Deep Learning Approaches for Heart Disease Detection Using MAX30102 Sensor Data

Pham Thanh Dat[1], Huynh Kha Tu[2] and Assoc. Prof. Le Trung Quan[3]

University of Information Technology, Viet Nam National University Ho Chi Minh City
20521175@gm.uit.edu.vn[1], 20522096@gm.uit.edu[2] and qualt@uit.edu.vn[3]

Abstract. Deep learning has recently garnered significant attention due to its potential to create fast, automated, and highly accurate systems for image recognition and classification. This study explores the application of deep learning through a Multilayer Perceptron model for the classification of heart rate data, aiming to identify abnormal patterns indicative of potential heart diseases. Utilizing a comprehensive dataset on heart rates and cardiovascular conditions, the model demonstrates the capacity for automated and precise analysis. Additionally, the integration of mobile technology is highlighted, with the development of a compact device incorporating the MAX30102 sensor. This device, capable of real-time heart rate monitoring, interfaces directly with a mobile app to provide immediate health status feedback to users. This research underscores the potential of combining deep learning and mobile technology for enhanced cardiovascular disease diagnosis and monitoring.

Keywords: Deep Learning, Mobile App, Multilayer Perceptron, Classification, MAX30102, ESP 8266.

1 Introduction

Heart disease continues to pose one of the most significant global health challenges, leading to high mortality rates worldwide [1]. Early detection and accurate diagnosis are crucial in treating and preventing cardiovascular conditions, underscoring the importance of heart rate as a primary indicator of cardiovascular health.

Abnormal or sudden changes in heart rate may signal underlying cardiac issues, emphasizing the importance of heart rate monitoring for early heart disease detection. This study aims to develop a system capable of swiftly classifying individuals at risk of heart disease based on their heart rate data.

Utilizing the proven effectiveness of deep learning across various fields, including healthcare, we constructed an efficient Multilayer Perceptron model. This model leverages a large dataset on heart rates and cardiovascular health conditions, enabling it to identify patterns indicative of heart disease.

Furthermore, this research introduces a mobile application integrated with a compact device utilizing the MAX30102 sensor. This device allows for continuous heart rate monitoring, transmitting data directly to the mobile app. The mobile app then employs

the deep learning model to classify the risk of heart disease based on heart rate data, providing immediate health status feedback to users.

The innovative combination of deep learning and mobile technology offers a promising tool for improved diagnosis and monitoring of cardiovascular diseases. It enables patients to check and monitor their condition anytime and anywhere, and physicians can remotely observe the status of their patients at once. This reduces physicians' workload and facilitates early detection of abnormal signs, increasing the chances of successful treatment.

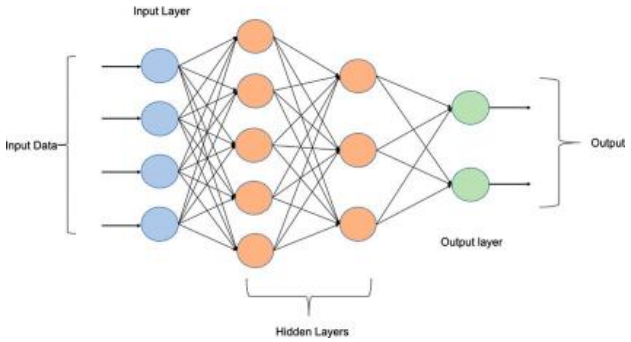

Fig 1. A Multi-layer perceptron (MLP-NN) basic architecture [2]

2 Proposed solution

In this section, we present the approaches we propose to implement this project. The ESP8266 device will connect to Wi-Fi and then receive heart rate information from the MAX30102 sensor, process the data, and upload it to the Google Firebase Realtime Database. Subsequently, an Android application will retrieve the data from the database, then utilize an integrated Machine Learning Model within the app to perform calculations for predicting the user's cardiovascular health condition.

We have divided our implementation into the following key stages:

(1) **Google Firebase Realtime Database:** We establish a Firebase Realtime Database, employing email authentication and combining it with the Database API for ESP8266 usage, aimed at accessing and writing data into the Database.

(2) **ESP8266:** Necessary libraries are installed, utilizing code snippets and libraries to calculate the heart rate from signals received from the sensor. This is followed by configuring the ESP8266 to connect to Wi-Fi. The subsequent step involves configuring the device to connect to the Database, enabling it to write data into the database.

(3) **Building Deep Learning Model:** We search for reputable and suitable datasets, develop models optimized for low-resource devices, and employ techniques to reduce overfitting and increase accuracy.

(4) **Building Android Application:** Development of a mobile application is undertaken, incorporating authentication methods to differentiate between

patients and doctors. Configuration for reading and retrieving data from Firebase Database is established, along with setting up and deploying the model using TensorFlow into the application.

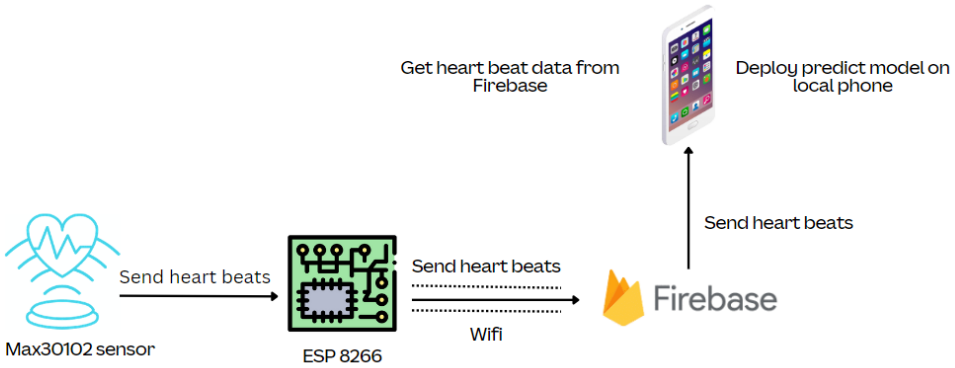

Figure 2. Data flow of this project

3 Practical application

3.1 Establishing Google Firebase Realtime Database

To initialize Firebase Realtime Database, configure the security rules to allow data access [3]. By default, Firebase requires user authentication to access data. For development purposes, you can temporarily set it to "true" to allow read/write access without authentication. The Firebase API can be utilized to read and write data to the Realtime Database:

Writing Data: Use firebase.database().ref('path/to/data').set({yourData}) to store data.

Reading Data: Use firebase.database().ref('path/to/data').on('value', snapshot => {console.log(snapshot.val())}) to listen and read data.

Configure the ESP8266 with Firebase connection information:

Firebase Host: The URL of the Realtime Database (excluding the 'https://' and the trailing '/').

Firebase Auth: The secret token for authentication.

This setup enables the ESP8266 to interact seamlessly with Firebase, facilitating real-time data communication for IoT applications.

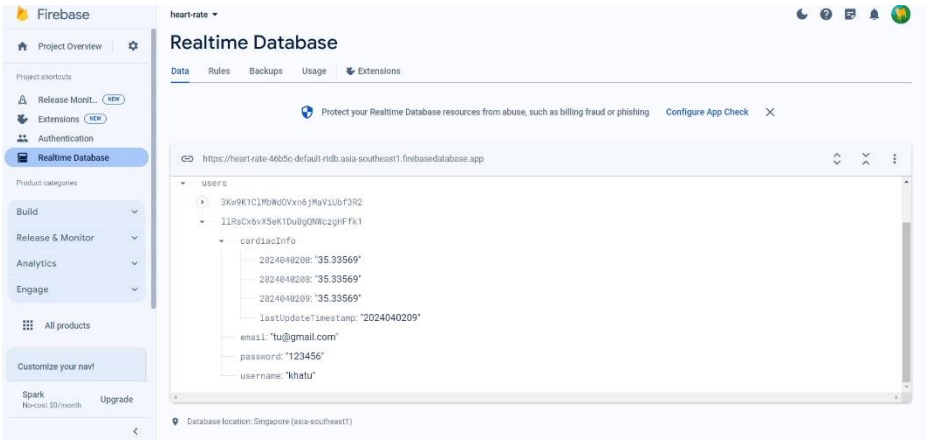

Figure 3. Configure Firebase Database

3.2 Establishing ESP8266 with MAX30102 Sensor

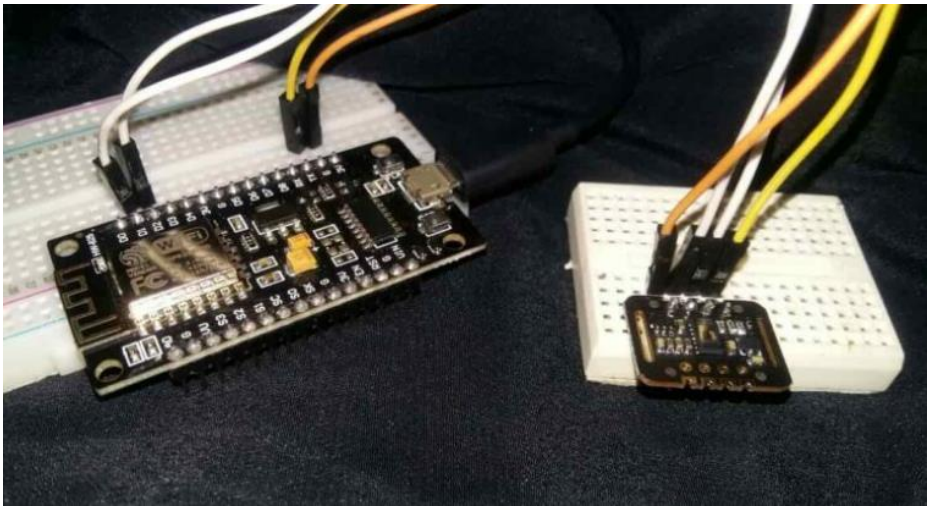

Figure 4. Image for building device. [4]

The connections between the sensor module and ESP8266 are as follows:

MAX30102 Module	ESP8266
VCC	3.3V
SCL	D1
SDA	D2
GND	GND

Link the standard I2C pins on the ESP8266 to the SCL and SDA pins on the module. Also, the sensor uses 3.3V from the ESP8266 for power, and they both share the same ground connection.

3.3 Build Deep Learning Model

3.3.1 Data Preprocessing

Data Loading: The data is loaded from a CSV file named 'heart_statlog_cleveland_hungary_final.csv'. This dataset is expected to contain features relevant to heart disease diagnosis and a target column indicating the presence or absence of the condition.

Heart Disease Dataset Attribute Description

S.No.	Attribute	Code given	Unit	Data type
1	age	Age	in years	Numeric
2	sex	Sex	1, 0	Binary
3	chest pain type	chest pain type	1,2,3,4	Nominal
4	resting blood pressure	resting bp s	in mm Hg	Numeric
5	serum cholesterol	cholesterol	in mg/dl	Numeric
6	fasting blood sugar	fasting blood sugar	1,0 > 120 mg/dl	Binary
7	resting electrocardiogram results	resting ecg	0,1,2	Nominal
8	maximum heart rate achieved	max heart rate	71–202	Numeric
9	exercise induced angina	exercise angina	0,1	Binary
10	oldpeak =ST	oldpeak	depression	Numeric
11	the slope of the peak exercise ST segment	ST slope	0,1,2	Nominal
12	class	target	0,1	Binary

Figure 5. Dataset Description [5]

Feature Selection: All columns except 'target' are used as features (X), and 'target' is used as the label (y).

Splitting Data: The dataset is split into training and testing sets, with 80% of the data used for training and 20% for testing. This is a common ratio that allows for both effective learning and a meaningful evaluation of the model's performance.

Data Scaling: The StandardScaler is applied to normalize the features, ensuring that the model isn't unfairly influenced by the scale of any feature.

Building the Basic Model

3.3.2 Model Architecture:

A Sequential model is constructed with Dense layers, each followed by BatchNormalization and Dropout. This architecture is aimed at reducing overfitting (through Dropout) and speeding up training (via BatchNormalization), with a final sigmoid activation function for binary classification.

Compilation: The model is compiled with the Adam optimizer, binary cross-entropy loss function, and accuracy as the metric, which are suitable choices for binary classification tasks.

3.3.3 Fine-tuning and Adjusting Hyperparameters:

Callbacks: The use of ModelCheckpoint and EarlyStopping callbacks is a form of hyperparameter tuning. ModelCheckpoint saves the best model based on validation accuracy, ensuring that the model's best state is retained. EarlyStopping prevents overfitting by halting training if the validation loss doesn't improve for a specified number of epochs.

Training Parameters: The model is trained with a validation split of 20%, a batch size of 32, and for up to 120 epochs, though training may stop early due to early stopping. These choices affect the model's learning process and can be adjusted for better performance.

3.3.4 Model Evaluation:

Evaluation on Test Set: The model's final accuracy and loss on the test set are reported, providing insight into its generalization ability.

Prediction and Comparison: Predictions are made on the test set, and a DataFrame comparing actual vs. predicted values is displayed, along with a 'Check' column to easily see correct predictions. This is helpful for understanding the model's performance at an individual level.

Conversion to TFLite: Finally, the best model is converted to TensorFlow Lite format, making it suitable for deployment in mobile or edge devices. This step indicates planning for practical application.

3.4 Building Android Application

We made a mobile app that lets doctors keep an eye on their patients' heart health, and lets patients check on themselves too. In Android Studio, we added some extra pieces called dependencies for Firebase and TensorFlow in a file named build.gradle. These include things like Firebase Auth and Database, and also TensorFlow Lite. We also used something called Firebase Authentication to make sure users are who they say they are.

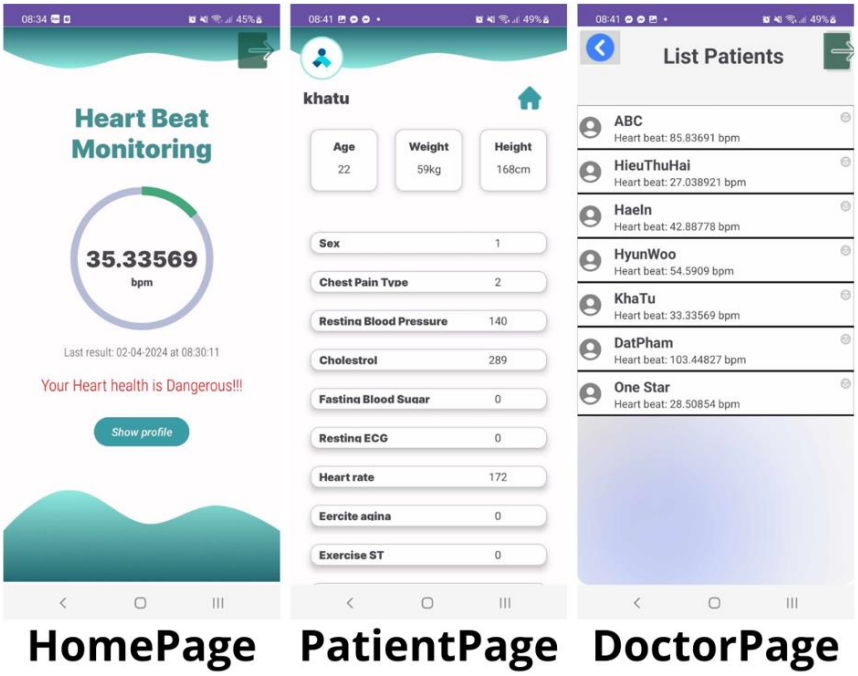

Figure 6. User Interface of our Application

4 Result and Future work

During our experiments, we achieved some impressive results with the Multilayer Perceptron model. Specifically, we hit an accuracy rate of 88.66% and an error rate of 31.26%.

```
8/8 [==============================] - 0s 4ms/step
 Real Values  Predicted Values  Check
           1                 1   True
           1                 1   True
           0                 0   True
           1                 1   True
           1                 1   True
           1                 1   True
           0                 0   True
           0                 0   True
           0                 0   True
           0                 0   True
           0                 0   True
           1                 0  False
           0                 0   True
           0                 0   True
           1                 1   True
```

Figure 7. Results of predict values.

This shows that our model works well in identifying and predicting heart disease, and it also shows how tech can be used in healthcare. After we put the model into mobile devices, we were able to send and receive data smoothly. This means doctors and patients can now keep an eye on heart health from anywhere.

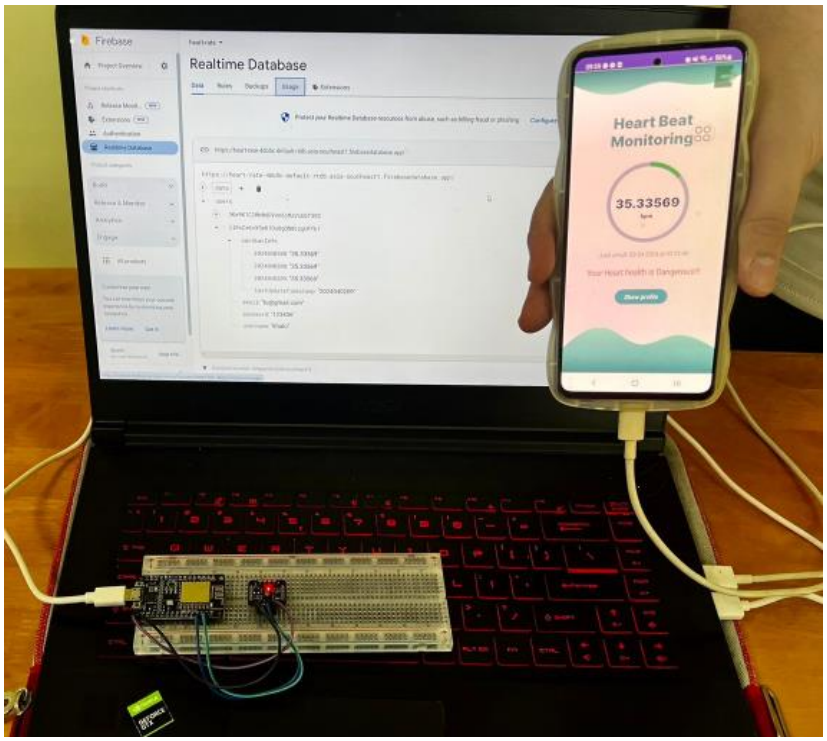

Figure 8. Full connected IoTs Device in Real-World

In the future, we want to build a broader healthcare system that goes beyond just monitoring heart health. We want this system to let doctors look after and take care of their patients from afar, and also let patients take control of their own health. We're also looking into making better devices for measuring blood oxygen levels. By doing this, we hope to gather more types of data, which can make our model even better. This could lead to discovering new ways to use tech in healthcare, which could improve how we look after health in our communities.

References

1. World Health Organization. (2021, June 11). Cardiovascular diseases (CVDs). https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)
2. Afan, A., Osman, A. A. H., Ahmedbahaaaldin, E., Yusuf, E., Ahmed, A. N., Huang, Y., Kisi, O., Sherif, M., Sefelnasr, A., Chau, K., & El-Shafie, A. (2021). Modeling the fluctuations of groundwater level by employing ensemble deep learning techniques. *Engineering Applications of Computational Fluid Mechanics*, *15*, 1420-1439. https://doi.org/10.1080/19942060.2021.1974093
3. ESP8266 Data Logging to Firebase Realtime Database. *Random Nerd Tutorials*. https://randomnerdtutorials.com/esp8266-data-logging-firebase-realtime-database/

4. MAX30102 Máy đo oxy xung và cảm biến nhịp tim với ESP8266. Mecsu. https://mecsu.vn/ho-tro-ky-thuat/max30102-may-do-oxy-xung-va-cam-bien-nhip-tim-voi-esp8266.pjg
5. Janosi,Andras, Steinbrunn,William, Pfisterer,Matthias, and Detrano,Robert. (1988). Heart Disease. UCI Machine Learning Repository. https://doi.org/10.24432/C52P4X.