ECG- Image based Heartbeat classification for Arrhythmia Detection Using IBM Watson Studio

INTRODUCTION

1.1 Overview

An electrocardiogram (ECG) is a complete representation of the electrical activity of the heart on the surface of the human body, and it is extensively applied in the clinical diagnosis of heart diseases, it can be reliably used as a measure to monitor the functionality of the cardiovascular system. ECG signals have been widely used for detecting heart diseases due to its simplicity and non-invasive nature. Features of ECG signals can be computed from ECG samples and extracted using some softwares.

Arrhythmia is a representative type of Cardio Vascular Disease that refers to any irregular change from the normal heart rhythms. There are several types of arrhythmia including atrial fibrillation, premature contraction, ventricular fibrillation, and tachycardia. Although a single arrhythmia heartbeat may not have a serious impact on life, continuous arrhythmia beats can result in fatal circumstances.

1.2 Purpose

To detect ECG image based Heartbeat Classification for classifing several types of Arrhythmia using Convolution Neural Networks(CNN). The target of the classification process is to obtain an intelligent model, that is capable to class any heartbeat signal to specific type of Arrhythmia clasification. The final results shows that the project is more efficient in terms of accuracy and also competitive in terms of sensitivity and specificity.

Literature Survey

2.1 Existing Problem

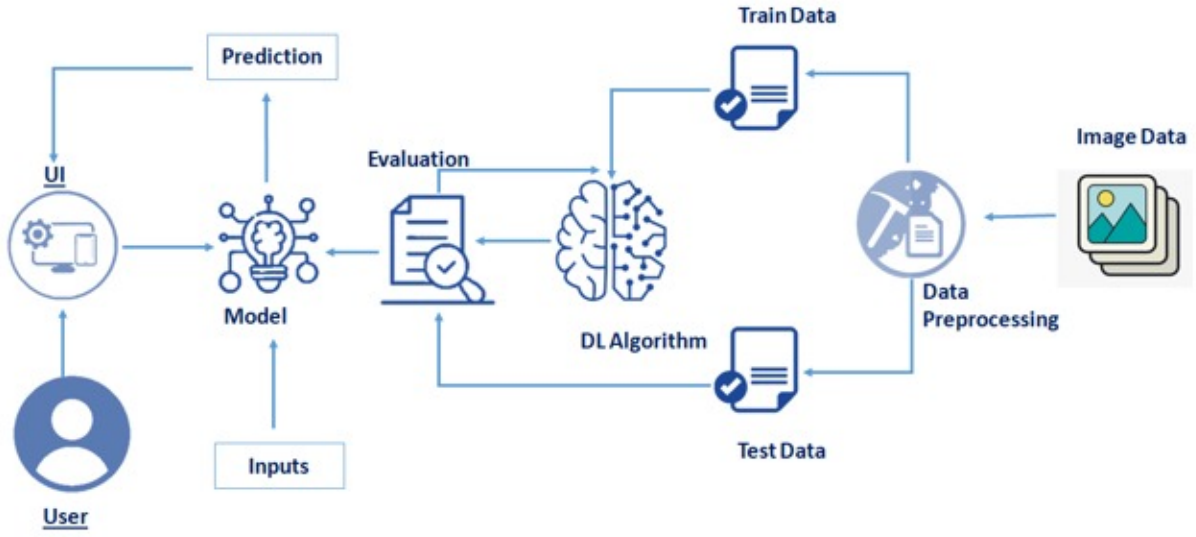
Cardiologists use mostly the raw ECG to diagnose. The simplest and fastest method of feature extraction is then to extract sampled points from an ECG signal curve. However, one should be aware of the fact that the amount of the extracted features used to characterize the heartbeat can be a burden for the classification algorithm. For this reason, most of the works that use the raw signal perform a down sampling of the waveform or some feature selection in order to reduce the computation time. In order to circumvent this issue, a simple machine learning method is chosen to classify the arrhythmias.

2.2 Proposed Solution

In this project, we built an effective electrocardiogram (ECG) arrhythmia classification method using a convolutional neural network (CNN), in which we classify ECG into seven categories, one being normal and the other six being different types of arrhythmia using deep two-dimensional CNN with grayscale ECG images. We have created a web application where the user selects the image which is to be classified. The image is fed into the model that is trained and the cited class will be displayed on the webpage. This simple machine learning method also allows a fast retraining of the classifier if new ECG data become available.

Theoritical Analysis

3.1 Block Diagram



3.2 Hardware/Software Designing

The Softwares that is required for the project are Anaconda Navigator, Jupyter Notebook, Spyder and Notepad. The Skills that are Required for the project are Python, Python Web Frame Works, ANN, CNN, Tensorflow, Keras, Open CV. One should have prior knowledge on Supervised and unsupervised learning, Regression Classification and Clustering, Artificial Neural Networks, Convolution Neural Networks and Flask. To build a Deep learning models it required the following packages Tensorflow, Keras, Open CV and Flask.

Experimental Investigations

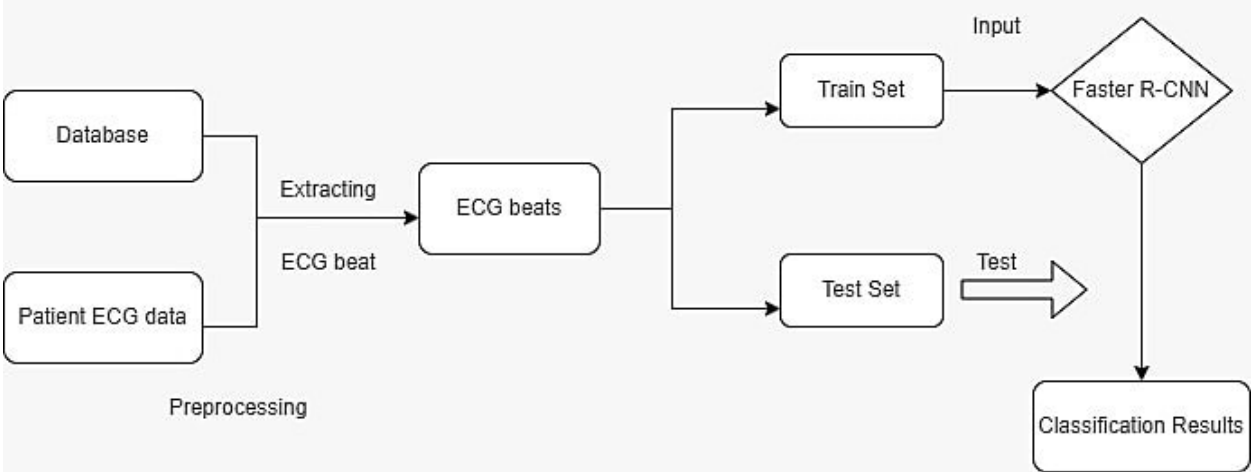
The challenge in ECG classification is to handle the irregularities in the ECG signals which is very important to detect the patient status. Each heartbeat is a combination of action impulse waveforms produced by different specialized cardiac heart tissues. Heartbeats classification faces some difficulties because these waveforms differ from person to another, they are described by some features. These features are the inputs of machine learning algorithm.

The project is designed in a way that first the User interacts with User interface to upload image, and then the Uploaded image is analyzed by the model which is integrated and finally Once the model analyses the uploaded image, the prediction is showcased on the UI.

To accomplish this, we have completed all the activities and tasks listed below:

- Data Collection.
    - Collect the dataset or Create the dataset
- Data Preprocessing.
    - Import the ImageDataGenerator library
    - Configure ImageDataGenerator class
    - Apply ImageDataGenerator functionality to Trainset and Testset
- Model Building
    - Import the model building Libraries
    - Initializing the model
    - Adding Input Layer
    - Adding Hidden Layer
    - Adding Output Layer
    - Configure the Learning Process
    - Training and testing the model
    - Optimize the Model
    - Save the Model
- Application Building
    - Create an HTML file
    - Build Python Code

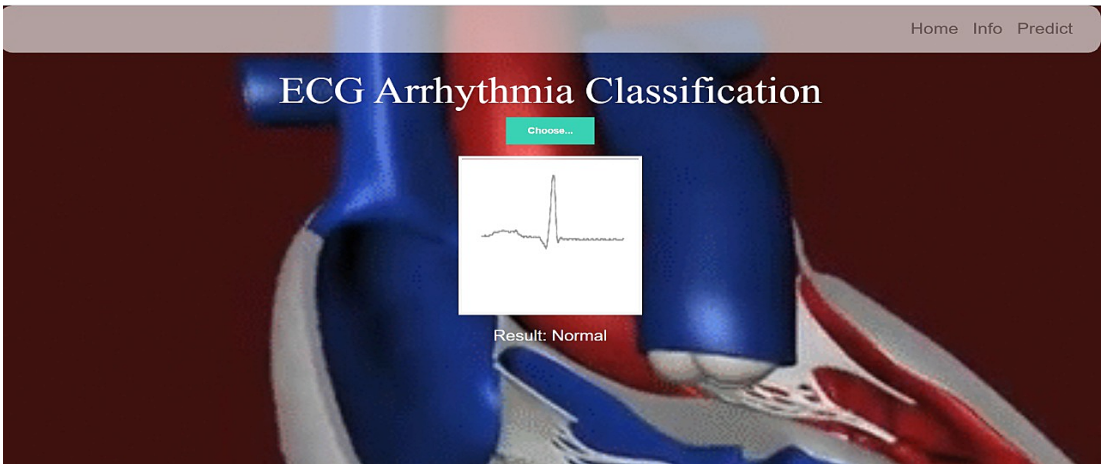
Flowchart



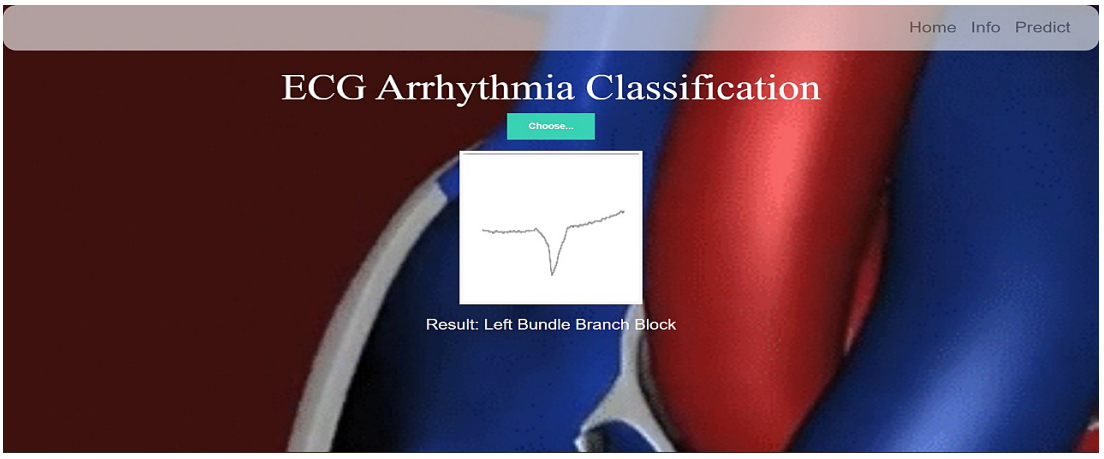
Result

From the project, we obtained the following output based on the classification.

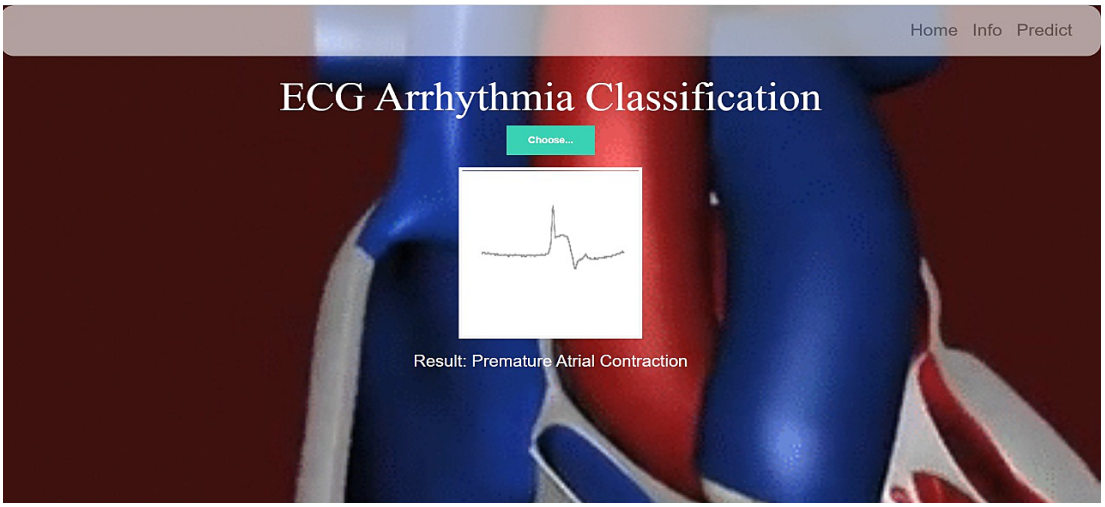
Normal



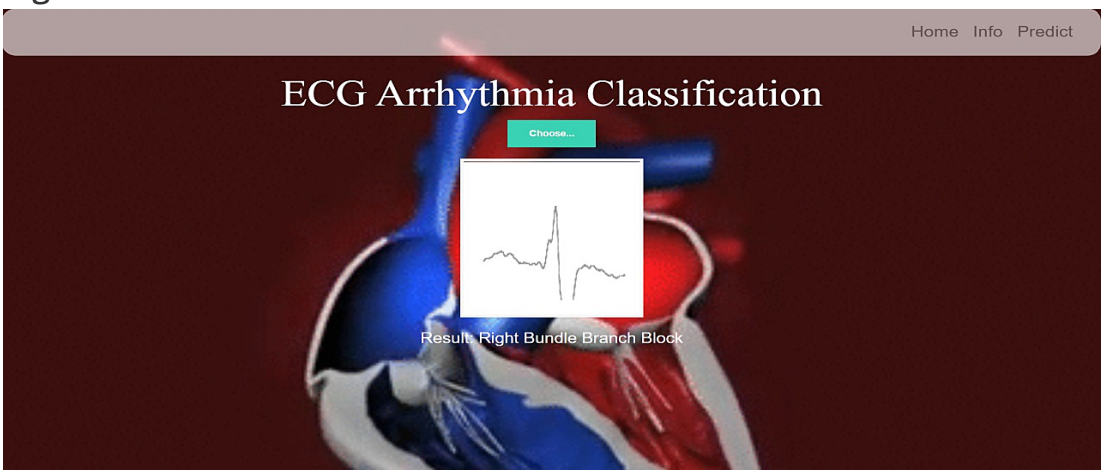
Left Bundel Branch Block



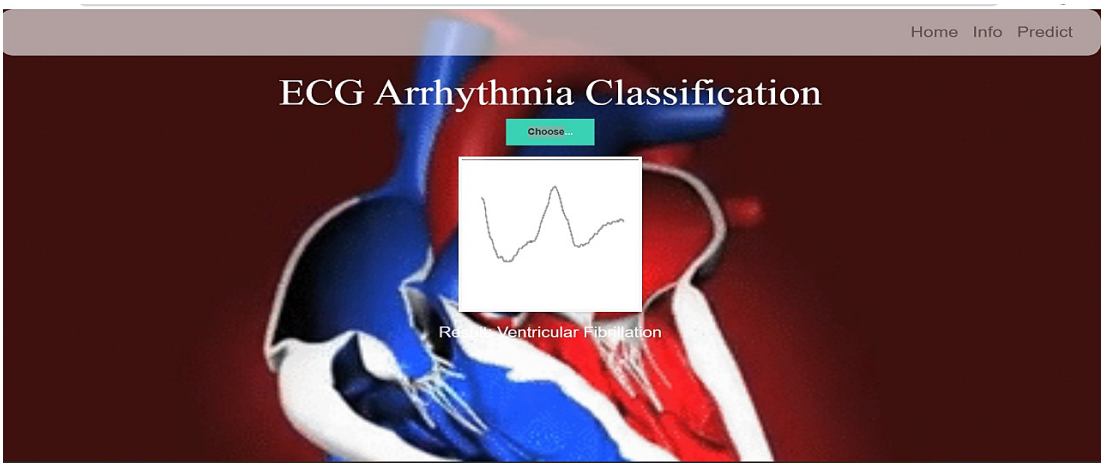
Premature Atrial Contraction



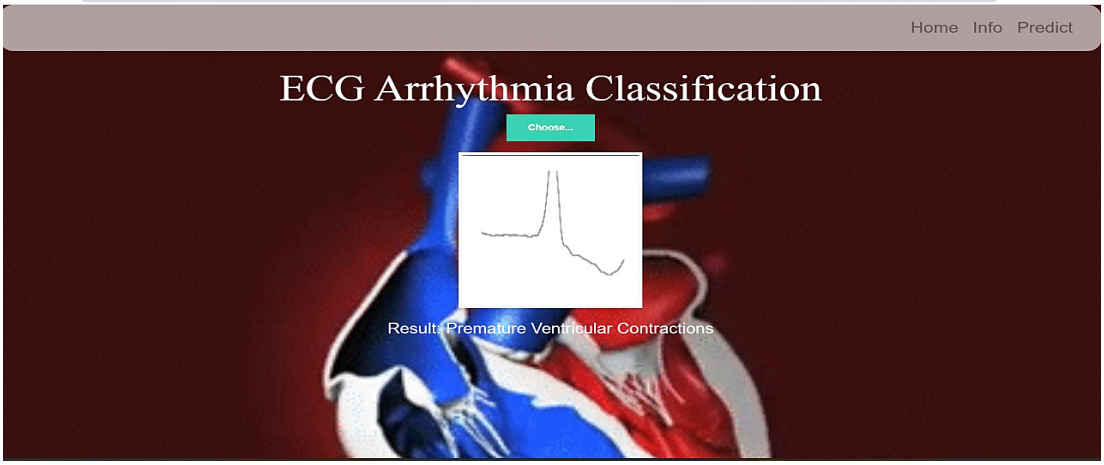
Right Bundle Branch Block



Ventricular Fibrillation



Premature Venticular Contraction



Advantages and Disadvantages

The advantage of this project is that the initiative for the early detection of diseases is a famous study and classification. The issues of biometric authentication and the application of emotional recognition can be resolved by various techniques, unlike heartbeat type detection.

The imbalance of the ECG dataset is a significant issue because certain classes have a lot of data relative to others, which might lead to false information about model performance.This is consider to be the disadavantage of the project

Applications

After the model is built, we will be integrating it into a web application so that normal users can also use it. The users need to give the image of the heartbeat to know if they arrhythmia or not.

Conclusion

In summary, This Project has validated an ability to classify heartbeats. Classification process is using some features of heartbeats and machine learning classification algorithms with local host pc working using only one node, which are crucial for diagnosis of cardiac arrhythmia.

Future Scope

The proposed model has the potential to be introduced into clinical settings as a helpful tool to aid the cardiologists in the reading of ECG heartbeat signals and to understand more about them.

Source Code

Flask Code

```
import os
import numpy as np
from flask import Flask,request,render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

app=Flask(__name__)
model=load_model('ECG.h5')

@app.route("/")
def about():
    return render_template("about.html")

@app.route("/about")
def home():
    return render_template("about.html")

@app.route("/info")
def information():
    return render_template("info.html")

@app.route("/upload")
def test():
    return render_template("upload.html")

@app.route("/predict",methods=["GET","POST"])
@app.route("/predict",methods=["GET","POST"])
def upload():
    if request.method=='POST':
        f=request.files['file']
        basepath=os.path.dirname('_file_')
        filepath=os.path.join(basepath,"uploads",f.filename)
        f.save(filepath)

        img=image.load_img(filepath,target_size=(64,64))
```

```
        x=image.img_to_array(img)
        x=np.expand_dims(x,axis=0)

        preds=model.predict(x)
        pred=np.argmax(preds,axis=1)
        print("prediction",pred)

        index=['Left Bundle Branch Block','Normal','Premature Atrial Contraction',
             'Premature Ventricular Contractions', 'Right Bundle Branch
Block','Ventricular Fibrillation']
        result=str(index[pred[0]])
        return result
    return None

if __name__=="__main__":
    app.run(debug=False)
```

Model Code

```
model=Sequential()
# adding model layer
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
#convolutional layer
model.add(MaxPooling2D(pool_size=(2,2)))
#MaxPooling2D-for downsampling the input

model.add(Conv2D(32,(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())#flatten the dimension of the image
model.add(Dense(32))#deeply connected neural network layers.
model.add(Dense(6,activation='softmax'))#output layer with 6 neurons

model.summary()#summary of our model
```