UNIVERSITY COLLEGE OF ENGINEERING , VILLUPURAM
(Constituent College of Anna University , Chennai)
Kakuppam , Villupuram , 605103

Department of Information Technology-2025

"Optimized Hybrid CNN-BiLSTM Approach for Accurate and Efficient Heart Disease prediction"

Second Review-2-(24-04-2025)

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

GUIDED BY
Mr.S.PRABHAKARAN
Department Of Information Technology
UCEV

ABSTRACT

This project presents a deep learning approach for heart disease classification using a hybrid Convolutional Neural Network (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) model. The model is trained and evaluated on the Herat Disease dataset, which contains patient health indicators and binary classification labels indicating the presence or absence of heart disease.

The preprocessing phase involves:

- Feature normalization using StandardScaler
- One-hot encoding of the binary target variable
- Reshaping the feature matrix to match the input requirements of a Conv1D layer

ABSTRACT(Cont…)

The proposed model architecture combines:

- Multiple Conv1D layers for feature extraction
- BatchNormalization and Dropout for regularization
- A Reshape layer to adapt the convolutional output for the LSTM layers
- Two stacked Bidirectional LSTM layers to capture temporal dependencies in both directions
- Dense layers for classification, with a softmax activation function for binary output

The model is compiled using the Adam optimizer and trained for 50 epochs with a batch size of 32. Performance is evaluated using test accuracy, and training progress is visualized through accuracy and loss curves.

LITERATURE SURVEY

S. No.	Title	Author(s) & Year	Algorithm / Model Used	Pros	Cons
1	**Heart Disease Prediction Using Deep Learning**	Kanumuri Vinay Varma et al. (2024)	Gaussian Naive Bayes	Simple, interpretable, handles continuous features, good baseline	Not suitable for nonlinear relationships; sensitive to data distribution
2	**A Novel Early Detection and Prevention of Coronary Heart Disease Framework Using Hybrid Deep Learning Model and Neural Fuzzy Inference System**	B. Ramesh, Kuruva Lakshmanna (2024)	O-SBGC-LSTM + Fuzzy Inference System	Very high accuracy (>98%), incorporates fuzzy logic for prevention	Complex architecture, computationally intensive

S. No.	Title	Author(s) & Year	Algorithm / Model Used	Pros	Cons
3	**Electrocardiogram-based Deep Learning to Predict Mortality in Paediatric and Adult Congenital Heart Disease**	Joshua Mayourian et al. (2024)	Convolutional Neural Network (CNN) on ECG	Temporal validation, high predictive value, applicable to large dataset	Low precision-recall score; real-world generalization needs validation
4	**Comprehensive Review of Deep Learning-Based Models for Heart Disease Prediction**	Chunjie Zhou et al. (2024)	CNN, LSTM, CNN-LSTM, Integrated DL Models	Extensive review of DL models and hybrid approaches	Survey paper – no original experiments; performance varies widely across models
5	**Race, Sex, and Age Disparities in the Performance of ECG Deep Learning Models Predicting Heart Failure**	Dhamanpreet Kaur et al. (2024)	CNN on ECG data	Highlights demographic biases, proposes tailored thresholds	Shows performance disparity in young Black women; complexity in practical deployment

PROPOSED SYSTEM

- The proposed work aims to design a deep learning-based model to predict heart disease using clinical data. The system will combine Convolutional Neural Networks (CNN) and Bidirectional Long Short-Term Memory (BiLSTM) networks to create a hybrid model capable of learning both spatial and sequential patterns in the data. This approach is intended to improve prediction accuracy over traditional machine learning models.
- The dataset includes various features such as age, cholesterol levels, resting blood pressure, and other medical attributes. These features are first preprocessed by handling missing values, normalizing the data using StandardScaler, and applying one-hot encoding to the target labels. The input data is then reshaped to be compatible with Conv1D layers.

PROPOSED SYSTEM (Cont…)

- The CNN layers will extract local feature patterns from the data, followed by reshaping and passing the output to BiLSTM layers. The BiLSTM layers will learn temporal dependencies and feature relationships in both forward and backward directions. The final classification is done using fully connected Dense layers with softmax activation.
- The model will be trained using the Adam optimizer and evaluated based on accuracy and loss metrics. The goal is to develop a robust and reliable prediction model that can assist in early diagnosis and improve clinical decision-making for heart disease patients.

PROPOSED SYSTEM ARCHITECTURE

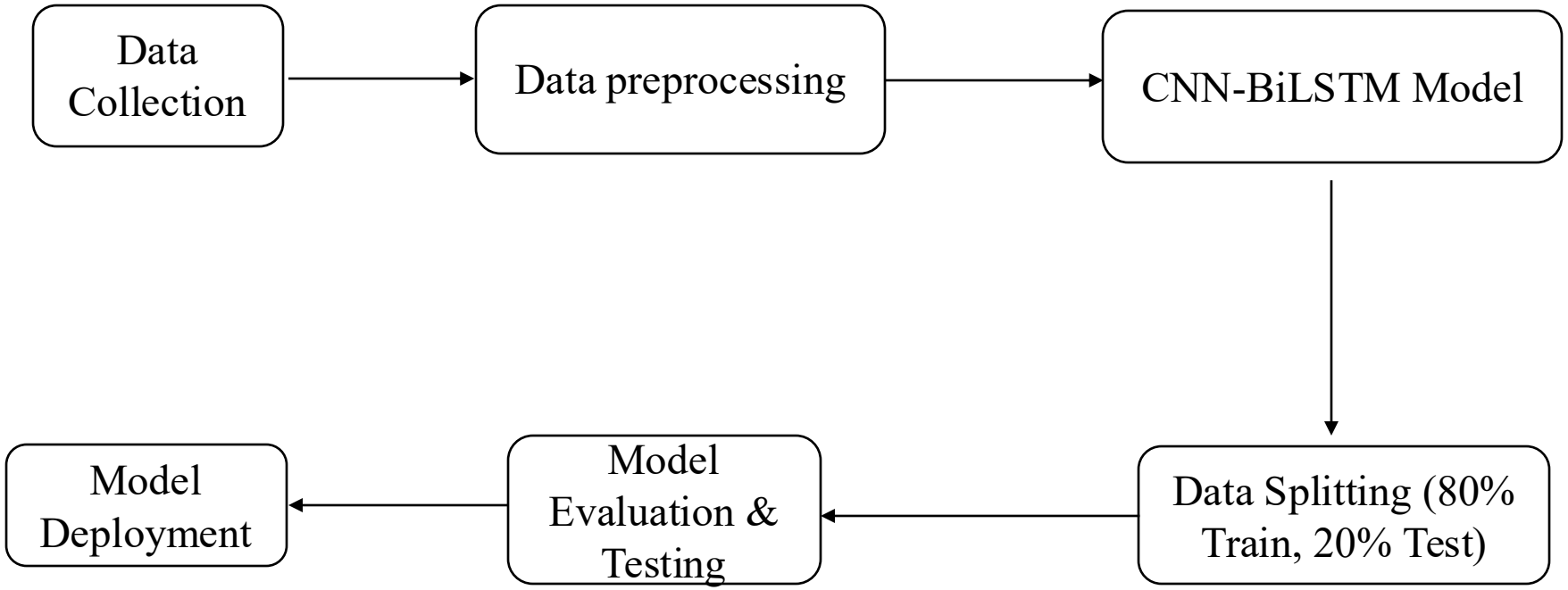


Figure 1:Proposed System for Heart Disease Prediction

EXPERIMENTAL SETUP

1. Environment & Tools

- **Libraries & Frameworks:**
  - TensorFlow/Keras for deep learning
  - NumPy, Pandas for data handling
  - scikit-learn for preprocessing and model evaluation
  - Matplotlib for visualization
  - OpenCV for image processing
- **Hardware:**
  - Intel Core i5/i7 or AMD equivalent
  - 8GB+ RAM
  - Optional: GPU (NVIDIA) for faster training

2. Dataset

- Source: heart_statlog_cleveland_hungary_final.csv
- Features: Medical attributes like age, sex, cholesterol, blood pressure, etc.
- Target: Binary classification (0 = No Heart Disease, 1 = Heart Disease)

3. Data Preprocessing

- Standardization using Standard Scaler
- One-hot encoding of the target label
- Reshaping input to fit Conv1D layer: (samples, features, 1)
- 80-20 train-test split using train_test_split

4. Model Configuration

- Architecture: CNN → BatchNorm → Dropout → CNN → Reshape → BiLSTM → Dense Layers
- Activation Functions: ReLU (hidden layers), Softmax (output)
- Optimizer: Adam (learning rate = 0.001)
- Loss Function: Categorical Crossentropy
- Epochs: 50Batch Size: 32

5. Evaluation Metrics

- Accuracy
- Loss
- Visualization: Accuracy/Loss plots for training vs. validation

CODE IMPLEMENTATION

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Conv1D, LSTM, Bidirectional, BatchNormalization, Reshape
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

# =====================
# Load and Preprocess Dataset
# =====================
# Path to the dataset
csv_path = "heart_statlog_cleveland_hungary_final.csv"

# Load CSV file
df = pd.read_csv(csv_path)
```

CODE IMPLEMENTATION(Cont…)

```
# Separate features and target
X = df.drop(columns=['target'])  # Features
y = df['target']  # Target

# Normalize feature values
scaler = StandardScaler()
X = scaler.fit_transform(X)

# One-hot encode target labels (binary classification)
y = to_categorical(y, num_classes=2)

# Reshape input data to match Conv1D input shape (samples, features, 1)
X = np.expand_dims(X, axis=2)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

CODE IMPLEMENTATION(Cont…)

```
# =====================
# Define CNN + BiLSTM Model
# =====================
model = Sequential([
    Conv1D(64, kernel_size=3, activation='relu', input_shape=(X_train.shape[1], 1)),
    BatchNormalization(),
    Dropout(0.3),

    Conv1D(128, kernel_size=3, activation='relu'),
    BatchNormalization(),
    Dropout(0.3),

    Reshape((X_train.shape[1] - 4, 128)),  # Adjusted shape after Conv1D with kernel_size=3 (twice)
    Bidirectional(LSTM(64, return_sequences=True)),
    Bidirectional(LSTM(32)),

    Dense(64, activation='relu'),
    Dropout(0.4),
    Dense(2, activation='softmax')
])
```

CODE IMPLEMENTATION(Cont…)

```
# =====================
# Compile and Train Model
# =====================
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train model
history = model.fit(X_train, y_train,
                    validation_data=(X_test, y_test),
                    epochs=50,
                    batch_size=32,
                    verbose=1)

# =====================
# Evaluate Model
# =====================
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

CODE IMPLEMENTATION(Cont…)

```
# =====================
# Plot Accuracy and Loss Graphs
# =====================
def plot_metrics(history):
    plt.figure(figsize=(12, 5))

    # Accuracy plot
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.legend()
    plt.title('Model Accuracy')

    # Loss plot
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Train Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.legend()
    plt.title('Model Loss')

    plt.tight_layout()
    plt.show()

# Plot training metrics
plot_metrics(history)
```

CODE IMPLEMENTATION(Cont…)

```
# Plot training metrics
plot_metrics(history)

# ====================
# Save Model
# ====================
model.save("/mnt/data/heart_disease_cnn_bilstm_model.h5")

# ====================
# Display Linked Model Architecture or Results Image
# ====================
# This image (DL.jpg) should be located in the same directory and may show:
# - Model Architecture Diagram
# - Confusion Matrix
# - Accuracy Graph

image_path = '/content/DL.jpg'

# Read the image using OpenCV
image = cv2.imread(image_path)

if image is None:
    print("Error: Unable to read the image. Please check the file path or file format.")
else:
    # Convert BGR (OpenCV format) to RGB (matplotlib format)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

CODE IMPLEMENTATION(Cont…)

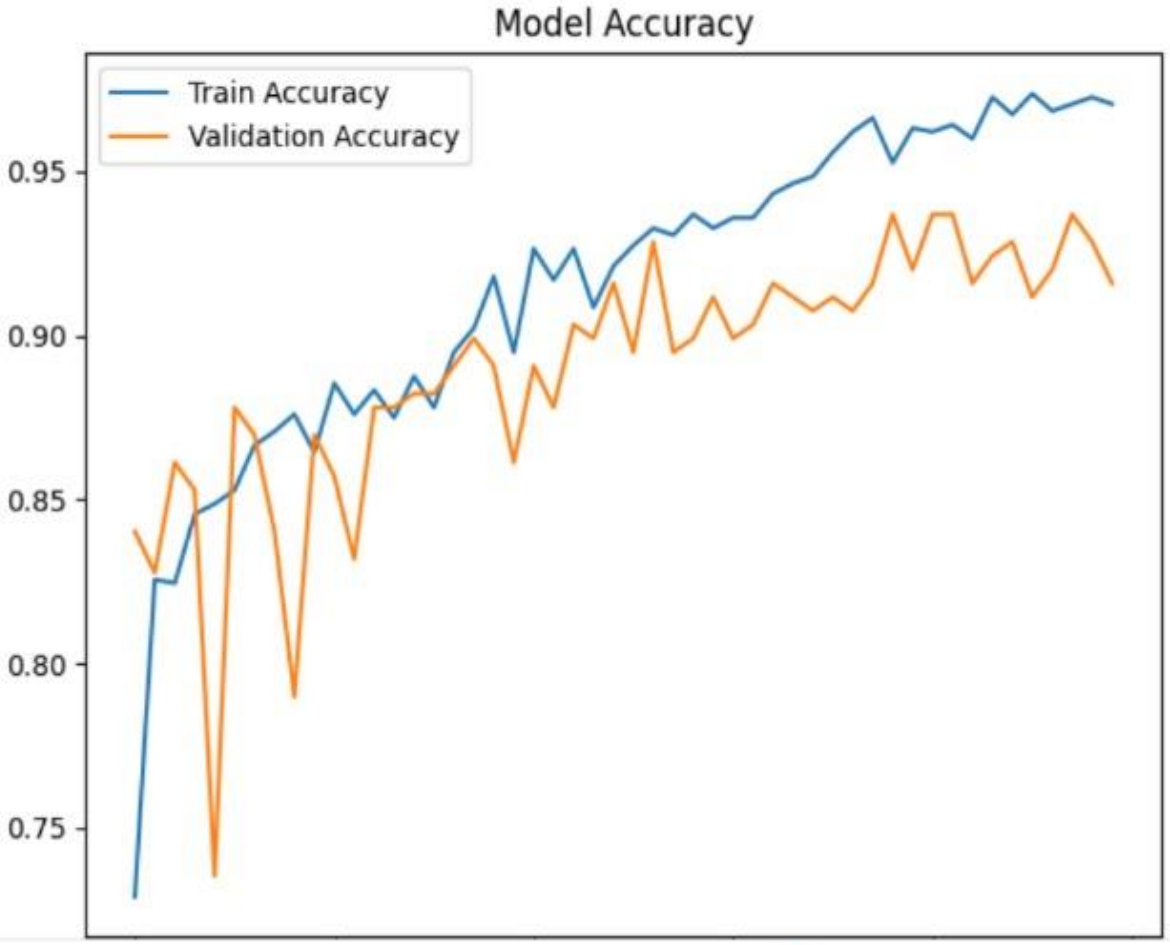
```
image_path = '/content/DL.jpg'

# Read the image using OpenCV
image = cv2.imread(image_path)

if image is None:
    print("Error: Unable to read the image. Please check the file path or file format.")
else:
    # Convert BGR (OpenCV format) to RGB (matplotlib format)
    image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Set figure size for clarity
    plt.figure(figsize=(12, 8))

    # Display image
    plt.imshow(image_rgb)
    plt.title("Figure: CNN-BiLSTM Architecture or Performance Graph")
    plt.axis('off')
    plt.show()
```

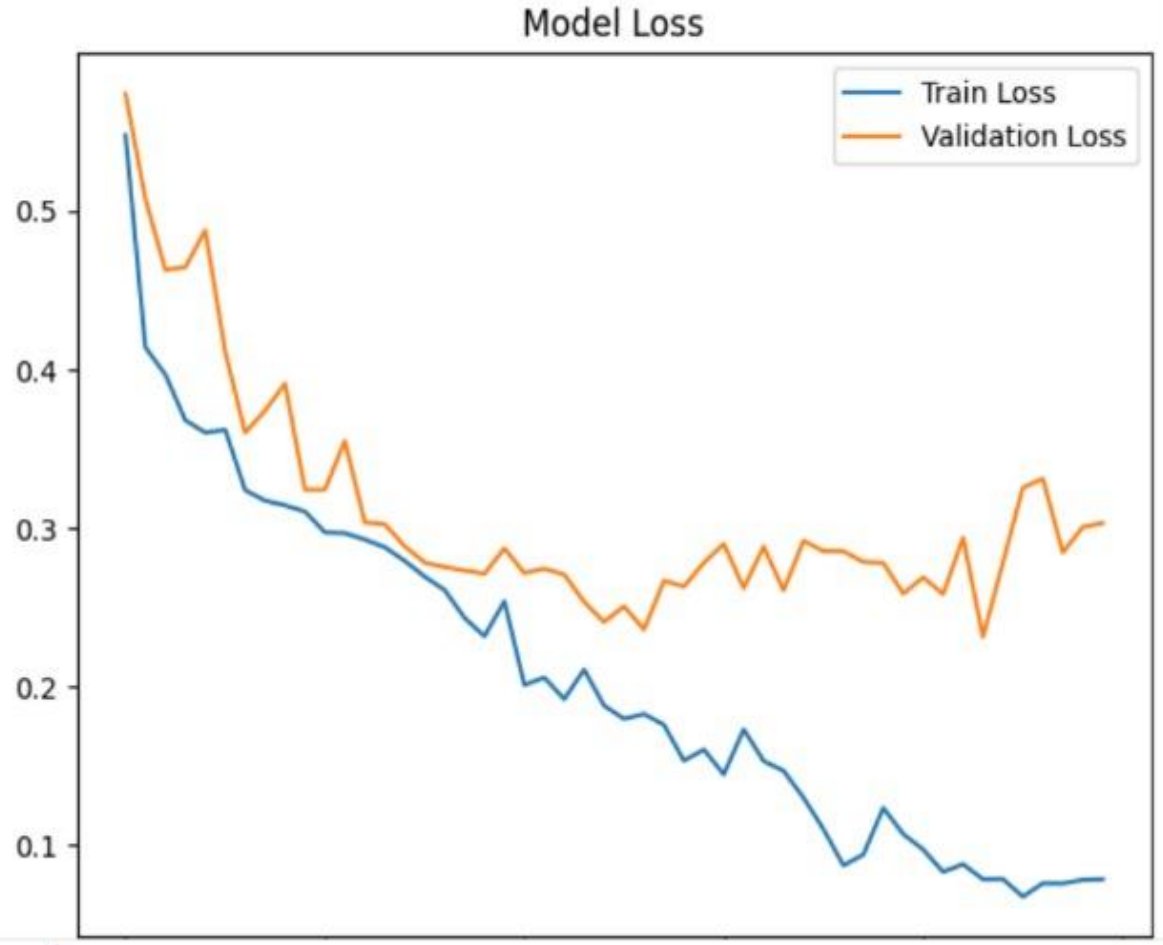
OUTPUT

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pa
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Epoch 1/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 12s 61ms/step - accuracy: 0.6897 - loss: 0.6064 - val_accuracy: 0.8403 - val_loss: 0.5739
Epoch 2/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.8224 - loss: 0.4247 - val_accuracy: 0.8277 - val_loss: 0.5076
Epoch 3/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8128 - loss: 0.4032 - val_accuracy: 0.8613 - val_loss: 0.4629
Epoch 4/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.8560 - loss: 0.3373 - val_accuracy: 0.8529 - val_loss: 0.4646
Epoch 5/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 47ms/step - accuracy: 0.8395 - loss: 0.3681 - val_accuracy: 0.7353 - val_loss: 0.4879
Epoch 6/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 2s 27ms/step - accuracy: 0.8600 - loss: 0.3494 - val_accuracy: 0.8782 - val_loss: 0.4111
Epoch 7/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.8742 - loss: 0.3057 - val_accuracy: 0.8697 - val_loss: 0.3602
Epoch 8/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.8683 - loss: 0.3171 - val_accuracy: 0.8403 - val_loss: 0.3742
Epoch 9/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.8779 - loss: 0.3039 - val_accuracy: 0.7899 - val_loss: 0.3911
Epoch 10/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.8689 - loss: 0.3175 - val_accuracy: 0.8697 - val_loss: 0.3243
Epoch 11/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8928 - loss: 0.2881 - val_accuracy: 0.8571 - val_loss: 0.3243
Epoch 12/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.8629 - loss: 0.3210 - val_accuracy: 0.8319 - val_loss: 0.3550
Epoch 13/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.8900 - loss: 0.2816 - val_accuracy: 0.8782 - val_loss: 0.3037
Epoch 14/50
```

OUTPUT(Cont…)

```
Epoch 50/50
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - accuracy: 0.9776 - loss: 0.0605 - val_accuracy: 0.9160 - val_loss: 0.3032
8/8 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.9172 - loss: 0.2538
Test Accuracy: 0.9160
```



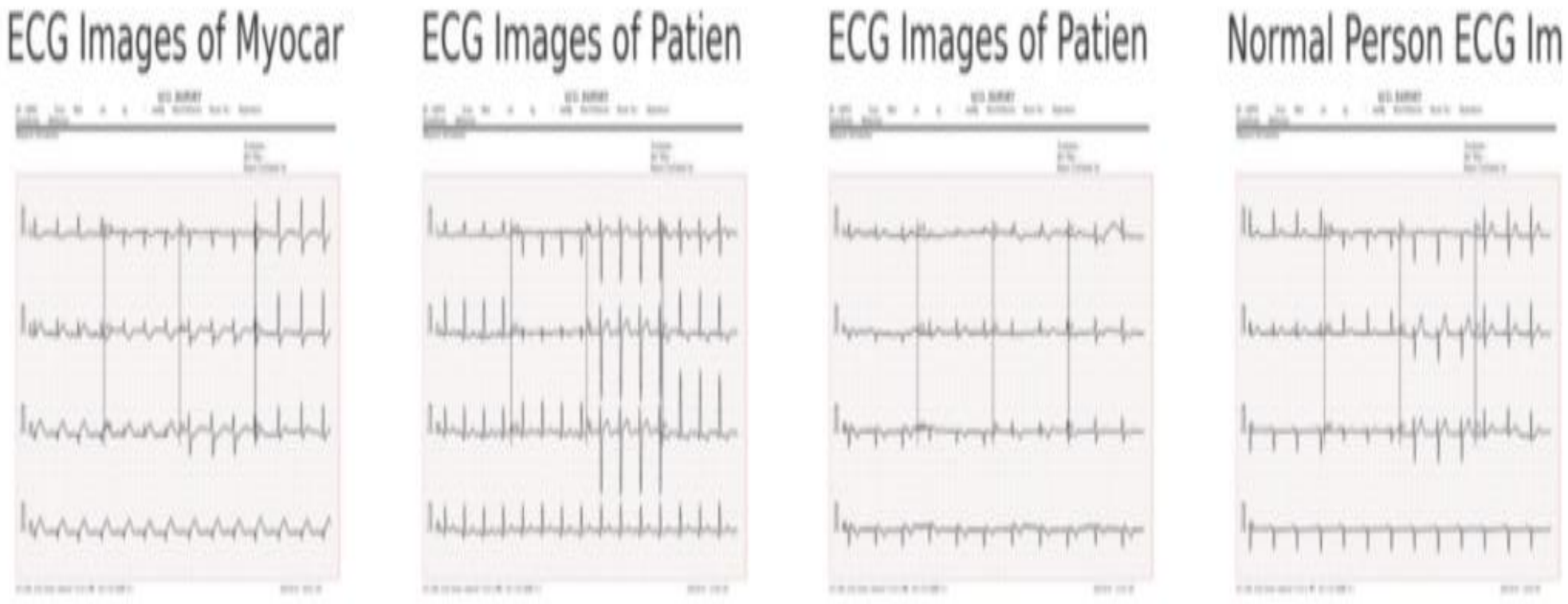


✓ 1m 13s completed at 1:25 AM

OUTPUT(Cont…)

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy
```

Figure: CNN-BiLSTM Architecture or Performance Graph



THANK YOU