**ECG Image Analysis For Arrhythmia Classification Using IBM Watson Studio**

An Internship Project Report

                                                     Submitted By

                             (BATCH NO: CSE\_AIML\_C07)

                              P.NAGA DEEPIKA(18481A05H2)

                              M.SAI LAKSHMI (18481A05E5)

                                   P.VINAY KUMAR (18481A05H4)

             P.HEMANTH (18481A05G9)

**INDEX**

**TITLE PAGE NO**

**CHAPTER 1: INTRODUCTION 3-4**

* 1. OVERVIEW 3-4
  2. PURPOSE 4

**CHAPTER 2: LITERATURE SURVEY 5**

* 1. EXISTING PROBLEM 5
  2. PROPOSED SOLUTION 5

**CHAPTER 3: THEORTICAL ANALYSIS 6-8**

3.1 BLOCK DIAGRAM 6

3.2 HARDWARE/SOFTWARE DESIGNING 7-8

**CHAPTER 4: EXPERIMENTAL INVESTIGATIONS 9-11**

**CHAPTER 5: FLOW CHART 12**

**CHAPTER 6: RESULTS 13-14**

**CHAPTER 7: ADVANTAGES AND DISADVANTAGES 14-15**

**CHAPTER 8: APLLICATIONS 15**

**CHAPTER 9: CONCLUSION 16**

**CHAPTER 10: FUTURE SCOPE 16-17**

**CHAPTER 11: BIBILOGRAPHY 17**

**CHAPTER12: APPENDIX 17-18**

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview:**

According to the World Health Organization (WHO), cardiovascular diseases (CVDs) are the number one cause of death today. Over 17.7 million people died from CVDs in the year 2017 all over the world which is about 31% of all deaths, and over 75% of these deaths occur in low and middle-income countries. Arrhythmia is a representative type of CVD 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. In this project, we build 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 are creating 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.

To accomplish this, we have to complete 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

**1.2 Purpose:**

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 (ex: Matlab). For instance, millions of people suffer from irregular heartbeats which can be lethal in some cases. Therefore, accurate and low-cost diagnosis of arrhythmic heartbeats is highly desirable . **CHAPTER 2**

**LITERATURE SURVEY**

* 1. **Existing Problem:**

Cardiovascular diseases (CVDs) are the leading cause of death today. The current identification method of the diseases is analyzing the Electrocardiogram (ECG), which is a medical monitoring technology recording cardiac activity. Unfortunately, looking for experts to analyze a large amount of ECG data consumes too many medical resources. Therefore, the method of identifying ECG characteristics based on machine learning has gradually become prevalent. However, there are some drawbacks to these typical methods, requiring manual feature recognition, complex models, and long training time.

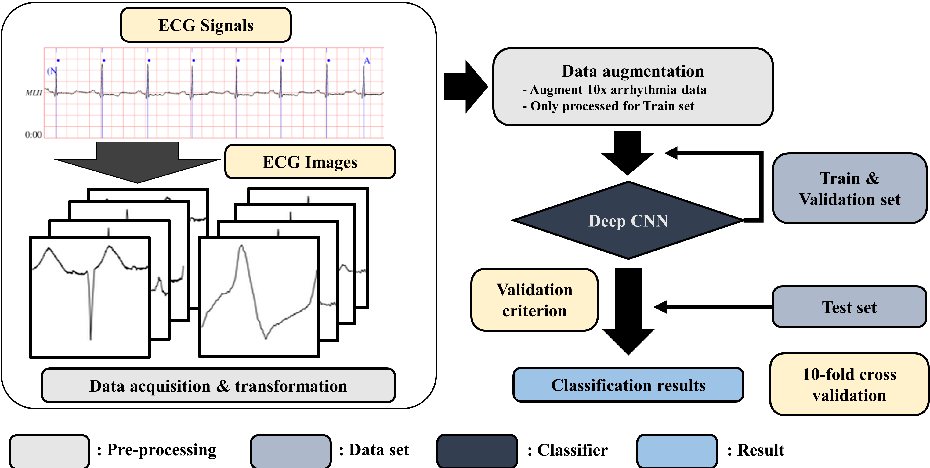
* 1. **Proposed Solution:**

Most of the research in this ﬁeld has been conducted by incorporating different approaches of machine learning (ML) techniques for the efﬁcient identiﬁcation and accurate examination of ECG signals . ECG signal classiﬁcation based on different approaches has been presented in the literature including frequency analysis , artiﬁcial neural networks (ANNs) , heuristic-based methods , statistical methods , support vector machines (SVMs) , wavelet transform , ﬁlter banks , hidden Markov models , and mixture-of-expert methods . An artiﬁcial neural network based method obtained an average accuracy of 90.6% for the classiﬁcation of ECG wave into six classes . Meanwhile, a feed-forward neural network was used as a classiﬁer for the detection of four types of arrhythmia classes and achieved an average accuracy of 96.95.

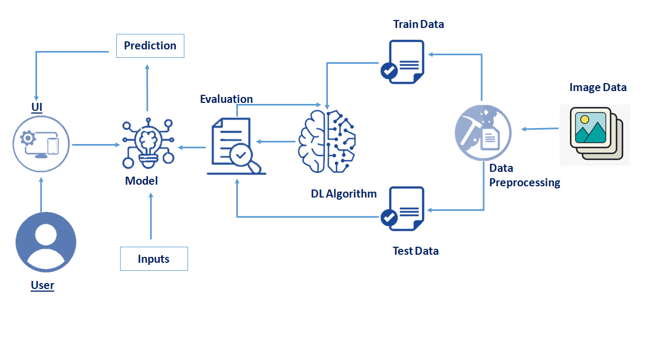
**CHAPTER 3**

**THEORTICAL ANALYSIS**

**3.1 Block Diagram:**

****

**Technical Architecture:**

****

**3.2 Hardware/software designing:**

**Software specifications:**

|  |  |
| --- | --- |
| **REQUIREMENT** | **SPECIFICATIONS** |
| Anaconda Navigator | Anaconda Navigator is a free and open-source distribution of the Python and R programming languages for data science and machine learning related applications. It can be installed on Windows, Linux, and macOS. |
| Python | The version required to run this project is 3.7 0r 3.8 to install libraries. |
| Numpy | The version required is 1.19.5 |
| Pandas | pandas is a software library written for the Python programming language for data manipulation and analysis.The version required is 1.3.0. |
| tensorflow | provides a collection of workflows to develop and train models usingPython.The version required is 2.5.0. |
| Flask | The version is required is 2.0.1 |
| Keras | The version is required is 2.5.0 |

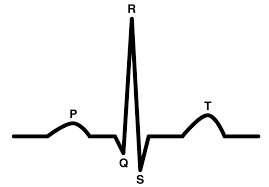
**Hardware Specifications:**

|  |  |
| --- | --- |
| **REQUIREMENT** | **SPECIFICATIONS** |
| Operating system | Microsoft Windows  UNIX  Linux® |
| Processing | Minimum: 4 CPU cores for one user. For each deployment, a sizing exercise is highly recommended. |
| RAM | Minimum 10 GB. |
| Operating system specifications | File descriptor limit set to 8192 on UNIX and Linux |
| Disk space | A minimum of 7 GB of free space is required to install the software and 5 GB of free space on the drive that contains the temporary directory used by IBM . |

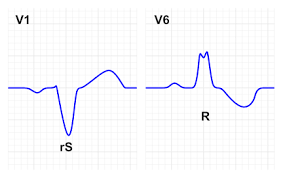
**CHAPTER 4**

**EXPERIMENTAL INVESTIGATIONS**

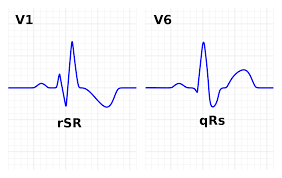
This 2D Convolutional Neural Network Classifier is 8 layers Neural network. As seen in Fig. 3, at the first layer, there are Convo2D with 32 filters and kernel size 3x3 and then 64 filters with 3x3 kernel size on the next layer. Next is to use max-pooling to pool the best feature. Afterward, the output randomly dropout data with a rate of 0.25 to remove inconsistency data. On the next layer, there is flatten to preparing data to be fully connected to the next layer. We do dropout again with rate 0.5 and then on the final layer, we are using Softmax activation to convert the matrix into probability.



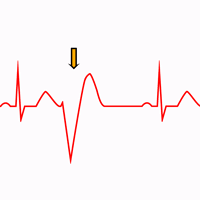
* Normal

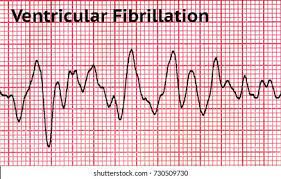


-Left Bundle Branch Block

-Right Bundle Branch Block

 -Premature Atrial Contraction

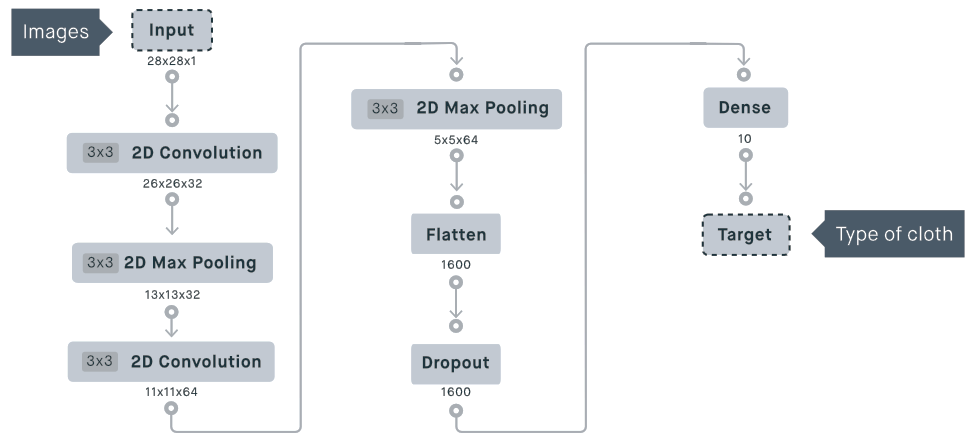
-Premature Ventricular Contraction

-Ventricular Fibrillation

Using Transformed ECG signal image we convert it to several sizes including 64x64, 32x32 and 16x16 as illustrated. 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.

**CHAPTER 5**

**FLOW CHART**

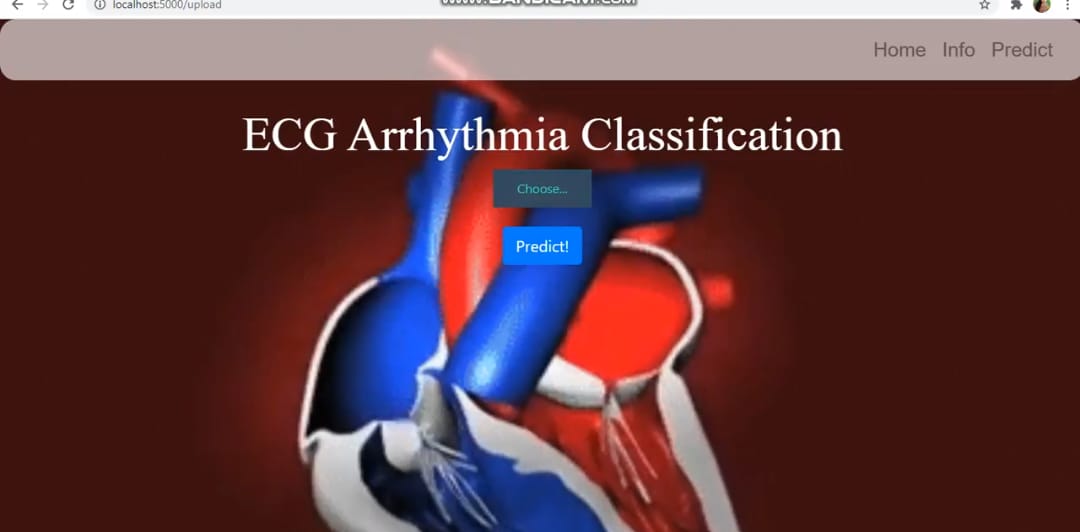


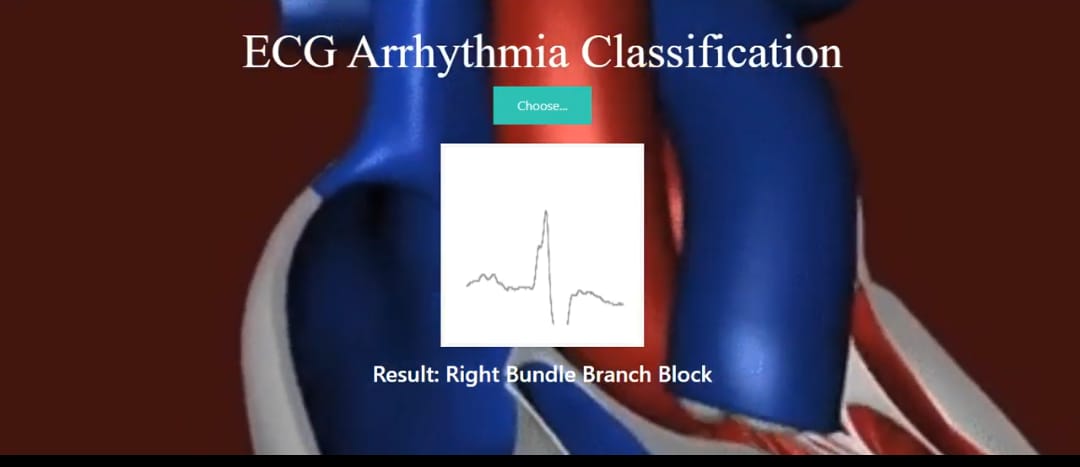
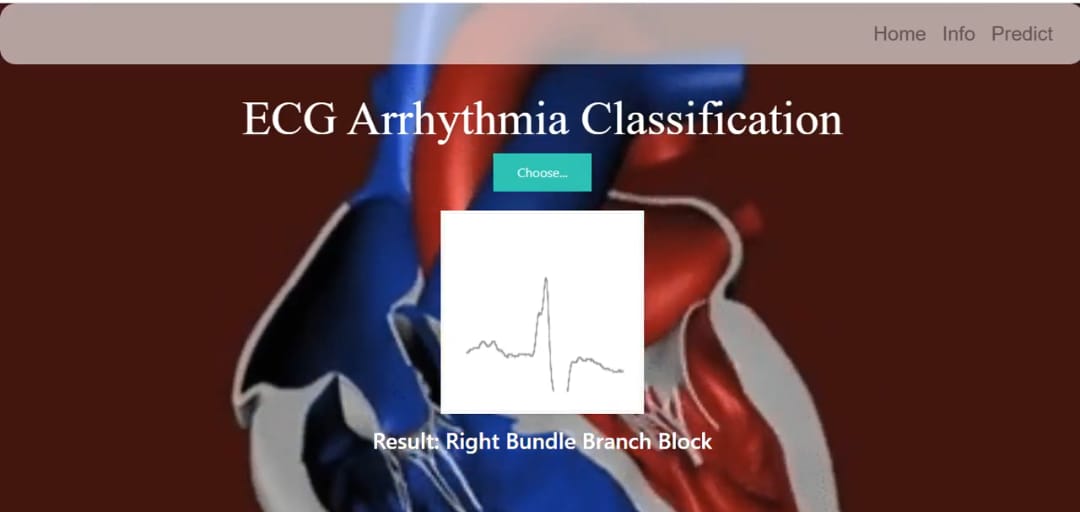
**2 D CNN Classifier**

**CHAPTER 6**

**RESULTS**

**Final output of the project:**

****

****

**CHAPTER 7**

**ADVANTAGES AND DISADVANTAGES**

**Advantages:**

* Furthermore, detecting ECG arrhythmia with ECG images resembles how medical experts diagnose arrhythmia since they observe an ECG graph from the patients throughout the monitor, which shows a series of ECG images. In other words, the proposed scheme can be applied to the medical robot that can monitors the ECG signals and helps the experts to identify ECG arrhythmia more precisely**.**
* Using ECG image as an input data of the ECG arrhythmia classiﬁcation also beneﬁts in the sense of robustness.
* In addition, training data can be enlarged by augmenting the ECG images which result in higher classiﬁcation accuracy.

**Disadvantages:**

* The symptoms of the arrhythmia might not be seen during the ECG signal capturing period.
* ECG signal properties (such as period, and amplitude) vary from person to person and depends on different factors such as age, gender, physical conditions, and lifestyle. Finding a generalized framework along with the related standards to be functional for general population is problematic.
* The volume of data to be considered for ECG signal analysis is large. Hence there is a higher probability of having a false diagnosis of arrhythmia.

**CHAPTER 8**

**APPLICATIONS**

* Evaluating the effect of cardic drugs.
* Systematic diseases
* High Performance Computing.

**CHAPTER 9**

**CONCLUSION**

In this study, we proposed a 2-D CNN-based classiﬁcation model for automatic classiﬁcation of cardiac arrhythmias using ECG signals. An accurate taxonomy of ECG signals is extremely helpful in the prevention and diagnosis of CVDs. Deep CNN has proven useful in enhancing the accuracy of diagnosis algorithms in the fusion of medicine and modern machine learning technologies.These results indicate that the prediction and classiﬁcation of arrhythmia with 2-D ECG representation as spectrograms and the CNN model is a reliable operative technique in the diagnosis of CVDs. The proposed scheme can help experts diagnose CVDs by referring to the automated classiﬁcation of ECG signals. The present research uses only a single-lead ECG signal. The effect of multiple lead ECG data to further improve experimental cases will be studied in future work.

**CHAPTER 10**

**FUTURE SCOPE**

**Enhancements that can be made in the future:**

According to the best classification methods represented in , CNN-based have proven to be effective for arrhythmia classification. Recent trend of research in this scope shows that dynamic classification methods that are capable to learn both short and long term contents of the signal in an efficient way, would be employed for such applications. CNN has shown excellent performance in classifying different types of arrhythmia. This powerful method would be one of the most efficient learning tool for this application.

**CHAPTER 11**

**BIBILOGRAPHY**

References of previous works or websites visited/books referred for analysis about the project, previous solution findings etc.

**https://arxiv.org/abs/1804.06812**

**CHAPTER12**

**APPENDIX**

* User interacts with User interface to upload image
* Uploaded image is analyzed by the model which is integrated
* Once model analyses the uploaded image, the prediction is showcased on the UI

To accomplish this, we have to complete 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

**Source 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("index6.html")

@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)

pred=model.predict\_classes(x)

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

port = int(os.getenv("PORT"))

if \_\_name\_\_=="\_\_main\_\_":

app.run(port=5000,debug=False)

**Source code(model building) :**

from keras.preprocessing.image import ImageDataGenerator

train\_datagen=ImageDataGenerator(rescale=1./255,shear\_range=0.2,zoom\_range=0.2,horizontal\_flip=True)

test\_datagen=ImageDataGenerator(rescale=1./255)

train=train\_datagen.flow\_from\_directory(directory=r'F:\dipu\data\train',target\_size=(64,64),batch\_size=32,class\_mode='categorical')

test=test\_datagen.flow\_from\_directory(directory=r'F:\dipu\data\test',target\_size=(64,64),batch\_size=32,class\_mode='categorical')

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.optimizers import Adam

model=Sequential()

#First convolution layer

model.add(Conv2D(32,(3,3),input\_shape=(64,64,3),activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#Second convolution layer

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

model.add(MaxPooling2D(pool\_size=(2,2)))

#Flattening the layers

model.add(Flatten())

#Adding first hidden layer

model.add(Dense(32))

#Adding first hidden layer

model.add(Dense(6,activation="softmax"))

model.summary()

model.compile(loss='categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

model.save('ECG.h5')

from tensorflow.keras.models import load\_model

from keras.preprocessing import image

model = load\_model("ECG.h5")