2. Extract Training Features:
   - X_train_features = feature_extractor.predict(X_train)
3. Extract Validation Features:
   - X_val_features = feature_extractor.predict(X_val)
4. Return Extracted Features:
   - Return X_train_features, X_val_features
```

TESTING

```
Input: image_path, target_size
Output: Predictions
1. preprocess_image: Load, resize, normalize, expand image.
2. Load: model, extractor, classifiers.
3. Preprocess: image.
4. If image:
   a. Extract: features.
   b. Scale: features.
   c. Predict: with classifiers, print results.
```

TABLE I

REUSED AND NEWLY WRITTEN

	Detection of Cardiovascular Diseases in ECG Images Using Machine Learning and Deep Learning Methods	**Cardiac Disorder Classification by Electrocardiogram Sensing Using Deep Neural Network**	**Proposed Model**
Goal	Develop a new lightweight CNN architecture for cardiovascular disease prediction using ECG images.	Develop an efficient automated model for the diagnosis of cardiac disorder on ECG that can feasibly be implemented on portable healthcare devices.	To implement a CNN- based deep learning model for detecting and classifying different ECG patterns associated with cardiovascular diseases.
Methodology	Explores transfer learning (SqueezeNet, AlexNet), proposes a new CNN, and uses these models for feature extraction with traditional ML.	Uses Single Shot Detection (SSD) MobileNet v2-based Deep Neural Network architecture.	Uses a CNN and optimizes using Adam, then trains and saves the model for future use.
Dataset	Public ECG images dataset of cardiac patients.	A dataset of 11,148 standard 12-lead-based ECG images manually collected from healthcare institutes and annotated by domain experts.	Dataset from Mendeley Data.
Novelty	Introduces a new lightweight CNN, achieves high	Proposes a generalized	Proposes a novel hybrid CNN

	accuracy, and demonstrates improved performance when used for feature extraction.	methodology to process all formats of ECG images, which are often nonuniform across healthcare institutes.	architecture with dual branches (stackfull), using both convolutional and fully connected layers for feature extraction and upsampling, optimizing for lightweight and efficient performance.
Results	Achieves 98.23% accuracy with the proposed CNN, and 99.79% accuracy when used for feature extraction with Naïve Bayes.	Achieves 98% accuracy in detecting four major cardiac abnormalities.	Achieves 99% accuracy for LightGBM classifier with the proposed CNN model for feature extraction.

PROPOSED METHOD PROS AND CONS

PROS

1. **Hybrid Approach**: Combines deep learning (CNN) for feature extraction and traditional ML models for classification, enhancing accuracy.
2. **Class Imbalance Handling**: Class weights address imbalance, improving model fairness.
3. **Data Augmentation**: Increases dataset diversity and helps prevent overfitting.
4. **Model Tracking**: Saves the best model and logs training progress.
5. **Scalable**: Easily adaptable to larger datasets and more classes.
6. **Multiple Classifiers**: Comparing different ML classifiers may lead to better performance.

CONS

1. **Complexity**: Combining CNN and ML models increases design and implementation complexity.

2. **Feature Extraction Bottleneck**: CNN feature extraction may not always produce optimal results for ML models.
3. **Overfitting Risk**: Traditional ML models may overfit on extracted features without regularization.
4. **Training Time**: Training both CNN and ML models can be resource-intensive and time-consuming.
5. **Manual Preprocessing**: Needs careful handling of image preprocessing for optimal results

IMPLEMENTATION SNAPSHOTS

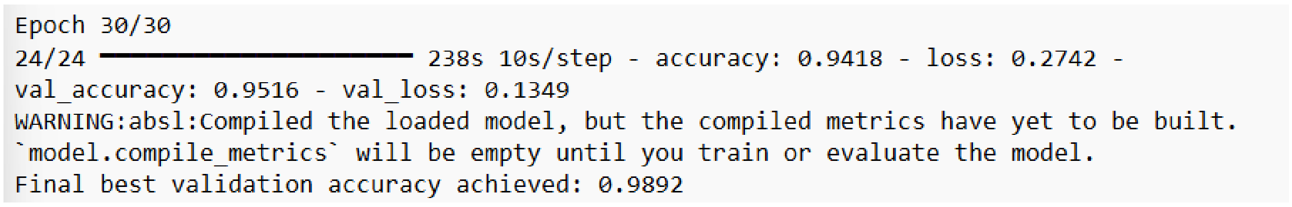
CNN MODEL

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
concatenate, BatchNormalization, LeakyReLU, Reshape, UpSampling2D

def build_proposed_cnn(input_shape=(227, 227, 3), num_classes=4):
    input_layer = Input(shape=input_shape)

    # Stack Branch
    x = Conv2D(64, (3, 3), padding='same')
    (input_layer) x = LeakyReLU(negative_slope=0.1)
    (x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((6, 6), strides=3)(x)

    x = Conv2D(128, (3, 3), padding='same')
    (x) x = LeakyReLU(negative_slope=0.1)
    (x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((6, 6), strides=3)(x)

    x = Conv2D(224, (3, 3), padding='same')
    (x) x = LeakyReLU(negative_slope=0.1)
    (x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((6, 6), strides=3)(x)
    stack_branch_output = x # Shape: (None, 6, 6,
    224) # Full Branch
    y = Dense(16)(Flatten()(input_layer))
    y = LeakyReLU(negative_slope=0.1)
    (y) y = BatchNormalization()(y)
    y = Dropout(0.5)(y)
    # Reshape the Dense layer output to a 4D tensor for
    Conv2D y = Reshape((1, 1, 16))(y) # Shape: (None, 1, 1,
    16)
    y1 = Conv2D(32, (2, 2), strides=1, padding='same')
    (y)  y2  =  Conv2D(64,  (3,  3),  strides=2,
    padding='same')(y)  y  =  concatenate([y1,  y2])  #
    Shape: (None, 1, 1, 96)
    y = Dropout(0.5)(y)

    # Upsample the Full Branch output to match the Stack Branch spatial
    dimensions y = UpSampling2D(size=(6, 6))(y) # Shape: (None, 6, 6, 96)

    # Concatenate branches along the channel dimension
    combined = concatenate([stack_branch_output, y]) # Shape: (None, 6, 6, 224 + 96)
    combined = Conv2D(256, (1, 1))(combined)
    combined = Flatten()(combined)
    combined = Dense(512)(combined)
    combined = Dropout(0.5)(combined)
    output_layer = Dense(num_classes, activation='softmax')(combined)

    model = Model(inputs=input_layer, outputs=output_layer)
    return model

# Build and compile the model
model = build_proposed_cnn()
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy',
metrics=['accuracy']) model.summary()
```

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 227, 227, 3)	0	-
conv2d (Conv2D)	(None, 227, 227, 64)	1,792	input_layer[0][0]
leaky_re_lu (LeakyReLU)	(None, 227, 227, 64)	0	conv2d[0][0]
batch_normalization (BatchNormalization)	(None, 227, 227, 64)	256	leaky_re_lu[0][0]
flatten (Flatten)	(None, 154587)	0	input_layer[0][0]
max_pooling2d (MaxPooling2D)	(None, 74, 74, 64)	0	batch_normalization[0][0]
dense (Dense)	(None, 16)	2,473,408	flatten[0][0]
conv2d_1 (Conv2D)	(None, 74, 74, 128)	73,856	max_pooling2d[0][0]

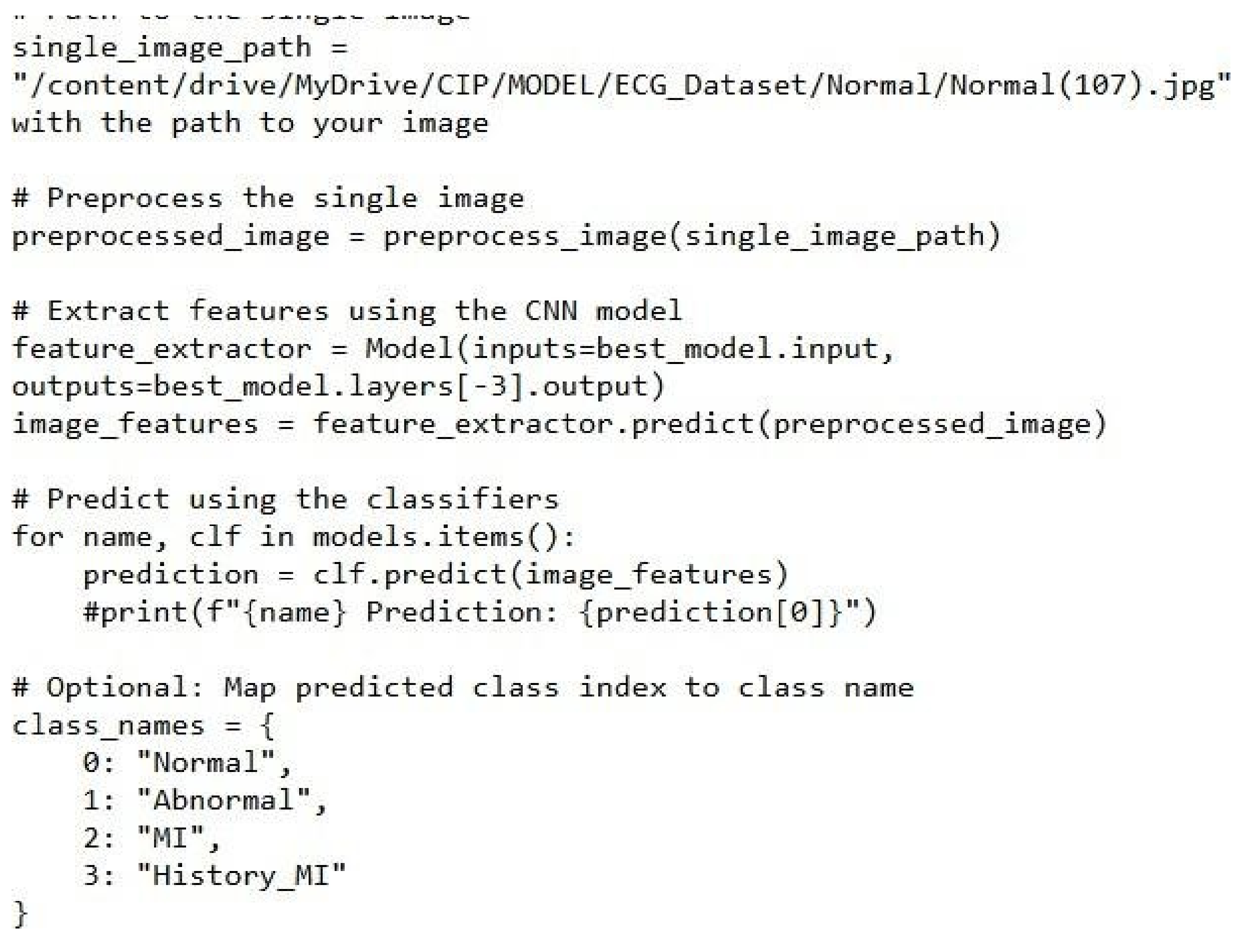
leaky_re_lu_3 (LeakyReLU)	(None, 16)	0	dense[0][0]
leaky_re_lu_1 (LeakyReLU)	(None, 74, 74, 128)	0	conv2d_1[0][0]
batch_normalization_3 (BatchNormalization)	(None, 16)	64	leaky_re_lu_3[0][0]
batch_normalization_1 (BatchNormalization)	(None, 74, 74, 128)	512	leaky_re_lu_1[0][0]
dropout (Dropout)	(None, 16)	0	batch_normalization_3[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 23, 23, 128)	0	batch_normalization_1[0][0]
reshape (Reshape)	(None, 1, 1, 16)	0	dropout[0][0]
conv2d_2 (Conv2D)	(None, 23, 23, 224)	258,272	max_pooling2d_1[0][0]
conv2d_3 (Conv2D)	(None, 1, 1, 32)	2,080	reshape[0][0]
conv2d_4 (Conv2D)	(None, 1, 1, 64)	9,280	reshape[0][0]
leaky_re_lu_2 (LeakyReLU)	(None, 23, 23, 224)	0	conv2d_2[0][0]
concatenate (Concatenate)	(None, 1, 1, 96)	0	conv2d_3[0][0], conv2d_4[0][0]
batch_normalization_2 (BatchNormalization)	(None, 23, 23, 224)	896	leaky_re_lu_2[0][0]
dropout_1 (Dropout)	(None, 1, 1, 96)	0	concatenate[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 224)	0	batch_normalization_2[0][0]
up_sampling2d (UpSampling2D)	(None, 6, 6, 96)	0	dropout_1[0][0]
concatenate_1 (Concatenate)	(None, 6, 6, 320)	0	max_pooling2d_2[0][0], up_sampling2d[0][0]
conv2d_5 (Conv2D)	(None, 6, 6, 256)	82,176	concatenate_1[0][0]
flatten_1 (Flatten)	(None, 9216)	0	conv2d_5[0][0]
dense_1 (Dense)	(None, 512)	4,719,104	flatten_1[0][0]
dropout_2 (Dropout)	(None, 512)	0	dense_1[0][0]
dense_2 (Dense)	(None, 4)	2,052	dropout_2[0][0]

ALGORITHM (INPUT & OUTPUT SNAPSHOTS)



```
Epoch 30/30
24/24 ━━━━━━━━━━━━━━━━━━━━ 238s 10s/step - accuracy: 0.9418 - loss: 0.2742 -
val_accuracy: 0.9516 - val_loss: 0.1349
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built.
`model.compile_metrics` will be empty until you train or evaluate the model.
Final best validation accuracy achieved: 0.9892
```

INPUT



```
# Path to the single image
single_image_path =
"/content/drive/MyDrive/CIP/MODEL/ECG_Dataset/Normal/Normal(107).jpg"
with the path to your image

# Preprocess the single image
preprocessed_image = preprocess_image(single_image_path)

# Extract features using the CNN model
feature_extractor = Model(inputs=best_model.input,
outputs=best_model.layers[-3].output)
image_features = feature_extractor.predict(preprocessed_image)

# Predict using the classifiers
for name, clf in models.items():
    prediction = clf.predict(image_features)
    #print(f"{name} Prediction: {prediction[0]}")

# Optional: Map predicted class index to class name
class_names = {
    0: "Normal",
    1: "Abnormal",
    2: "MI",
    3: "History_MI"
}
```

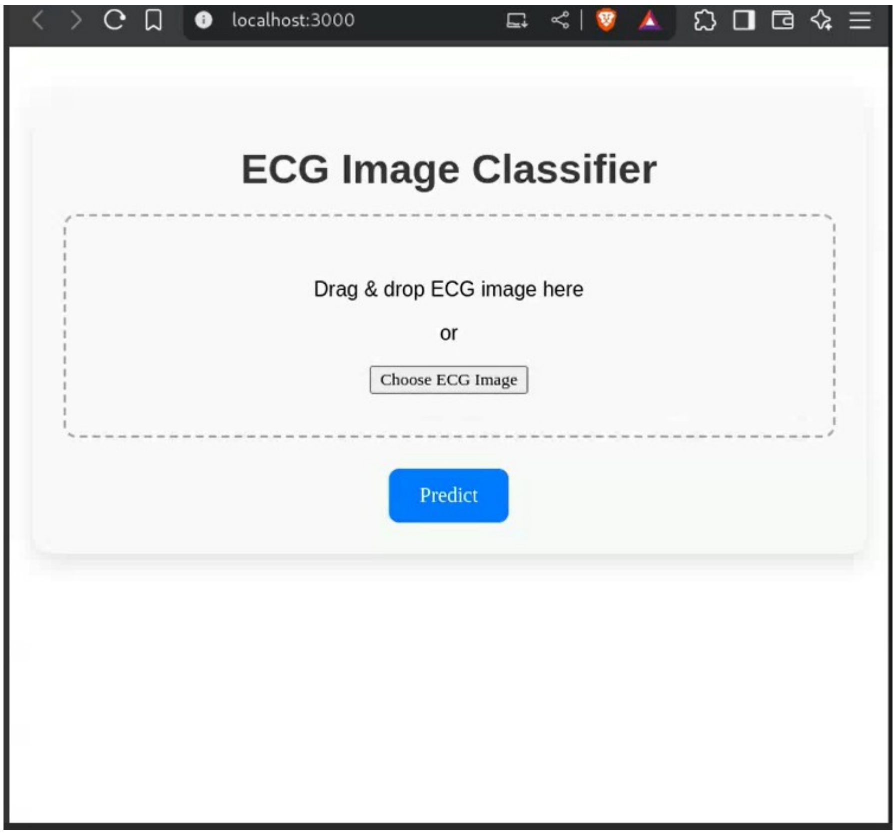
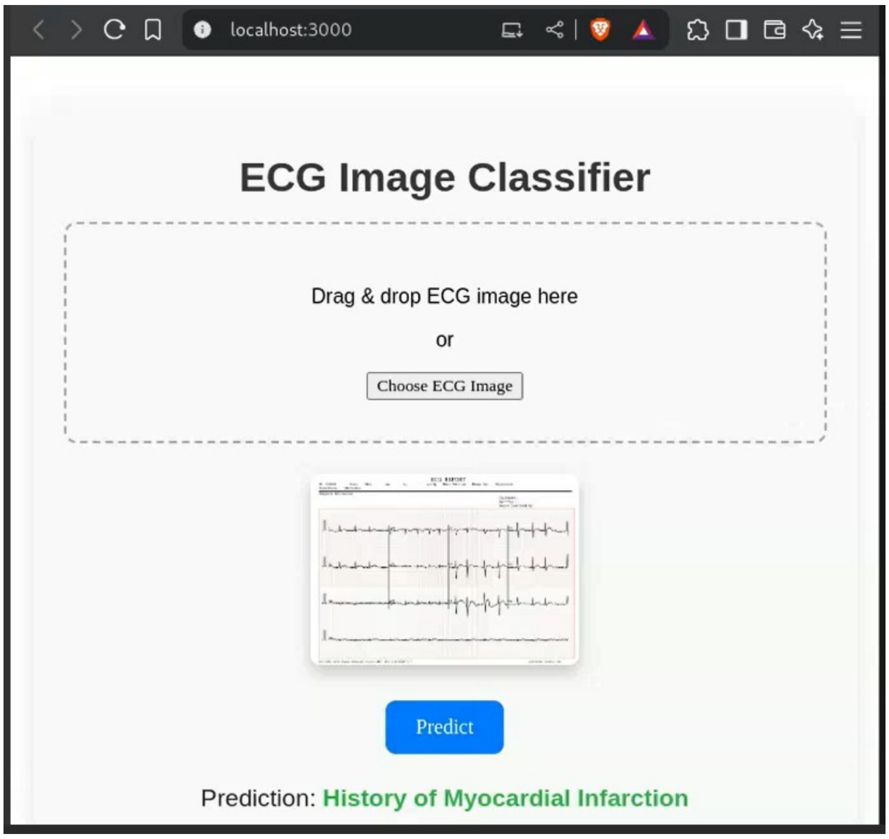


Fig. 2. ECG image classifier drag & drop

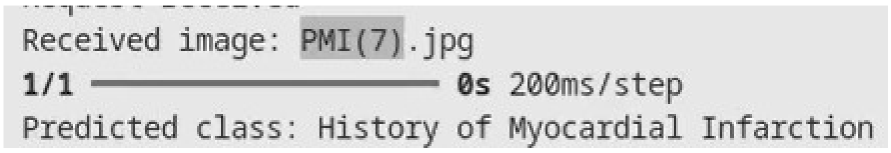
OUTPUT

```
Output:
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 308ms/step
SVM Prediction: Normal
K-NN Prediction: Normal
Decision Tree Prediction: Normal
Random Forest Prediction: Normal
Naive Bayes Prediction: Normal
XGBoost Prediction: Normal
LightGBM Prediction: Normal
```



Prediction: History of Myocardial Infarction

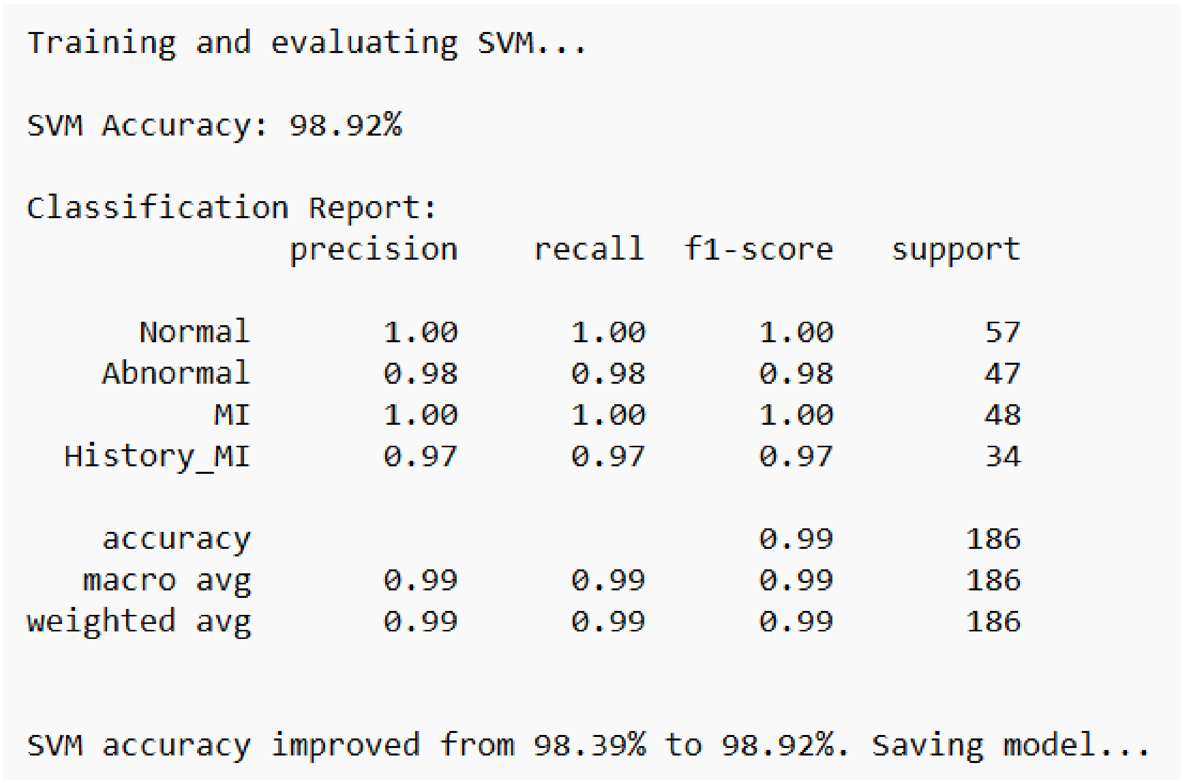
Fig. 3. Prediction of ECG Image classifier



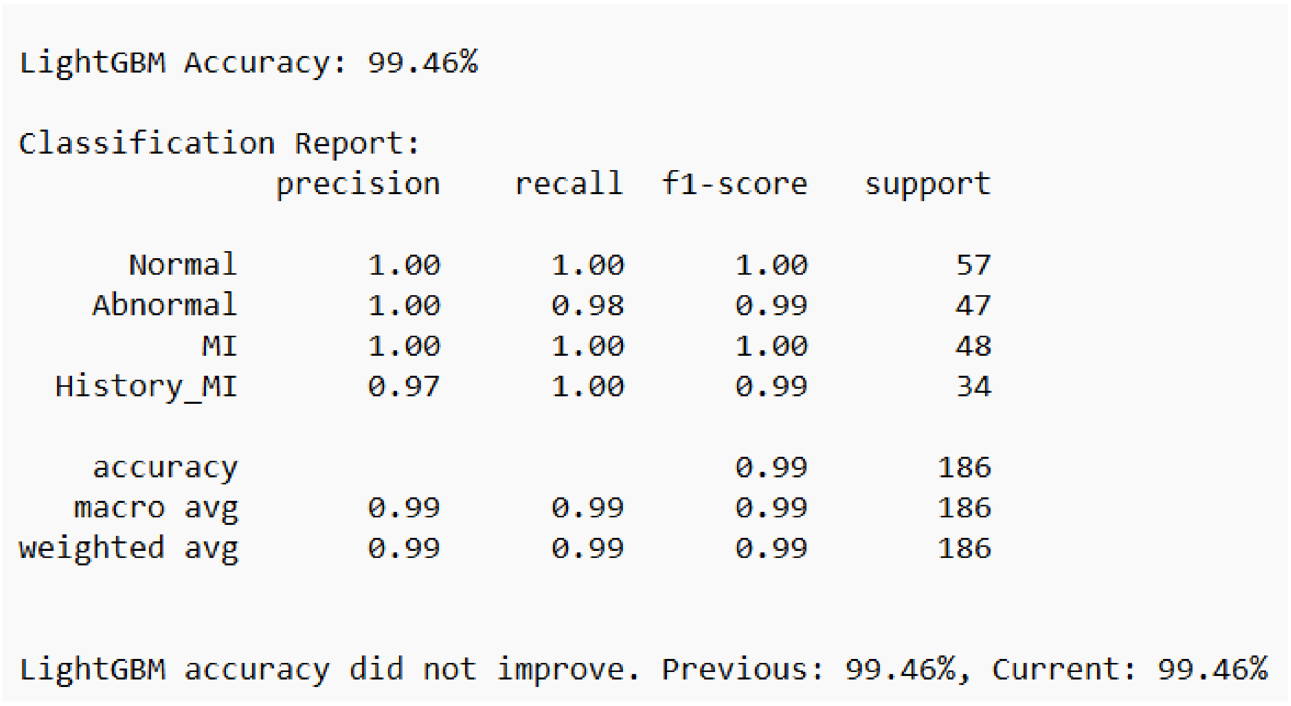
```
Received image: PMI(7).jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 200ms/step
Predicted class: History of Myocardial Infarction
```

Fig. 4.   Sample output

EVALUATION METRICS



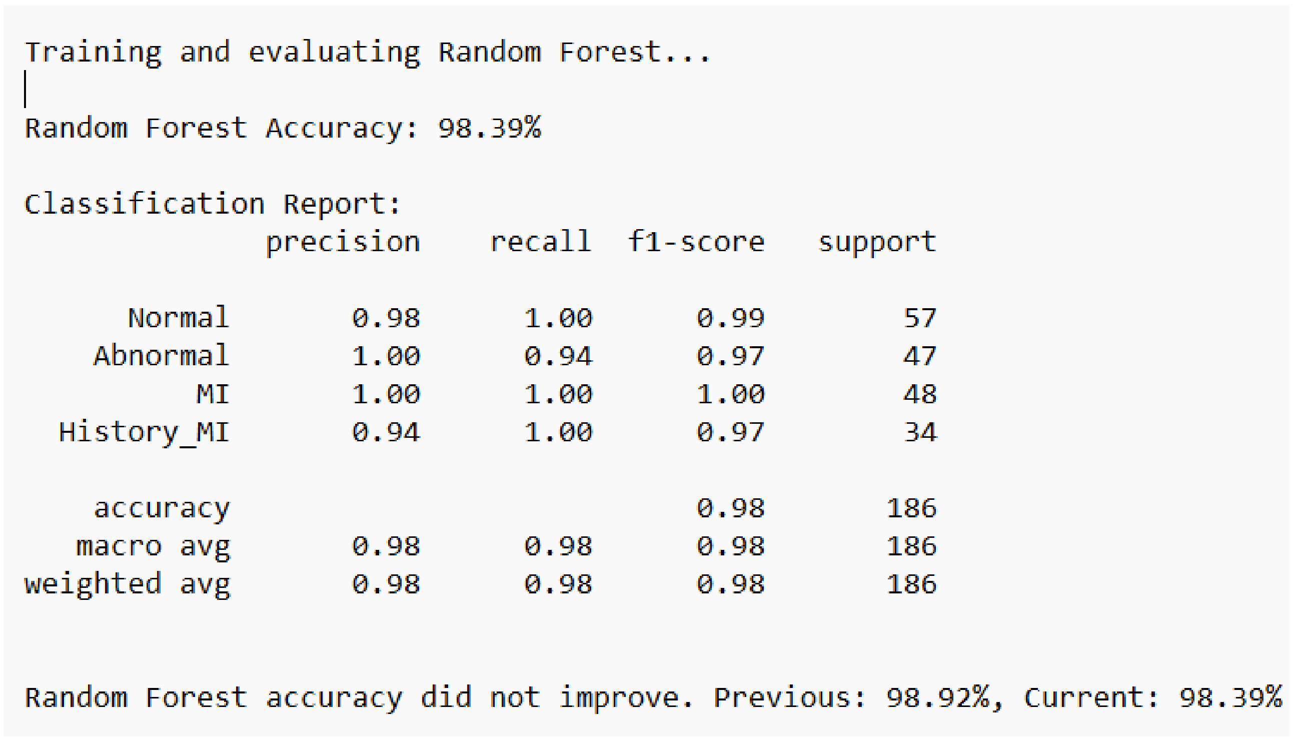
```
Training and evaluating SVM...

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


SVM accuracy improved from 98.39% to 98.92%. Saving model...
```



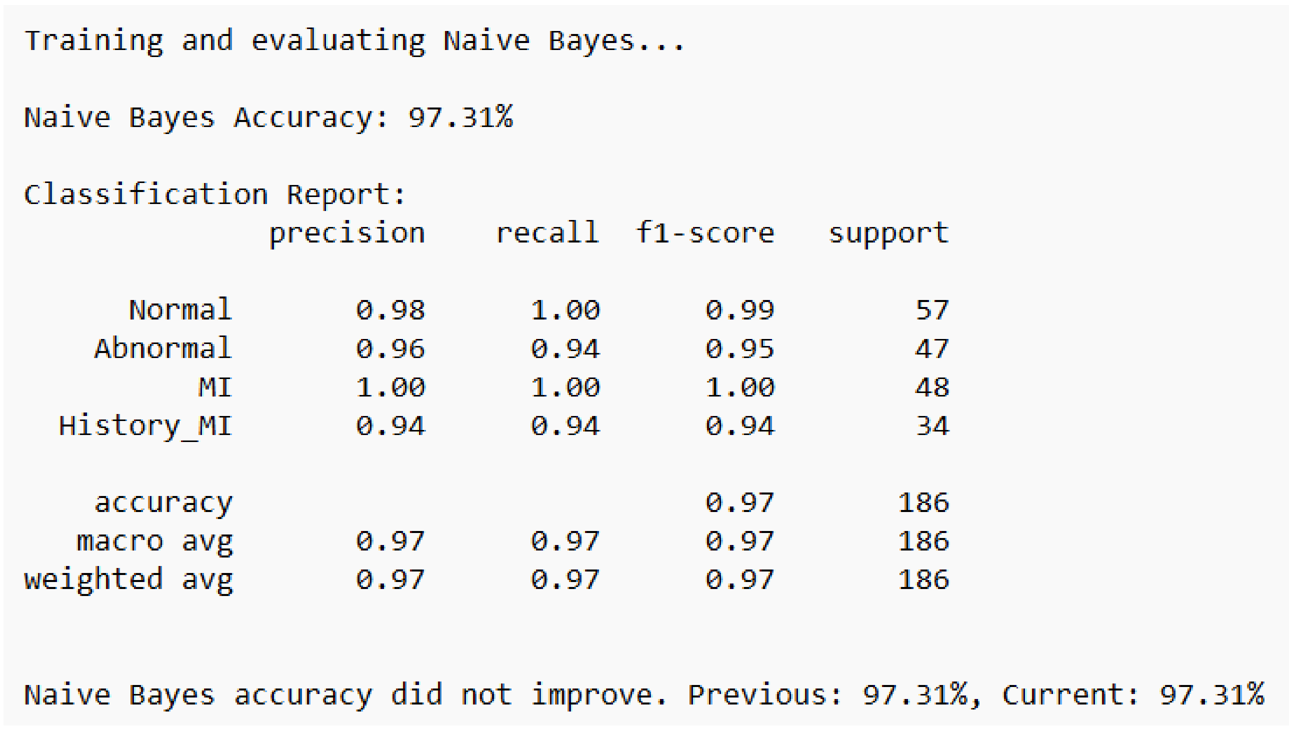
```
LightGBM Accuracy: 99.46%

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


LightGBM accuracy did not improve. Previous: 99.46%, Current: 99.46%
```

```
Training and evaluating K-NN...

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


K-NN accuracy did not improve. Previous: 97.85%, Current: 97.85%
```

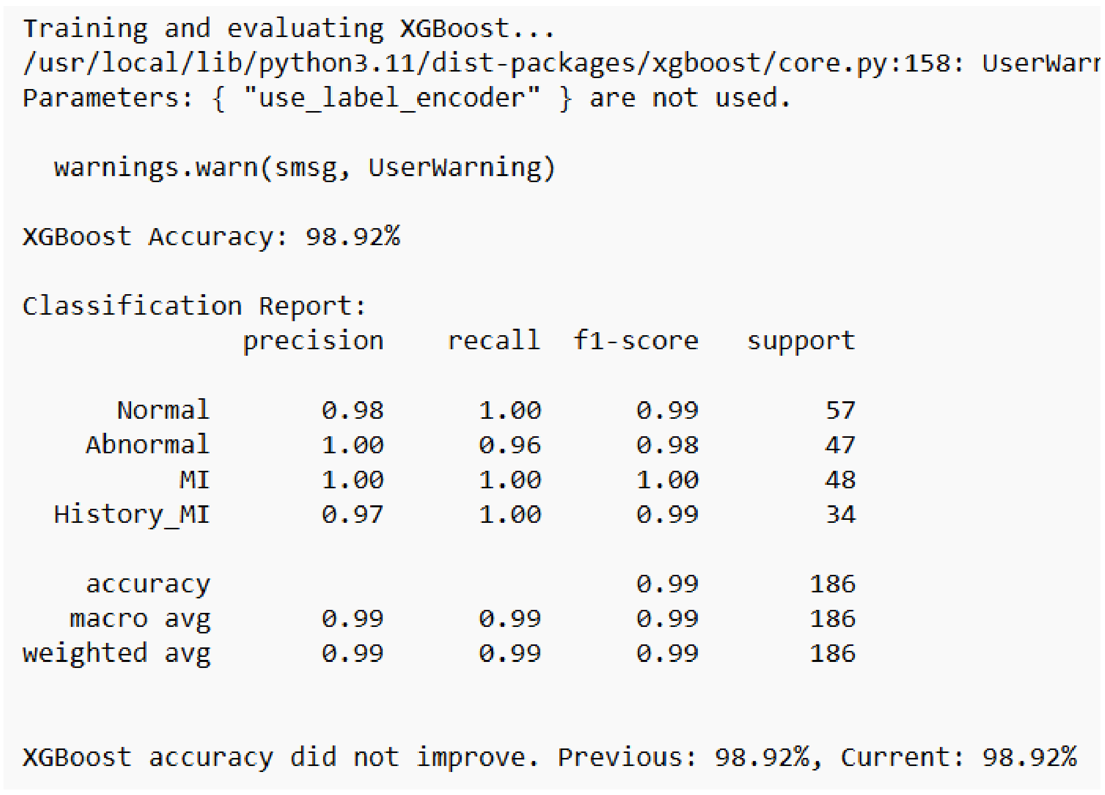
```
Training and evaluating Decision Tree...

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


Decision Tree accuracy improved from 96.24% to 97.85%. Saving model...
```



```
Training and evaluating Random Forest...

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


Random Forest accuracy did not improve. Previous: 98.92%, Current: 98.39%
```



```
Training and evaluating Naive Bayes...

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


Naive Bayes accuracy did not improve. Previous: 97.31%, Current: 97.31%
```



```
Training and evaluating XGBoost...
/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarr
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)

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


XGBoost accuracy did not improve. Previous: 98.92%, Current: 98.92%
```

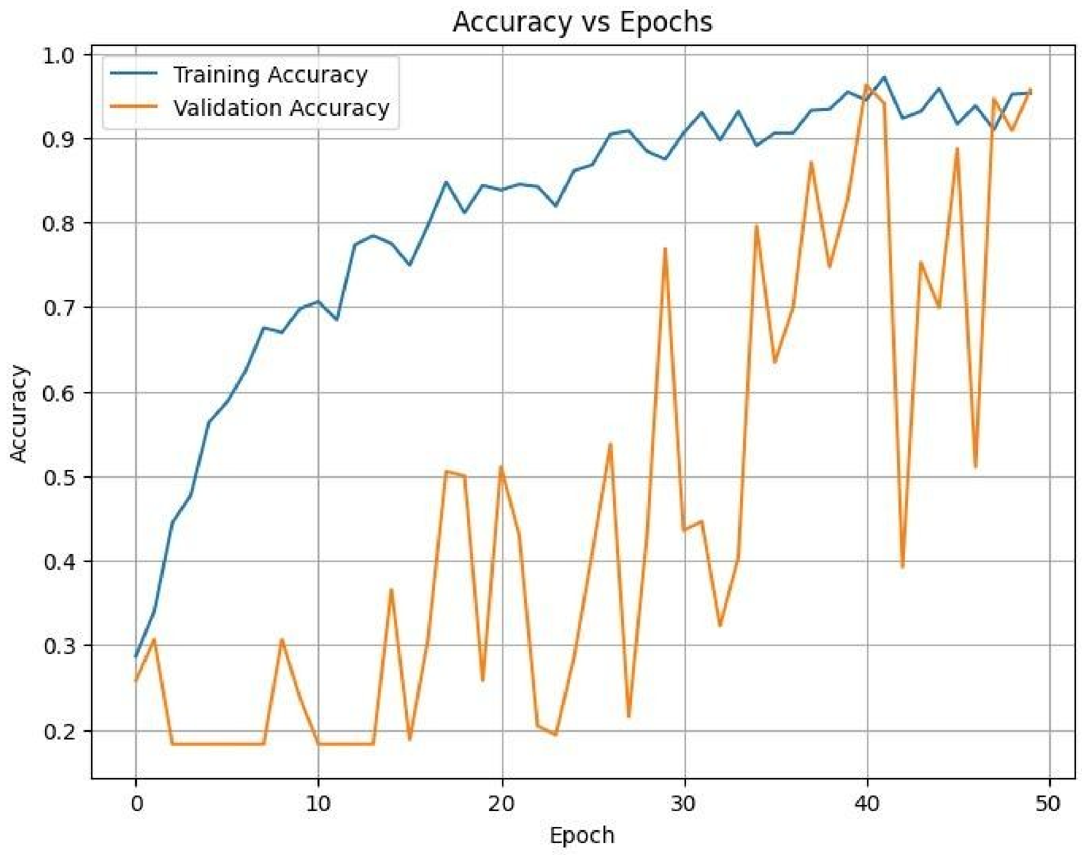


Fig. 5. Training Progress for our proposed CNN model on the ECG images Dataset with LR 0.001(Epoch Vs Accuracy).

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.