

PROJECT REPORT

Classification Of Arrhythmia By Using Deep Learning With 2-d Ecg Spectral Image Representation

SUBMITTED BY

PNT2022TMID36525

SHANMUGAM M- 112519106004

SARANYA S - 112519106003

SHARMILAR -112519106005

VIGNESHT - 112519106006

TABLE OF CONTENTS

- 1. INTRODUCTION**
 - a. Project Overview
 - b. Purpose
 - 2. LITERATURE SURVEY**
 - a. Existing problem
 - b. References
 - c. Problem Statement Definition
 - 3. IDEATION & PROPOSED SOLUTION**
 - a. Empathy Map Canvas
 - b. Ideation & Brainstorming
 - c. Proposed Solution
 - d. Problem Solution fit
 - 4. REQUIREMENT ANALYSIS**
 - 5. PROJECT DESIGN**
 - a. Data Flow Diagrams
 - b. Solution & Technical Architectur
 - 6. PROJECT PLANNING & SCHEDULING**
 - a. Sprint Planning & Estimation
 - 7. CODING & SOLUTIONING (Explain the features added in the project along with code)**
 - 8. TESTING**
 - a. Test Cases
 - b. User Acceptance Testing
 - 9. RESULTS**
 - a. Performance Metrics
 - 10. ADVANTAGES & DISADVANTAGES**
 - 11. CONCLUSION**
 - 12. FUTURE SCOPE**
 - 13. APPENDIX**
- Source Code
- GitHub & Project Demo Link

1.Introduction:

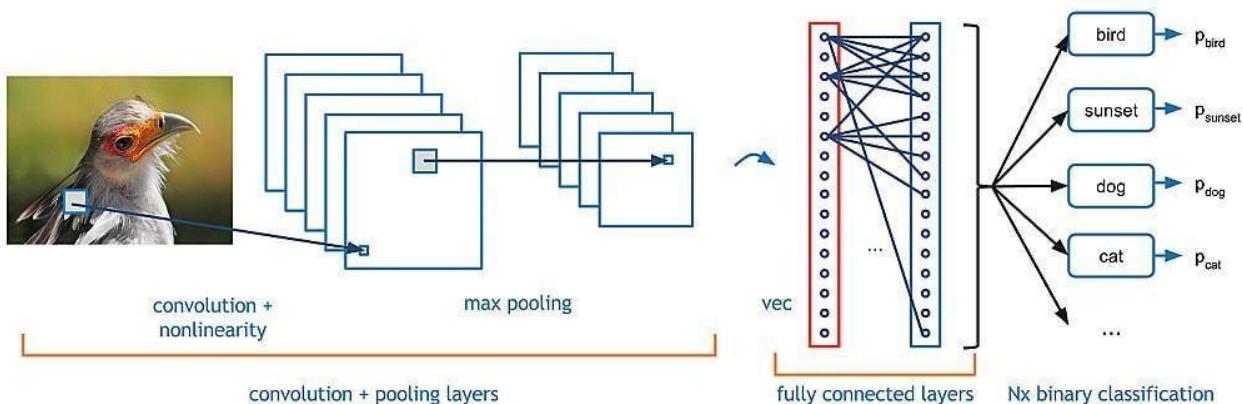
a. Overview:

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.

b. Purpose:

In the past few decades, Deep Learning has proved to be a compelling tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is AI Convolution Neural Networks.



In deep learning, a convolutional neural network(CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

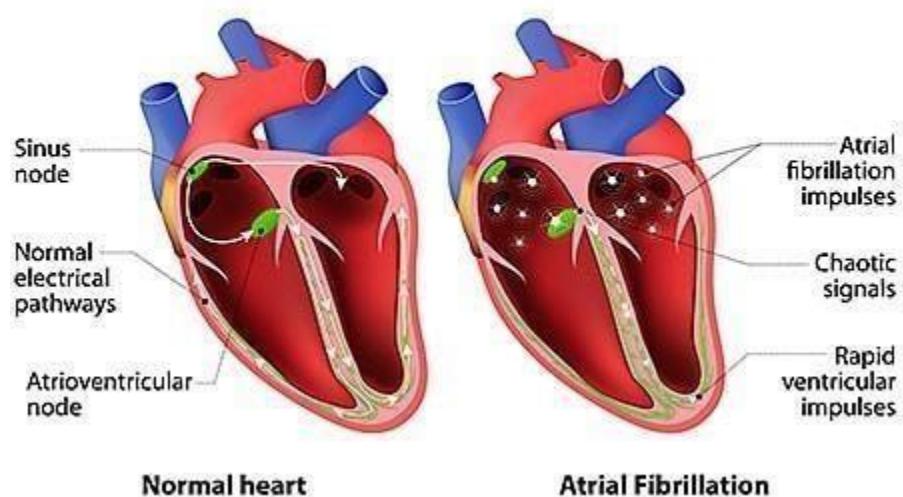
2.Literature Survey:

2.1 Existing Problem:

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**.

Cardiac arrhythmia

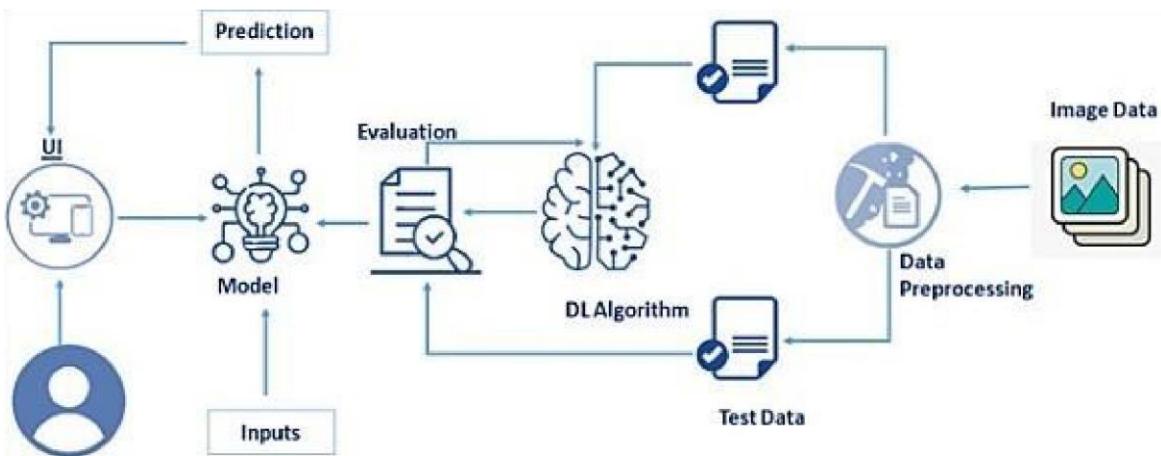


2.1 Proposed Solution:

An "ambulatory electrocardiogram" or an ECG about the size of a postcard or digital camera that the patient will be using for 1 to 2 days, or up to 2 weeks. The test measures the movement of electrical signals or waves through the heart. These signals tell the heart to contract (squeeze) and pump blood. The patient will have electrodes taped to your skin. It's painless, although some people have mild skin irritation from the tape used to attach the electrodes to the chest. They can do everything but shower or bathe while wearing the electrodes. After the test period, patient will go back to see your doctor. They will be downloading the information.

Theoretical Experience:

Block Diagram:

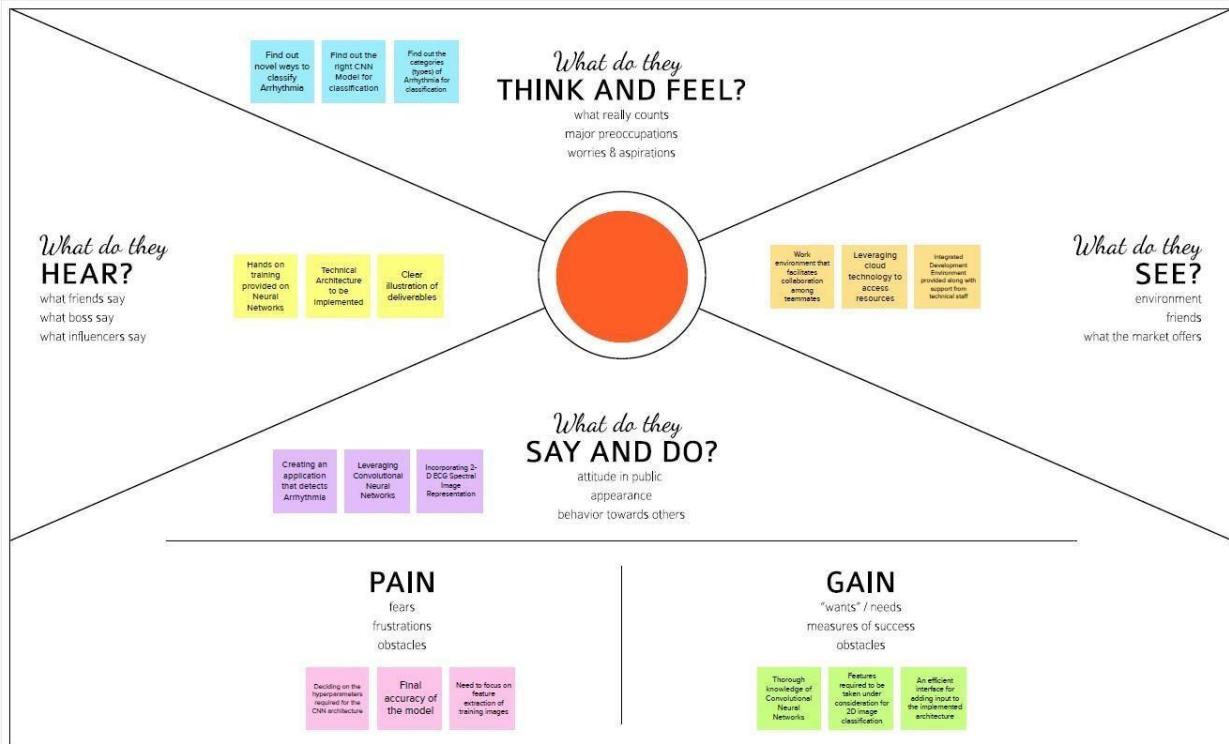
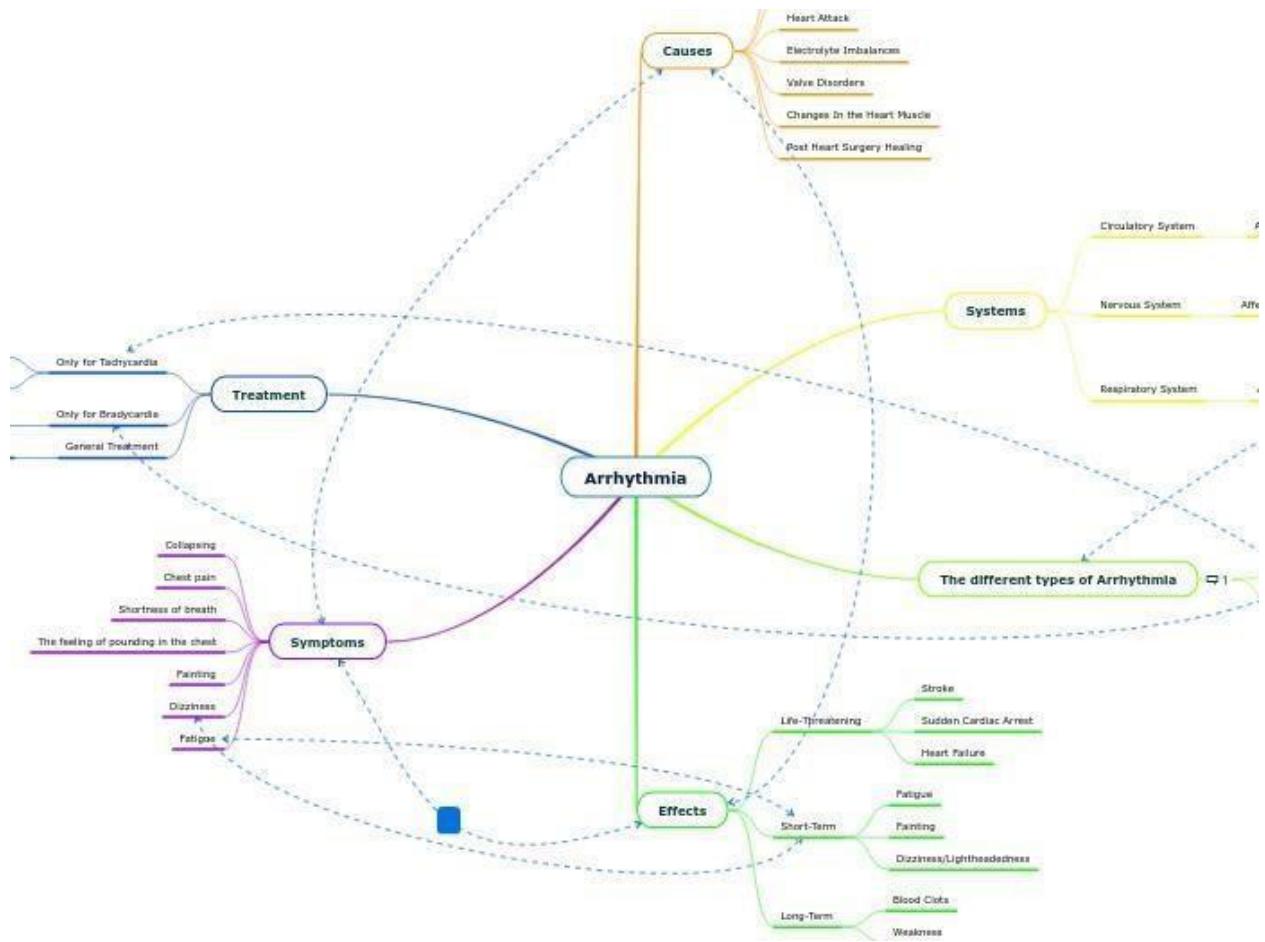


We will prepare the project by following the below steps:

- i. We will be working with Sequential type of modeling ii. We will be working with Keras capabilities iii. We will be working with image processing techniques iv. We will build a web application using the Flask framework.
- v. Afterwards we will be training our dataset in the IBM cloud and building another model from IBM and we will also test it.

3 IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP:



3.2 PROBLEM SOLUTION FIT:

<p>1. CUSTOMER SEGMENT(S) Who is your customer?</p> <p>People who are diagnosed with heart ailments or people who at risk of developing heart complications. Average age of target customers is above fifty.</p>	<p>6. CUSTOMER What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.</p> <p>Lack of awareness about checking for heart ailments due to Arrhythmia (irregular heart beats). Quite often, people think that it is not required to invest in an application that they think is unimportant to have.</p>	<p>5. AVAILABLE SOLUTIONS Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking</p> <p>The existing norm followed by people is get themselves diagnosed only after an accident occurs (such as a heart attack). Our idea goes by the saying 'prevention is better than cure'. By detecting the presence of heart ailments at the early stages, customers can benefit greatly by taking preventing measures.</p>
<p>2. JOBS-TO-BE-DONE / PROBLEMS Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.</p> <p>The major problem faced by people diagnosed with Arrhythmia related issues is the lack of awareness. Almost everyone is unaware of the fact that one can detect the type of heart ailment that the particular individual can be affected of, by analyzing the ECG patterns of heart beats. A simple software/hardware solution that addresses this issue can be a lifesaver for many.</p>	<p>9. PROBLEM ROOT CAUSE What is the real reason that this problem exists? What is the back story behind the need to do this job? i.e. customers have to do it because of the change in regulations.</p> <p>Negligence in taking care of health is the major reason for heart related risks. Also, the lack of an easy to use product that diagnoses the heart ailment makes it further complicated for people to get themselves diagnosed. The only solution that currently exists is going to hospitals once they get into an accident. The void present in this context promoted us to work on this particular product, and the goal is to make it easy to use so that everyone can benefit.</p>	<p>7. BEHAVIOUR What does your customer do to address the problem and get the job done? i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)</p> <p>At present, the only solution that exists is to go to a hospital, more preferably – a multi-speciality clinic and consult with a cardiologist and get themselves diagnosed. While visiting hospitals is absolutely necessary in this context, the lack of a tool that detects such ailments in advance might turn out to be quite problematic to people.</p>
<p>3. TRIGGERS What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.</p> <p>The need to prevent any heart related untoward incident that causes loss of life</p>	<p>10. YOUR SOLUTION If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality. If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.</p> <p>Develop an easy-to-use application that efficiently classifies and indicates the type of heart ailment (heart attack, myocardial infarction etc.) based on analyzing 2D spectral images of Arrhythmia. The application should indicate the type of risk, and at the same time it should indicate the preventive measures that need to be followed in order to keep oneself healthy.</p>	<p>8. CHANNELS OF BEHAVIOUR What kind of actions do customers take online? Extract online channels from #7</p> <p>Customers do basic googling to know about heart ailments in general. But there is no efficient online tool that studies Arrhythmia and gives out the heart ailment.</p> <p>8.1 ONLINE What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.</p> <p>Customers visit hospitals and go for general check-ups.</p>

4. PREREQUISITES ANALYSIS:

Hardware Components used:

Since we are using the IBM cloud as a platform to execute this project we don't need any hardware components other than our system.

Software Components Used:

We will be using Visual Studio which is installed in our system and Watsonstudio from the IBM cloud to complete the project.

Visual Studio Code

Visual Studio Code, also commonly referred to as VS Code is a source-code editor made by Microsoft with the Electron Framework, for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality

WATSON STUDIO:

Watson Studio is one of the core services in Cloud Pak for Data as a Service.

Watson Studio provides you with the environment and tools to solve your business problems by collaboratively working with data. You can choose the tools you need to analyze and visualize data, to cleanse and shape data, or to build machine learning models.

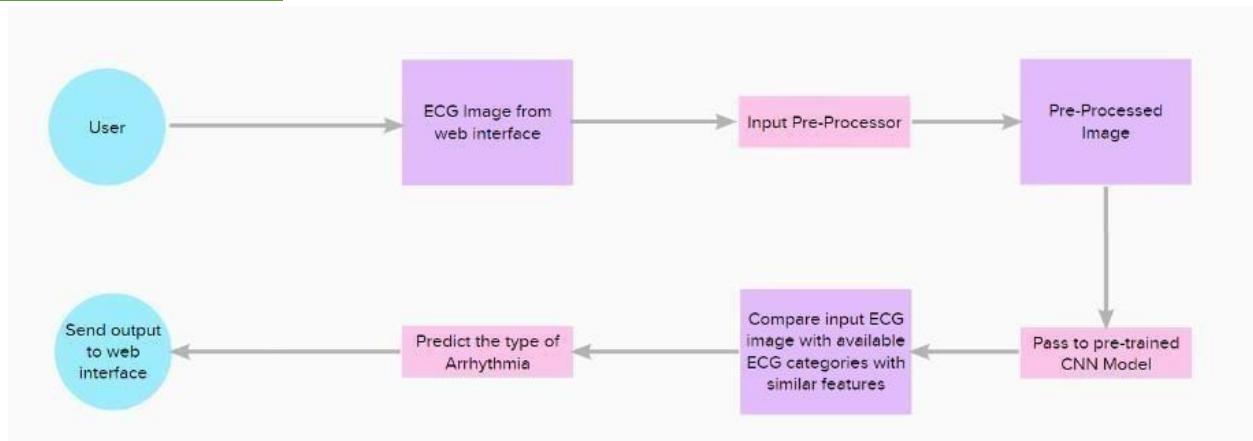
This illustration shows how the architecture of Watson Studio is centered around the project.

A project is a workspace where you organize your resources and work with data.

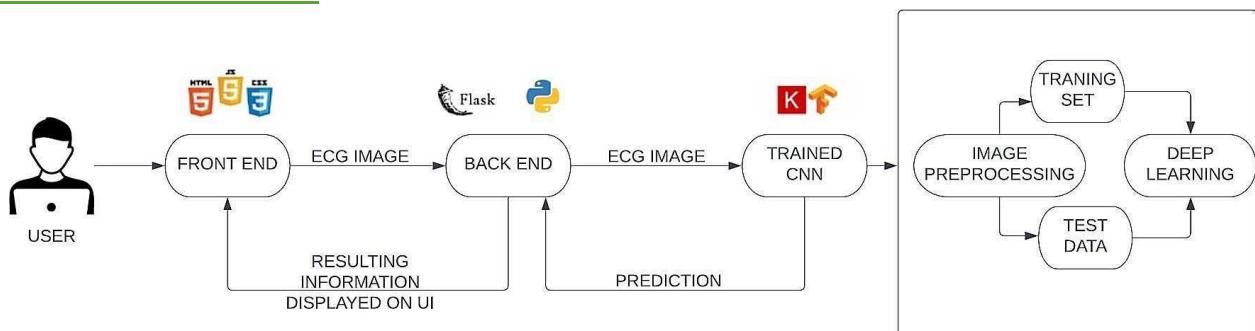
Watson Studio projects fully integrate with the catalogs and deployment spaces:

1. Deployment spaces are provided by the Watson Machine Learning service. You can easily move assets between projects and deployment spaces.

5. DATAFLOW DIAGRAM:



SOLUTION ARCHITECTURE:



6. PROJECT PLANNING:

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	7 Days	21 Oct 2022	27 Oct 2022	20	28 Oct 2022
Sprint-2	20	7 Days	29 Oct 2022	04 Nov 2022	20	04 Nov 2022
Sprint-3	20	7 Days	05 Nov 2022	11 Nov 2022	20	12 Nov 2022
Sprint-4	20	7 Days	13 Nov 2022	19 Nov 2022	20	19 Nov 2022

7. CODING AND SOLUTION:

Experimental Investigations:

In this project, we have deployed our training model using CNN on IBM Watson studio and in our local machine. We are deploying 4 types of CNN layers in a sequential manner, starting from :

- **Convolutional layer 2D:** A 2-D convolutional layer applies sliding convolutional filters to 2-D input. The layer convolves the input by moving the filters along the input vertically and horizontally and computing the dot product of the weights and the input, and then adding a bias term.
- **Pooling Layer :** Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer.
- **Fully-Connected layer :** After extracting features from multiple convolution layers and pooling layers, the fully-connected layer is used to expand the connection of all features. Finally, the SoftMax layer makes a logistic regression classification. Fully-connected layer transfers the weighted sum of the output of the previous layer to the activation function.
- **Dropout Layer :** There is usually a dropout layer before the fully connected layer. The dropout layer will temporarily disconnect some neurons from the network according to a certain probability during the training of the convolution neural network, which reduces the joint adaptability between neuron nodes, reduces overfitting, and enhances the generalization ability of the network.

Flow Chart & Results with Screenshots:

a. Flow Chart & Results by training model in local machine:

i. Dataset Collection:

The dataset contains six classes:

1. Left BundleBranch Block
2. Normal
3. Premature Atrial Contraction
4. Premature Ventricular Contractions
5. Right BundleBranch Block
6. Ventricular Fibrillation

ii. Image Preprocessing:

Image Pre-processing includes the following main tasks

1. Import ImageDataGenerator Library:

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.

```
In [5]: 1 from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

1. Configure ImageDataGenerator Class:

There are five main types of data augmentation techniques for image data; specifically:

1. Image shifts via the width_shift_range and height_shift_range arguments.
2. Image flips via the horizontal_flip and vertical_flip arguments.

3. Image rotates via the rotation_range argument
 4. Image brightness via the brightness_range argument.
5. Image zooms via the zoom_range argument.

An instance of the ImageDataGenerator class can be constructed for train and test.

```
In [6]: 1 train_datagen = ImageDataGenerator(rescale = 1./255,shear_range = 0.2,zoom_range = 0.2,horizontal_flip = True)
2 test_datagen = ImageDataGenerator(rescale = 1./255)
```

1. Applying ImageDataGenerator functionality to the trainset and test set:

We will apply ImageDataGenerator functionality to Trainset and Testset by using the following code

This function will return batches of images from the subdirectories Left Bundle Branch Block, Normal, Premature Atrial Contraction, Premature Ventricular Contractions, Right Bundle Branch Block and Ventricular Fibrillation, together with labels 0 to 5{'Left Bundle Branch Block': 0, 'Normal': 1, 'Premature Atrial Contraction': 2, 'Premature Ventricular Contractions': 3, 'Right Bundle Branch Block': 4, 'Ventricular Fibrillation': 5}

```
In [7]: 1 x_train = train_datagen.flow_from_directory("/content/data/train",target_size = (64,64),batch_size = 32,\
2                                         class_mode = "categorical")
3 x_test = test_datagen.flow_from_directory("/content/data/test",target_size = (64,64),batch_size = 32,\
4                                         class_mode = "categorical")
```

```
Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.
```

We can see that for training there are 15341 images belonging to 6 classes and for testing there are 6825 images belonging to 6 classes.

1. Model Building

We are ready with the augmented and pre-processed image data, we will begin our build our model by following the below steps:

1. Import the model building Libraries:

```
In [4]: 1 from tensorflow.keras.models import Sequential  
2 from tensorflow.keras.layers import Dense  
3 from tensorflow.keras.layers import Convolution2D  
4 from tensorflow.keras.layers import MaxPooling2D  
5 from tensorflow.keras.layers import Flatten
```

- **Initializing the model:**

Keras has 2 ways to define a neural network:

1. Sequential
2. Function API

The Sequential class is used to define linear initializations of network layers which then, collectively, constitute a model. In our example below, we will use the Sequential constructor to create a model, which will then have layers added to it using the add ()method. Now, will initialize our model.

- 1. **Adding CNN Layers:**

We are adding a convolution layer with an activation function as “relu” and with a small filtersize (3,3) and a number of filters as (32) followed by a max-pooling layer.

The Max pool layer is used to downsample the input. The flatten layer

flattens the input.

```
In [9]: 1 #MODEL BUILDING  
  
In [10]: 1 model = Sequential()  
  
In [11]: 1 model.add(Convolution2D(32,(3,3),input_shape = (64,64,3),activation = "relu"))  
  
In [12]: 1 model.add(MaxPooling2D(pool_size = (2,2)))  
  
In [13]: 1 model.add(Convolution2D(32,(3,3),activation='relu'))  
  
In [14]: 1 model.add(MaxPooling2D(pool_size=(2,2)))  
  
In [15]: 1 model.add(Flatten()) # ANN Input...
```

- **Adding Hidden Layers:**

Dense layer is deeply connected neural network layer. It is most common and frequently used layer

```
In [16]: 1 #Adding Dense Layers  
  
In [17]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))  
  
In [18]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))  
  
In [19]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))  
  
In [20]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))  
  
In [21]: 1 model.add(Dense(units = 128,kernel_initializer = "random_uniform",activation = "relu"))
```

● Adding Output Layer:

Understanding the model is very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

```
In [22]: 1 model.add(Dense(units = 6,kernel_initializer = "random_uniform",activation = "softmax"))
```

```
In [23]: 1 model.summary()
```

```
Model: "sequential"  
  
=====  
Layer (type)          Output Shape         Param #  
=====  
conv2d (Conv2D)       (None, 62, 62, 32)    896  
max_pooling2d (MaxPooling2D) (None, 31, 31, 32) 0  
conv2d_1 (Conv2D)      (None, 29, 29, 32)    9248  
max_pooling2d_1 (MaxPooling2D) (None, 14, 14, 32) 0  
flatten (Flatten)     (None, 6272)        0  
dense (Dense)          (None, 128)        802944  
dense_1 (Dense)        (None, 128)        16512  
dense_2 (Dense)        (None, 128)        16512  
dense_3 (Dense)        (None, 128)        16512  
dense_4 (Dense)        (None, 128)        16512  
dense_5 (Dense)        (None, 6)           774  
=====  
Total params: 879,910  
Trainable params: 879,910  
Non-trainable params: 0
```

1. Configure the Learning Process:

1. The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find error or deviation in the learning process. Keras requires loss function during the model compilation process.
2. Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
3. Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in the training process.

```
In [24]: 1 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

- **Training the model:**

We will train our model with our image dataset. `fit_generator` functions used to train a deep learning neural network.

```
In [25]: 1 model.fit_generator(generator=x_train, steps_per_epoch = len(x_train), epochs=9, validation_data=x_test,\n          validation_steps = len(x_test))
Epoch 1/9
480/480 [=====] - 99s 203ms/step - loss: 1.4415 - accuracy: 0.4788 - val_loss: 1.6093 - val_accuracy: 0.3193
Epoch 2/9
480/480 [=====] - 96s 201ms/step - loss: 0.9465 - accuracy: 0.6495 - val_loss: 1.3444 - val_accuracy: 0.5121
Epoch 3/9
480/480 [=====] - 97s 201ms/step - loss: 0.5540 - accuracy: 0.8018 - val_loss: 0.7785 - val_accuracy: 0.7698
Epoch 4/9
480/480 [=====] - 99s 205ms/step - loss: 0.2770 - accuracy: 0.9069 - val_loss: 0.6690 - val_accuracy: 0.8296
Epoch 5/9
480/480 [=====] - 97s 201ms/step - loss: 0.2037 - accuracy: 0.9388 - val_loss: 0.6057 - val_accuracy: 0.8416
Epoch 6/9
91/480 [====>.....] - ETA: 1:09 - loss: 0.1595 - accuracy: 0.9499
```

- **Saving the model:**

The model is saved with .h5 extension as follows

An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

```
In [26]: 1 #Saving Model.
2 model.save('ECG.h5')
```

1. Testing the model:

Load necessary libraries and load the saved model using `load_model`. Taking an image as input and checking the results

Note: The target size should be for the image that should be the same as the target size that you have used for training.

```

In [26]: 1 #Saving Model.
          2 model.save('ECG.h5')

In [28]: 1 from tensorflow.keras.models import load_model
          2 from tensorflow.keras.preprocessing import image

In [29]: 1 model=load_model('ECG.h5')

In [30]: 1 img=image.load_img("/content/Unknown_image.png",target_size=(64,64))

In [31]: 1 x=image.img_to_array(img)

In [32]: 1 import numpy as np

In [33]: 1 x=np.expand_dims(x, axis=0)

In [34]: 1 pred = model.predict(x)
          2 y_pred=np.argmax(pred)
          3 y_pred

Out[34]: 1

In [35]: 1 index=['left Bundle Branch block',
          2           'Normal',
          3           'Premature Atrial Contraction',
          4           'Premature Ventricular Contraction',
          5           'Right Bundle Branch Block',
          6           'Ventricular Fibrillation']
          7 result = str(index[y_pred])
          8 result

Out[35]: 'Normal'

```

The unknown image uploaded is:



Here the output for the uploaded result is normal.

1. Application Building:

In this section, we will be building a web application that is integrated into the model we built. A UI is provided for the user where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.

This section has the following tasks • **Building**

HTML Pages:

- We use HTML to create the front end part of the web page.
- Here, we created 4 html pages- home.html, predict_base.html, predict.html, information.html.

home.html displays the home page.

Welcome To ECG- Image Based Heartbeat Classification Application For Arrhythmia Detection

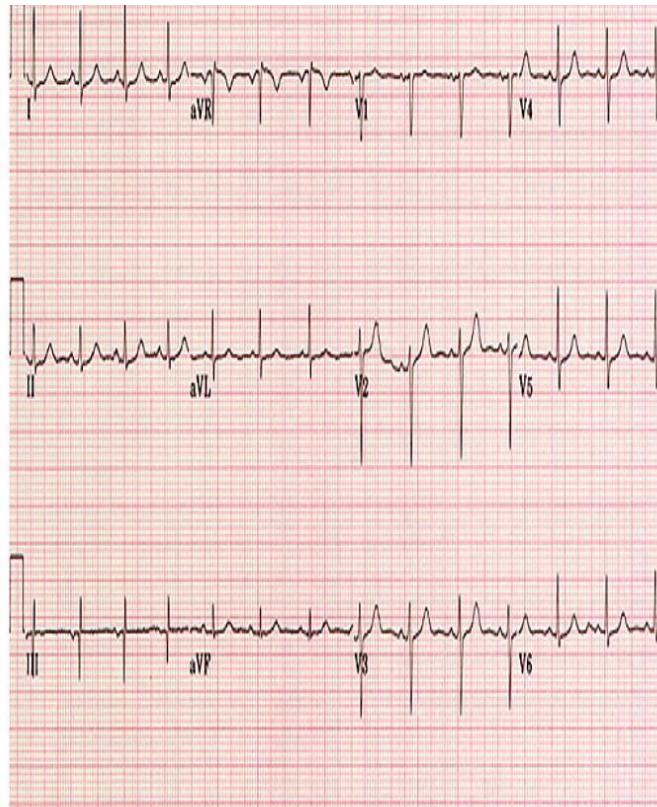


[information.html](#) displays all important details to be known about ECG.

NORMAL

Note that the heart is beating in a regular sinus rhythm between 60 - 100 beats per minute (specifically 82 bpm). All the important intervals on this recording are within normal ranges.

The normal ECG patterns seen in children differ considerably from those in adults.



RIGHT BUNDLE BRANCH BLOCK

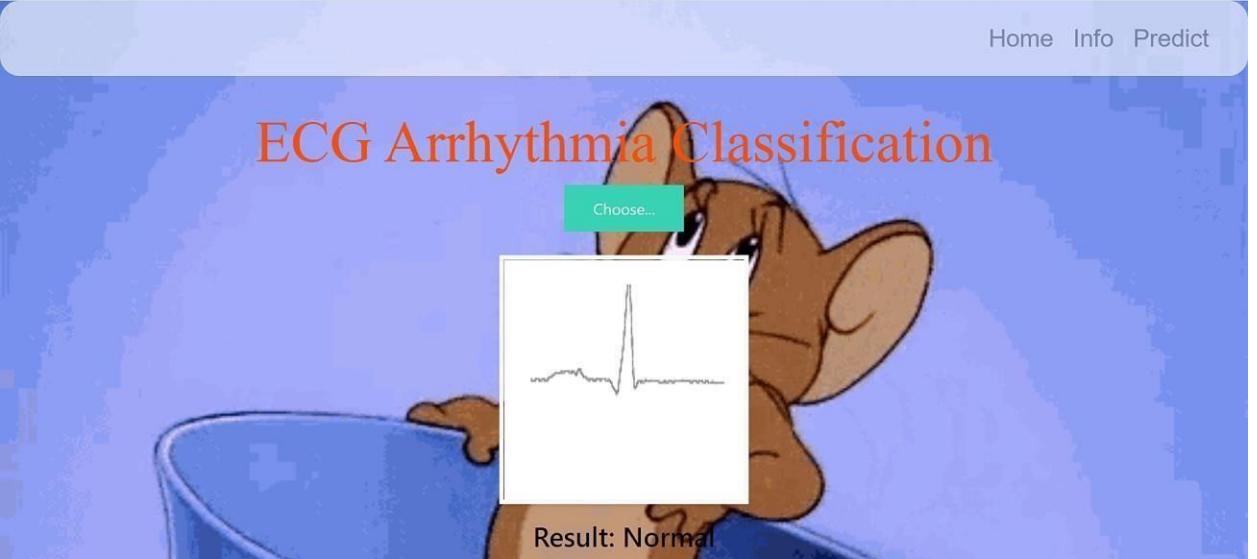
Right bundle branch block is associated with structural changes from stretch or ischemia to the myocardium. It can also occur iatrogenically from certain common cardiac procedures, such as right heart catheterization. Although there is no significant association with cardiovascular risk factors, the presence of a right bundle branch block is a predictor of mortality in myocardial infarction, heart failure, and certain heart blocks. In asymptomatic patients, isolated right bundle branch block typically does not need further evaluation.



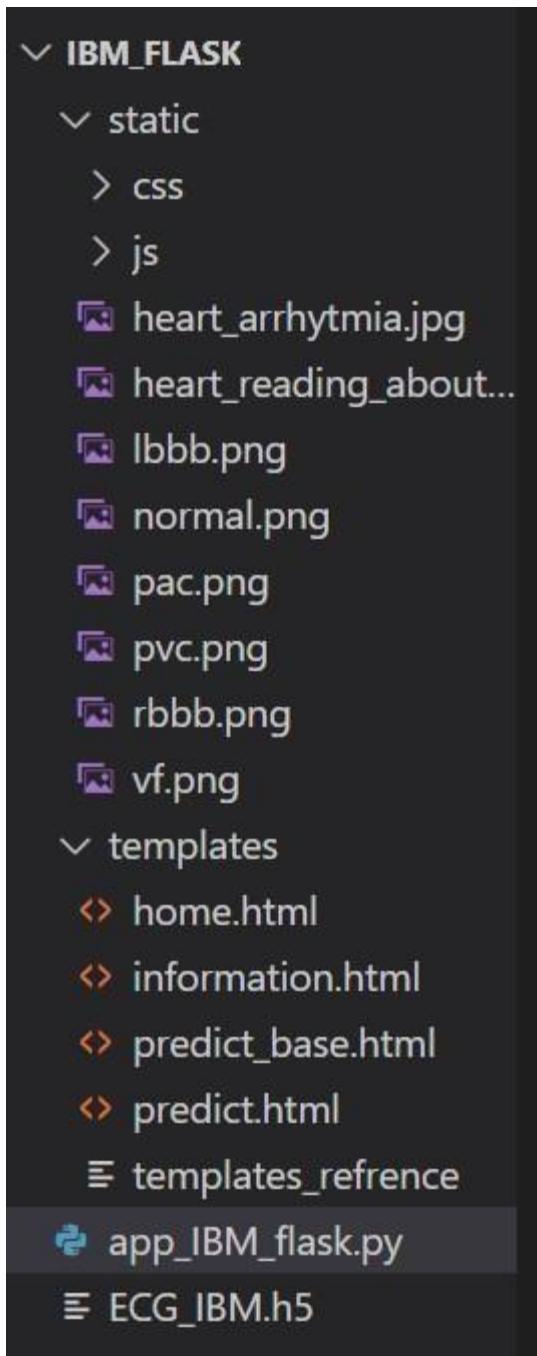
- predict-base.html and predict.html accept input from the user and predicts the values.

ECG Arrhythmia Classification

Choose...



Result: Normal



- **Building server-side script:**

We will build the file 'app.py' which is a web framework written in python for server-side scripting.

1. The app starts running when the “`_name_`” constructor is called in main.
2. `render_template` is used to return HTML file.
3. “GET” method is used to take input from the user.
4. “POST” method is used to display the output to the user.

```

import os
import numpy as np #used for numerical analysis
from flask import Flask,request,render_template
from tensorflow.keras.models import load_model#to load our trained model
from tensorflow.keras.preprocessing import image
app=Flask(__name__)#our flask app
model=load_model('ECG_IBM.h5')#loading the model
@app.route("/") #default route
def about():
    return render_template("home.html")#rendering html page
@app.route("/about") #default route
def home():
    return render_template("home.html")#rendering html page
@app.route("/info") #default route
def information():
    return render_template("information.html")#rendering html page
@app.route("/upload") #default route
def test():
    return render_template("predict.html")#rendering html page
@app.route("/predict",methods=["GET","POST"]) #route for our prediction
def upload():
    if request.method=='POST':
        f=request.files['file'] #requesting the file
        basepath=os.path.dirname('__file__')#storing the file directory
        filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
        f.save(filepath)#saving the file
        img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
        x=image.img_to_array(img)#converting image to array
        x=np.expand_dims(x,axis=0)#changing the dimensions of the image

```

```

def upload():
    if request.method=='POST':
        f=request.files['file'] #requesting the file
        basepath=os.path.dirname('__file__')#storing the file directory
        filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
        f.save(filepath)#saving the file

        img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
        x=image.img_to_array(img)#converting image to array
        x=np.expand_dims(x,axis=0)#changing the dimensions of the image

        pred=model.predict(x)#predicting classes
        y_pred = np.argmax(pred)
        print("prediction",y_pred)#printing the prediction

        index=['Left Bundle Branch Block','Normal','Premature Atrial Contraction',
        'Premature Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
        result=str(index[y_pred])

        return result#resturing the result
    return None

#port = int(os.getenv("PORT"))
if __name__=="__main__":
    app.run(host='127.0.1.10',debug=False)#running our app
    #app.run(host='0.0.0.0', port=8000)

```

1. Running The App:

Run the file as : [python app_IBM_flask.py](#)

```
* Serving Flask app "app_IBM_flask" (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.1.10:5000/ (Press CTRL+C to quit)
```

Navigate to the localhost (<http://127.0.1.10:5000/>) where you can view your web page.

Flow Chart & Results by training model in IBM WATSON STUDIO:

Creating IBM cloud account:

We have to create an IBM Cloud Account and should log in.

Creating Watson StudioService & MachineLearning Service:

AI / Machine Learning (2)				
 Watson Machine Learning-bi	Default	Dallas	Watson Machine Learni...	 Active
 Watson Studio-yk	Default	Dallas	Watson Studio	 Active

The screenshot shows a user interface for managing assets in a project. On the left, there's a sidebar with a search bar and sections for 'Asset types' and 'Data'. The 'Data' section is expanded, showing four items: 'Unknown_image.png', 'data.zip', 'IBM_flash.rar', and 'my_model_vishva.tar'. Each item has a file icon, name, type, last modified date, and a three-dot menu. On the right, there's a large table titled 'Data' with columns for 'Name', 'Last modified', and a three-dot menu. Below the table is a placeholder area for dropping or browsing files.

Name	Last modified	
Unknown_image.png	17 hours ago Modified by you	⋮
data.zip	17 hours ago Modified by you	⋮
IBM_flash.rar	1 day ago Modified by you	⋮
my_model_vishva.tar	2 days ago Modified by you	⋮

This screenshot shows the 'Notebooks' section of the asset management interface. It lists two notebooks: 'CNN_ECG_IBM' and 'IBM_Training', both created with Python 3.9. The interface includes a sidebar with a search bar and sections for 'Asset types' and 'Notebooks'.

Name	Language	Last modified	
CNN_ECG_IBM	Python 3.9	2 hours ago Modified by you	⋮
IBM_Training	Python 3.9	15 hours ago Modified by you	⋮

Apply CNN algorithm and save the model and deploy it using API key generated:

```
In [77]: model.save('ECG_IBM.h5')
```

```
In [109]: !tar -zcvf ECG-arrhythmia-classification-model_new.tgz ECG_IBM.h5
ECG_IBM.h5
```

```
In [110]: ls -1
data/
ECG-arrhythmia-classification-model_new.tgz
ECG-classification.tgz
ECG_IBM.h5
```

```

Successfully installed watson-machine-learning-client-1.0.391

In [25]: from ibm_watson_machine_learning import APIClient
wml_credentials={
    "url": "https://us-south.ml.cloud.ibm.com",
    "apikey": "E9NC5hM6K_G7Uqd-SGH2TSeVUDsgsrDQMDq9Jra5Z"
}
client=APIClient(wml_credentials)

In [26]: client.spaces.list()
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
-----
ID          NAME      CREATED
e3662a1b-04df-46ed-9550-b081d08af72f  ECG_Classification  2022-11-16T15:33:42.493Z
-----

In [27]: space_uid=e3662a1b-04df-46ed-9550-b081d08af72f

In [28]: client.set.default_space(space_uid)

Out[28]: 'SUCCESS'

In [29]: client.set.default_space(space_uid)

Out[29]: 'SUCCESS'

In [30]: client.software_specifications.list()
-----
NAME          ASSET_ID          TYPE
base          0062b8c9-8b7d-44a0-a9b9-46c415adcb9  base
base          020d69ce-7ac1-5e68-ac1a-31189867356a  base
base          069ea134-3346-5748-b513-49120e15d288  base
-----

In [32]: model_details = client.repository.store_model(model='ECG_arrhythmia-classification-model_new.tgz',meta_props={
    client.repository.NodeMetaNames.NAME:"ECG_IBM",
    client.repository.NodeMetaNames.TYPE:"tensorflow_2.7",
    client.repository.NodeMetaNames.SOFTWARE_SPEC_UID:software_spec_uid)
model_id=client.repository.get_model_uid(model_details)

This method is deprecated, please use get_model_id()
/opt/conda/envs/Python-3.9/lib/python3.9/site-packages/ibm_watson_machine_learning/repository.py:1453: UserWarning: This method is deprecated, please use get_model_id()
  warn("This method is deprecated, please use get_model_id()")

In [33]: model_id
Out[33]: 'f5de404a-f13f-414f-9ee2-65185753e484'

In [34]: # @hidden_cell
# The following code contains the credentials for a file in your IBM Cloud Object Storage.
# You might want to remove those credentials before you share your notebook.
metadata_1 = {
    'IAM_SERVICE_ID': 'iam-ServiceId-dd7ca25c-789c-4f4a-aa0e-1b9f6ba759c1',
    'IBM_API_KEY_ID': 'EGYWm2jPVCTSQfMXOUZcqCb5gr05YDch2kklr_W802mzR',
    'ENDPOINT': 'https://s3.private.us.cloud-object-storage.appdomain.cloud',
    'IBM_AUTH_ENDPOINT': 'https://iam.cloud.ibm.com/oidc/token',
    'BUCKET': 'classificationarrhythmiausingdonotdelete-pr-jviatmq4bsdjnb',
    'FILE': 'Unknown_image.png'
}
client.repository.download(model_id,'my_model_vishva.tar.tar'
Successfully saved model content to file: 'my_model_vishva.tar.tar'
Out[34]: '/home/ususer/work/my_model_vishva.tar.tar'

```

For downloading the model we have to run the last part of the above code in the local jupyter notebook:

```

client.repository.download('2702a906-fdce-41d7-94d6-bfe8f920aca8','my_model_vishva.tar.gz')

Successfully saved model content to file: 'my_model_vishva.tar.gz'
'd:\Jithu Pgm working\Jithu\Vishva_programs\IBM_nalayathiran\IBM_training\my_model_vishva.tar.gz'

```

```
import os
import numpy as np # used for numerical analysis
from flask import Flask, request, render_template
# Flask-It is our framework which we are going to use to run/serve our app
# request-for accessing file which was uploaded by the user on our application
# render_template- used for rendering the html pages
from tensorflow.keras.models import load_model # to load our trained model
from tensorflow.keras.preprocessing import image

app = Flask(__name__) # our flask app
model = load_model('ECG_IBM.h5') # loading the model

@app.route("/")
def about():
    return render_template("home.html") # rendering html page
```

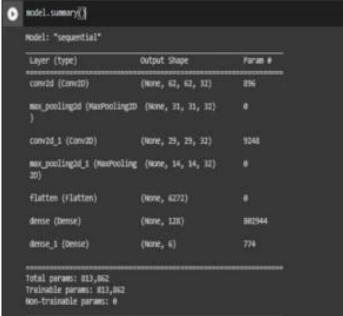
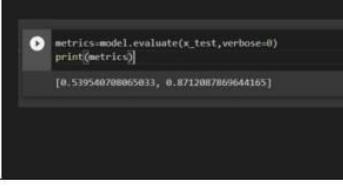
Hence we trained the model using IBM Watson.

8. TESTING:

a. PERFORMANCE TESTING:

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

S.No.	Parameter	Values	Screenshot
1.	Model Summary	-	
2.	Accuracy	Training Accuracy – 0.539540708065 Validation Accuracy -0.871208786964	
3.	Confidence Score (Only Yolo Projects)	Class Detected - Confidence Score -	-

b. USER ACCEPTANCE TESTING

This report shows the count of the bugs at each severity level, and how they were fixed.

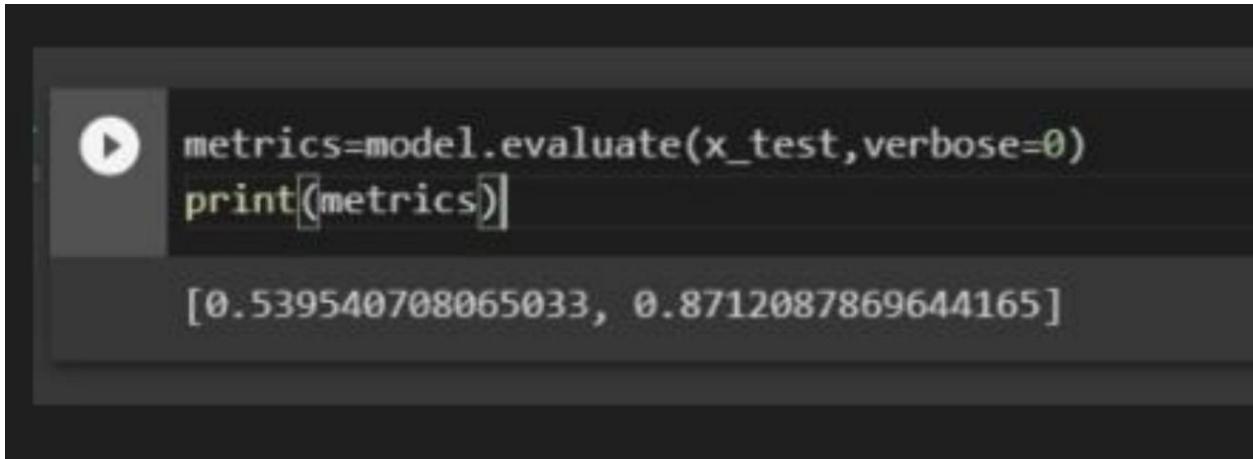
Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	4	2	3	14
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	9	2	4	15	30
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	17	14	13	21	65

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4

9.RESULT

a. PERFORMANCE METRICS:



```
metrics=model.evaluate(x_test,verbose=0)
print(metrics)

[0.539540708065033, 0.8712087869644165]
```

10. Advantages & Disadvantages:

a. Advantages:

- i. The proposed model predicts Arrhythmia in images with a high accuracy rate of nearly 96%
1. The early detection of Arrhythmia gives better understanding of disease causes, initiates therapeutic interventions and enables developing appropriate treatments.

b. Disadvantages:

- i. Not useful for identifying the different stages of Arrhythmia disease.
- ii. Not useful in monitoring motor symptoms

Applications : iii. It is useful for identifying the arrhythmia disease at an early stage.

- iv. It is useful in detecting cardiovascular disorders

11. Conclusion:

1. Cardiovascular disease is a major health problem in today's world. The early diagnosis of cardiac arrhythmia highly relies on the ECG.
2. Unfortunately, the expert level of medical resources is rare, visually identify the ECG signal is challenging and time-consuming.
3. The advantages of the proposed CNN network have been put to evidence.
4. It is endowed with an ability to effectively process the non-filtered dataset with its potential anti-noise features. Besides that, tenfold cross-validation is implemented in this work to further demonstrate the robustness of the network.

12. Future Scope:

For future work, it would be interesting to explore the use of optimization techniques to find a feasible design and solution. The limitation of our study is that we have yet to apply any optimization techniques to optimize the model parameters and we believe that with the implementation of the optimization, it will be able to further elevate the performance of the proposed solution to the next level.

13. APPENDIX:

SOURCE CODE:

[Model_generator.py](#)

```

[ ] from keras.preprocessing.image import ImageDataGenerator

[ ] from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

[ ] !unzip -u "/content/drive/MyDrive/IBM_nalayathiran/ECG.zip" -d "/content/Untitled Folder" #Consist of ECG DATASET

Double-click (or enter) to edit

[ ] train_datagen=ImageDataGenerator(rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True)
test_datagen=ImageDataGenerator(rescale=1./255)

[ ] x_train= train_datagen.flow_from_directory(directory=r"/content/Untitled Folder/data/train",target_size=(64,64),batch_size=32,class_mode='categorical')
x_test= train_datagen.flow_from_directory(directory=r"/content/Untitled Folder/data/test",target_size=(64,64),batch_size=32,class_mode='categorical')

Found 15341 images belonging to 6 classes.
Found 6825 images belonging to 6 classes.

[ ] import numpy as np
import tensorflow
from tensorflow.keras import models
from tensorflow.keras.models import Sequential
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,flatten
from tensorflow.keras.layers import Conv2D,MaxPooling2D

[ ] model = tensorflow.keras.Sequential()
model.add(Conv2D(32,(3,3),input_shape=(64,64,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Flatten())

[ ] model.add(Dense(128,kernel_initializer='random_uniform',activation='relu'))
model.add(Dense(6, activation='softmax'))

[ ] model.summary()

Model: "sequential"
Layer (type)          Output Shape         Param #
conv2d (Conv2D)        (None, 62, 62, 32)      896
max_pooling2d (MaxPooling2D) (None, 31, 31, 32)     0
conv2d_1 (Conv2D)       (None, 29, 29, 32)      9248
max_pooling2d_1 (MaxPooling2D) (None, 14, 14, 32)     0
flatten (Flatten)      (None, 6272)           0
dense (Dense)          (None, 128)            802944
dense_1 (Dense)         (None, 6)              774
Total params: 813,862
Trainable params: 813,862
Non-trainable params: 0

[ ] model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])

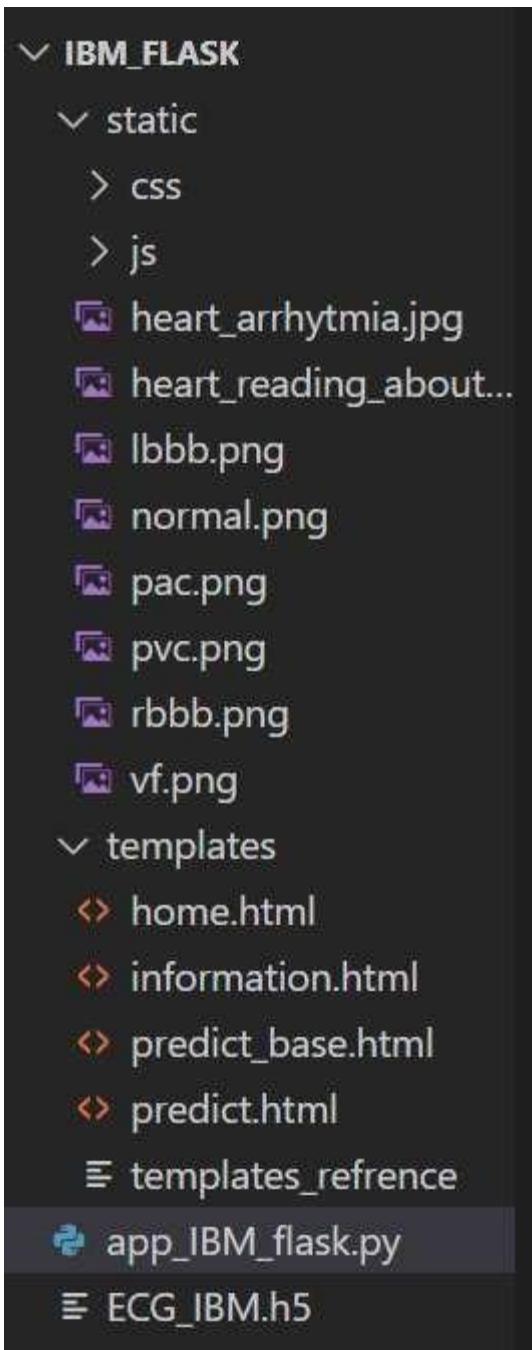
[ ] model.compile(optimizer='adam', loss='categorical_crossentropy',metrics=['accuracy'])

[ ] model.fit_generator(generator=x_train,steps_per_epoch=len(x_train),epochs=10,validation_data=x_test,validation_steps=len(x_test))

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: 'Model.fit_generator' is deprecated and will be removed in a future version. Please use 'Model.fit', which supports
    """Entry point for launching an IPython kernel.
Epoch 1/10
480/480 [=====] - 45s 75ms/step - loss: 0.8750 - accuracy: 0.6991 - val_loss: 0.5700 - val_accuracy: 0.7916
Epoch 2/10
480/480 [=====] - 35s 73ms/step - loss: 0.2549 - accuracy: 0.9232 - val_loss: 0.4851 - val_accuracy: 0.8431
Epoch 3/10
480/480 [=====] - 35s 74ms/step - loss: 0.1751 - accuracy: 0.9473 - val_loss: 0.3870 - val_accuracy: 0.8746
Epoch 4/10
480/480 [=====] - 35s 73ms/step - loss: 0.1487 - accuracy: 0.9555 - val_loss: 0.4410 - val_accuracy: 0.8621
Epoch 5/10
480/480 [=====] - 36s 76ms/step - loss: 0.1256 - accuracy: 0.9624 - val_loss: 0.4161 - val_accuracy: 0.8604
Epoch 6/10
480/480 [=====] - 35s 73ms/step - loss: 0.1034 - accuracy: 0.9686 - val_loss: 0.4483 - val_accuracy: 0.8709
Epoch 7/10
480/480 [=====] - 35s 72ms/step - loss: 0.0939 - accuracy: 0.9701 - val_loss: 0.4801 - val_accuracy: 0.8725
Epoch 8/10
480/480 [=====] - 36s 75ms/step - loss: 0.0817 - accuracy: 0.9751 - val_loss: 0.5306 - val_accuracy: 0.8643
Epoch 9/10
480/480 [=====] - 36s 74ms/step - loss: 0.0740 - accuracy: 0.9778 - val_loss: 0.4038 - val_accuracy: 0.8886
Epoch 10/10
480/480 [=====] - 35s 74ms/step - loss: 0.0666 - accuracy: 0.9792 - val_loss: 0.4857 - val_accuracy: 0.8828
<keras.callbacks.History at 0x7f3a3738ea50>

[ ] model.save("/content/drive/MyDrive/IBM_nalayathiran/ECG.h5")

```



[app_IBM_flask.py](#)

```

import os
import numpy as np #used for numerical analysis
from flask import Flask,request,render_template
from tensorflow.keras.models import load_model#to load our trained model
from tensorflow.keras.preprocessing import image
app=Flask(__name__)#our flask app
model=load_model('ECG_IBM.h5')#loading the model
@app.route("/") #default route
def about():
    return render_template("home.html")#rendering html page
@app.route("/about") #default route
def home():
    return render_template("home.html")#rendering html page
@app.route("/info") #default route
def information():
    return render_template("information.html")#rendering html page
@app.route("/upload") #default route
def test():
    return render_template("predict.html")#rendering html page
@app.route("/predict",methods=["GET","POST"]) #route for our prediction
def upload():
    if request.method=='POST':
        f=request.files['file'] #requesting the file
        basepath=os.path.dirname('__file__')#storing the file directory
        filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
        f.save(filepath)#saving the file
        img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
        x=image.img_to_array(img)#converting image to array
        x=np.expand_dims(x,axis=0)#changing the dimensions of the image

def upload():
    if request.method=='POST':
        f=request.files['file'] #requesting the file
        basepath=os.path.dirname('__file__')#storing the file directory
        filepath=os.path.join(basepath,"uploads",f.filename)#storing the file in uploads folder
        f.save(filepath)#saving the file

        img=image.load_img(filepath,target_size=(64,64)) #load and reshaping the image
        x=image.img_to_array(img)#converting image to array
        x=np.expand_dims(x,axis=0)#changing the dimensions of the image

        pred=model.predict(x)#predicting classes
        y_pred = np.argmax(pred)
        print("prediction",y_pred)#printing the prediction

        index=['Left Bundle Branch Block','Normal','Premature Atrial Contraction',
        'Premature Ventricular Contractions', 'Right Bundle Branch Block','Ventricular Fibrillation']
        result=str(index[y_pred])

        return result#resturing the result
    return None

if __name__=="__main__":
    app.run(host="127.0.1.10",debug=False)#running our app

```

[home.html](#)

```
<html>
<head>
<title>Home</title>
<style>
body{
    background-image: url('https://rh.gatech.edu/sites/default/files/images/research_horizons/BrokenHeart/solution_6_.png');
    background-size:1300px 1000px;
    background-repeat:no-repeat;
    padding:0;
    margin:0;
}
.pd{
padding-bottom:100%;}
.navbar
{
margin: 0px;
padding:20px;
background-color: #white;
opacity:0.6;
color:#yellow;
font-family:'Roboto',sans-serif;
font-style: italic;
border-radius:20px;
font-size:25px;
}
a
{
color:#red;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;
}
a:hover{
background-color:#blue;
color:#red;
border-radius:15px;
font-size:30px;
padding-left:10px;
}
a{
color:#red;
float:right;
text-decoration:none;
font-style:normal;
padding-right:20px;}
p{
font-color:"#3498eb";
font-style:italic;
font-size:30px;
font-family:Bell MT
```

information.html

```
<html>
<head>
<style>
.navbar
{
margin: 0px;
padding: 20px;
background-color: white;
opacity: 0.6;
color: black;
font-family: 'Roboto', sans-serif;
font-style: italic;
border-radius: 20px;
font-size: 25px;
}
a
{
color: red;
float: right;
text-decoration: none;
font-style: normal;
padding-right: 20px;
}
a:hover{
background-color: blue;
color: red;
border-radius: 15px;
font-size: 30px;
padding-left: 10px;
}
img{
width: 1200px;
height: 1000px;
padding: 10px;
}
h1{
font-size: 60px;
text-align: center;
color: #397ae3;
font-style: italic;
font-weight: bolder;
}
</style>
<title>Info</title>
</head>
<body>
<div class="navbar">
<a href="/upload" >Predict</a>
<a href="/info" >Info</a>
<a href="/about" >Home</a>
<br>
</div>
```

```

<br>
</div>
<div>
<h1><u>ECG- Image Based Heartbeat Classification Information Guide</u></h1>
<div>
<span></span>
<span></span>
<span></span>
<span></span>
<span></span>
<span></span>
</div>
</body>
</html>

```

[predict.html](#)

```

{% extends "predict_base.html" %} {% block content %}
<h2 style="color: #fc4e03; font-family:Times New Roman; font-size:60">ECG Arrhythmia Classification</h2>
<div>
    <form id="upload-file" method="post" enctype="multipart/form-data">
        <center>    <label for="imageUpload" class="upload-label">
            Choose...
        </label>
        <input type="file" name="file" id="imageUpload" accept=".png, .jpg, .jpeg">
    </center></form>
    <center> <div class="image-section" style="display:none;">
        <div class="img-preview">
            <div id="imagePreview">
            </div></center>
        </div>
        <center><div>
            <button type="button" class="btn btn-primary btn-lg " id="btn-predict">Predict!</button>
        </div></center>
    </div>
    <div class="loader" style="display:none;"></div>
    <h3 style="color: #800000" id="result">
        <span> </span>
    </h3>
</div></div>

```

{% endblock %}

[predict_base.html](#)

```
html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Predict</title>
    <link href="https://cdn.bootcss.com/bootstrap/4.0.0/css/bootstrap.min.css" rel="stylesheet">
    <script src="https://cdn.bootcss.com/popper.js/1.12.9/umd/popper.min.js"></script>
    <script src="https://cdn.bootcss.com/jquery/3.3.1/jquery.min.js"></script>
    <script src="https://cdn.bootcss.com/bootstrap/4.0.0/js/bootstrap.min.js"></script>
    <link href="{{ url_for('static', filename='css/flask_main_style.css') }}" rel="stylesheet">
<style>
.bar
{
margin: 0px;
padding: 20px;
background-color: #white;
opacity: 0.6;
color: #black;
font-family: 'Roboto', sans-serif;
font-style: italic;
border-radius: 20px;
font-size: 25px;
}

a
{
color: #grey;
float: right;
text-decoration: none;
font-style: normal;
padding-right: 20px;
}
a:hover{
background-color: #black;
color: #white;
border-radius: 15px;
font-size: 30px;
padding-left: 10px;
}
body
{
background-image: url("https://media.tenor.com/uFB7tkZq3QYAAAAd/heart-heartbeat.gif");
background-size: cover;
}
</style>
</head>
```

```

<body>
<div class="bar">
<a href="/upload" >Predict</a>
<a href="/info">Info</a>
<a href="/about">Home</a>
<br>
</div>
<div class="container">
    <center> <div id="content" style="margin-top:2em">{% block content %}{% endblock %}</div></center>
</div>
</body>

<footer>
    <script src="{{ url_for('static', filename='js/flask_main_js.js') }}" type="text/javascript"></script>
</footer>

</html>

```

flask_main.js

```

$(document).ready(function () {
    // Init
    $('.image-section').hide();
    $('.loader').hide();
    $('#result').hide();

    // Upload Preview
    function readURL(input) {
        if (input.files && input.files[0]) {
            var reader = new FileReader();
            reader.onload = function (e) {
                $('#imagePreview').css('background-image', 'url(' + e.target.result + ')');
                $('#imagePreview').hide();
                $('#imagePreview').fadeIn(650);
            }
            reader.readAsDataURL(input.files[0]);
        }
    }
    $("#imageUpload").change(function () {
        $('.image-section').show();
        $('#btn-predict').show();
        $('#result').text('');
        $('#result').hide();
        readURL(this);
    });
});

```

```

$( '#btn-predict' ).click(function () {
    var form_data = new FormData( $('#upload-file')[0] );

    // Show loading animation
    $(this).hide();
    $('.loader').show();

    // Make prediction by calling api /predict
    $.ajax({
        type: 'POST',
        url: '/predict',
        data: form_data,
        contentType: false,
        cache: false,
        processData: false,
        async: true,
        success: function (data) {
            // Get and display the result
            $('.loader').hide();
            $('#result').fadeIn(600);
            $('#result').text(' Result: ' + data);
            console.log('Success!');
        },
    });
});

```

flask_main_style.css

```

.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #39D2B4;
}

```

```
        background: #39D2B4;
        color: #fff;
        font-size: 1em;
        transition: all .4s;
        cursor: pointer;
    }

.upload-label:hover{
    background: #34495E;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #3498db; /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

GITHUB LINK:

<https://github.com/IBM-EPBL/IBM-Project-54600-1662355856>

<https://github.com/SHANSCIENTIST/Shanscientist>

THE END