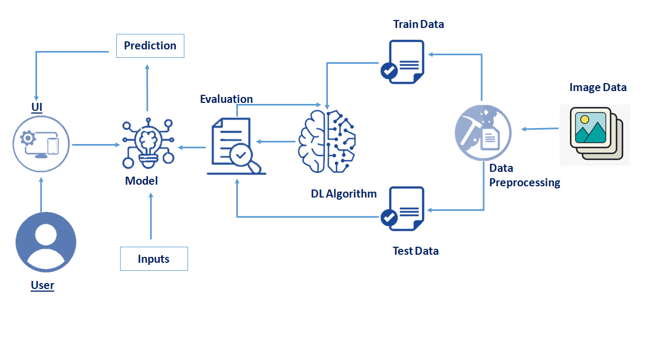
|  |  |
| --- | --- |
| Team ID | PNT2022TMID30495 |
| Project Name | Classification of arrhythmia by using deep learning with 2-d ECG Spectral Image Representation |

**Classification Of Arrhythmia By Using Deep Learning With 2-D ECG Spectral Image Representation**

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.

**Technical Architecture:**



**Project Objectives**

**By the end of this project you will:**

* Know fundamental concepts and techniques of the Artificial Neural Network and Convolution Neural Networks
* Gain a broad understanding of image data.
* Work with Sequential type of modeling
* Work with Keras capabilities
* Work with image processing techniques
* know how to build a web application using the Flask framework.

**Project Flow**

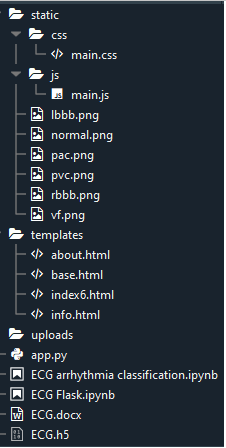
* User interacts with User interface to upload image
* Uploaded image is analyzed by the model which is integrated
* Once model analyses the uploaded image, the prediction is showcased on the UI

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

**Project Structure**

Create a Project folder which contains files as shown below



* We are building a Flask Application that needs  HTML pages stored in the templates folder and a python script app.py for serverside scripting
* we need the model which is saved and the saved model in this content is ECG.h5
* The static folder will contain js and CSS files.
* Whenever we upload an image to predict, those images are saved in the uploads folder.

**Prerequisites**

**To complete this project you should have the following software  and packages**

**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.Conda is an open-source, cross-platform,  package management system. Anaconda comes with so very nice tools like JupyterLab, Jupyter Notebook, QtConsole, Spyder, Glueviz, Orange, Rstudio, Visual Studio Code. For this project, we will be using Jupiter notebook and spyder

To install Anaconda navigator and to know how to use Jupyter Notebook a Spyder using Anaconda watch the video

To build Deep learning models you must require the following packages

**Tensor flow:**TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers can easily build and deploy ML powered applications.

**Keras :** Keras leverages various optimization techniques to make high level neural network API easier and more performant. It supports the following features:

* Consistent, simple and extensible API.
* Minimal structure - easy to achieve the result without any frills.
* It supports multiple platforms and backends.
* It is user friendly framework which runs on both CPU and GPU.
* Highly scalability of computation.

**Flask:** Web frame work used for building  Web applications

Watch the below video to Install the necessary Packages

### Prior Knowledge

**Supervised and unsupervised learning:**

Watch the below video to know about the types of machine learning

Dataset Collection

Artificial Intelligence is a data hunger technology, it depends heavily on data, without data, it is impossible for a machine to learn. It is the most crucial aspect that makes algorithm training possible. In Convolutional Neural Networks, as it deals with images, we need training and testing data set. It is the actual data set used to train the model for performing various actions. In this activity lets focus of gathering the dataset

### Download The Dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository, etc.

 The dataset used for this project is in this [Link](https://drive.google.com/file/d/16SUrk6lMaakmVf4axGNDub3joHl-XdBT/view?usp=sharing)  Please refer to the link to download the data set 

The dataset contains six classes:

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

https://ssl.gstatic.com/images/branding/product/1x/drive_2020q4_32dp.png

<https://drive.google.com/file/d/16SUrk6lMaakmVf4axGNDub3joHl-XdBT/view?usp=sharing>

**Image Preprocessing**

Image Pre-processing includes the following main tasks

* Import ImageDataGenerator Library.
* Configure ImageDataGenerator Class.
* Applying ImageDataGenerator functionality to the trainset and test set.

**Note:** The ImageDataGenerator accepts the original data, randomly transforms it, and returns only the new, transformed data.

To know more about the data generator class  click on this [**link**](https://www.pyimagesearch.com/2019/07/08/keras-imagedatagenerator-and-data-augmentation/)

### Import The 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.

Let us import the ImageDataGenerator class from Keras

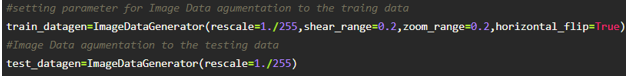
C:\Users\ELCOT\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\A525E6E2.tmp

**Configure ImageDataGenerator Class**

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

* Image shifts via the width\_shift\_range and height\_shift\_range arguments.
* Image flips via the horizontal\_flip and vertical\_flip arguments.
* Image rotates  via the rotation\_range argument
* Image brightness via the brightness\_range argument.
* Image zooms via the zoom\_range argument.

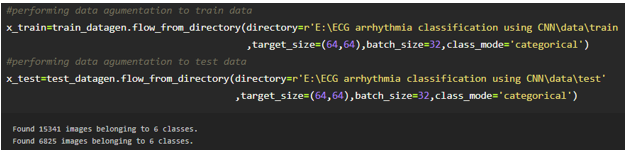
An instance of the ImageDataGenerator class can be constructed for train and test.



**Apply ImageDataGenerator Functionality To Trainset And Testset**

Let us apply ImageDataGenerator functionality to Train set and Test set 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}



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.

Arguments:

* directory: Directory where the data is located. If labels is "inferred", it should contain subdirectories, each containing images for a class. Otherwise, the directory structure is ignored.
* batch\_size: Size of the batches of data. Default: 32.
* target\_size: Size to resize images to after they are read from disk.
* class\_mode:
  + ‘int': means that the labels are encoded as integers (e.g. for sparse\_categorical\_crossentropy loss).
  + 'categorical' means that the labels are encoded as a categorical vector (e.g. for categorical\_crossentropy loss).
  + 'binary' means that the labels (there can be only 2) are encoded as float32 scalars with values 0 or 1 (e.g. for binary\_crossentropy).
  + None (no labels).

### Model Building

### We are ready with the augmented and pre-processed image data, Lets begin our model building, this activity includes the following steps

### Import the model building Libraries

### Initializing the model

### Adding CNN Layers

### Adding Hidden Layer

### Adding Output Layer

### Configure the Learning Process

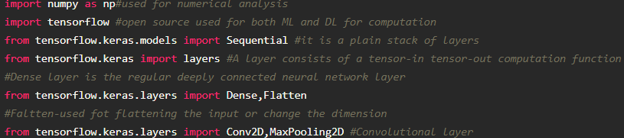
### Training and testing the model

### Saving the model

### To know more about model building please [click here](https://keras.io/about/)

### Import The Libraries

This is a very crucial step in our deep learning model building process. We have to define how our model will look and that requires.



**Initialize The Model**

Keras has 2 ways to define a neural network:

* Sequential
* 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.

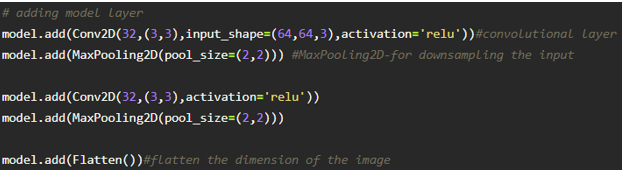
### Adding CNN Layers

For information regarding CNN Layers refer to the [**link**](link:%20https://victorzhou.com/blog/intro-to-cnns-part-1/)

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

The **Max pool layer** is used to down sample the input.

The **flatten layer** flattens the input.



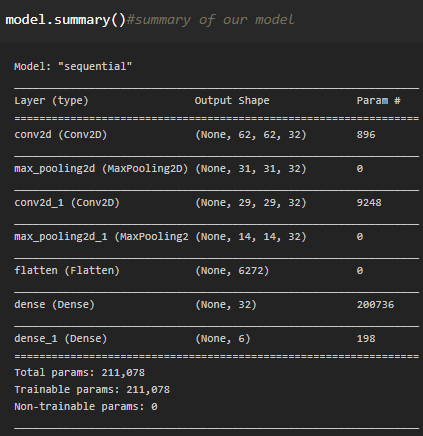
### Adding Dense Layers

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

C:\Users\ELCOT\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\213DDA98.tmp

 We have 6 neurons in op layer as we have considered 6 classes

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.



**Configure The Learning Process**

* 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.
* Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
* Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in the training process

C:\Users\ELCOT\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\14BAC564.tmp

**Train The Model**

Now, let us train our model with our image dataset. fit\_generator functions used to train a deep learning neural network

C:\Users\ELCOT\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\F835A512.tmp

steps\_per\_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and next epoch has started. We can calculate the value of     steps\_per\_epoch as the total number of samples in your train dataset divided by the batch size.

Epochs: an integer and number of epochs we want to train our model for.

**validation\_data**can be either:

* an inputs and targets list
* a generator
* an inputs, targets, and sample\_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.

**validation\_steps:**only if the **validation\_data** is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your test dataset divided by the validation batch size.

Note: it will take time for training your data based on epochs

### Save The Model

The model is saved with .h5 extension as follows 

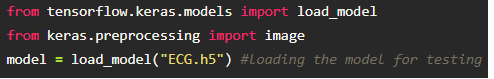
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data.

C:\Users\ELCOT\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\AC\INetCache\Content.MSO\DCC5D6F0.tmp

### Test The Model

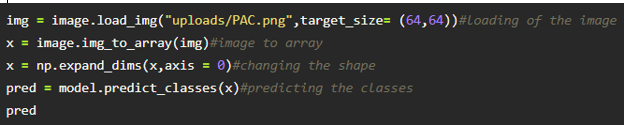
Open the other jupyter file and write the code mentioned below

Load necessary libraries, Load the saved model using load\_model

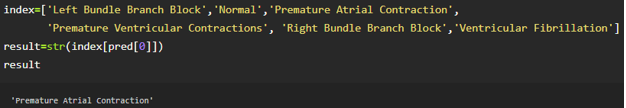


Taking an image as input and checking the results 

Note the target size should for the image that is should be the same as the target size that you have used for training



By using the model we are predicting the output for the given input images



The predicted class index name will be printed here.

**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 uses where he has uploaded an image. The uploaded image is given to the saved model and prediction is showcased on the UI.

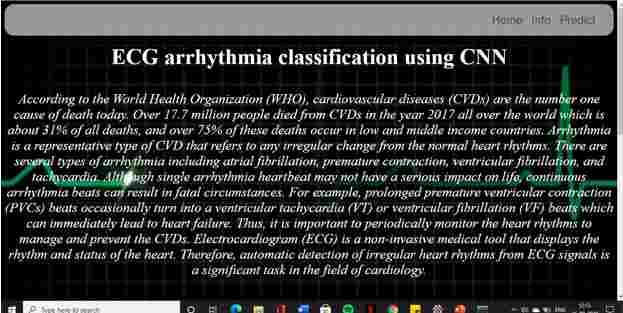
This section has the following tasks

* Building HTML Pages
* Building server-side script

**Create HTML Files**

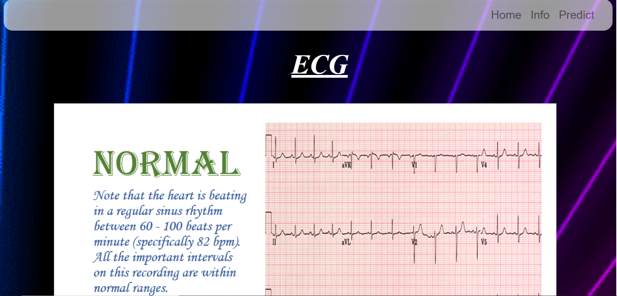
* We use HTML to create the front end part of the web page.
* Here, we created 4 html pages- about.html, base.html, index6.html, info.html.
* about.html displays the home page.
* Info.html displays all important details to be known about ECG.
* base.html and index6.html accept input from the user and predicts the values.

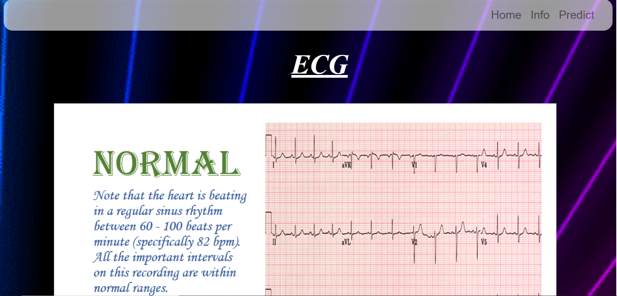
Your Home page looks like



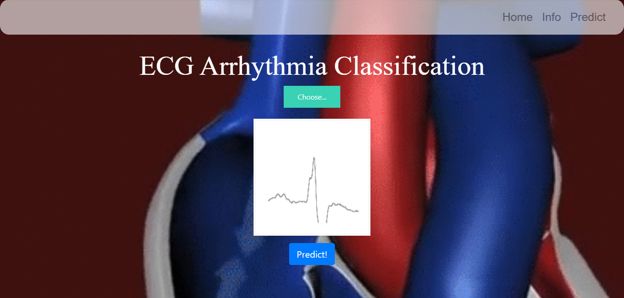
When the “Info” button is clicked, localhost redirects to “info.html”

Your info.html Page looks like:





Your prediction page will look like :

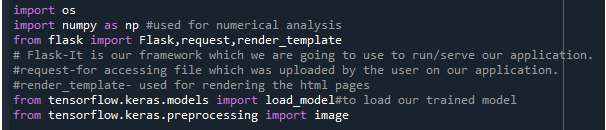


Upload the image and click on the Predict button to view the result on the “base.html” page on the localhost.

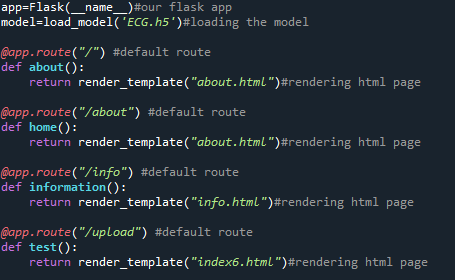
**Build Python Code**

* Let us build the flask file ‘app.py’ which is a web framework written in python for server-side scripting. Let’s see step by step procedure for building the backend application.
* The app starts running when the “\_\_name\_\_” constructor is called in main.
* render\_template is used to return HTML file.
* “GET” method is used to take input from the user.
* “POST” method is used to display the output to the user.

**Import the libraries**

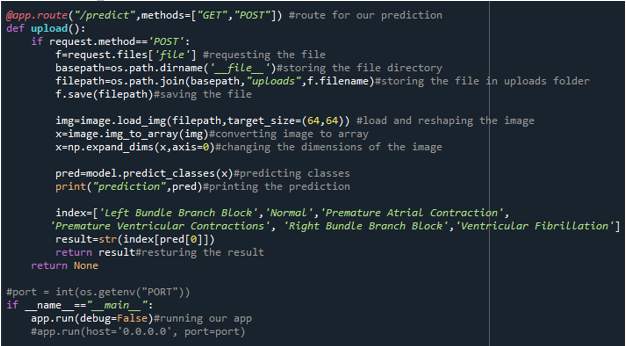


**Routing to the HTML Page**



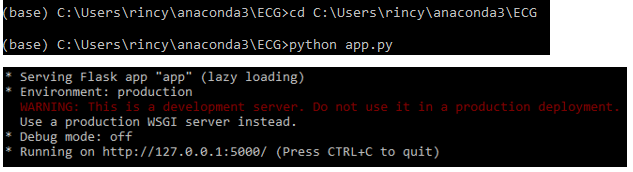
Showcasing prediction on UI

When the image is uploaded, it predicts the category of uploaded the image is either 'Left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature Ventricular Contractions', 'Right Bundle Branch Block', 'Ventricular Fibrillation'. If the image predicts value as 0, then it is displayed as “Left Bundle Branch”. Similarly, if the predicted value is 1, it displays “Normal” as output and so on.



### Run The APP

* Open anaconda prompt from the start menu
* Navigate to the folder where your python script is.
* Now type “python app.py” command
* Navigate to the localhost where you can view your web page
* Then it will run on localhost:5000
* Navigate to the localhost (http://127.0.0.1:5000/)where you can view your web page.



Upload an image and see the predicted result

### Train The Model On IBM

In this milestone, you will learn how to build Deep Learning Model Using the IBM cloud.

**Register For IBM Cloud**

**To complete this project you must have an IBM account**

**IBM Account:**

* Please click [**here**](https://cloud.ibm.com/registration)to register for IBM
* Please click [**here**](https://cloud.ibm.com/login)to log in to IBM Account.

**Watch the below video to register and login into your IBM account**

Train The Model On IBM Watson

**Please watch the below video to train the model on IBM  and integrate it with the flask Application.**

**Ideation Phase**

In this milestone you are expected to get started with the Ideation process.

**Literature Survey On The Selected Project & Information Gathering**

In this activity you are expected to gather/collect the relevant information on project use case, refer the existing solutions, technical papers, research publications etc.

**Prepare Empathy Map**

In this activity you are expected to prepare the empathy map canvas to capture the user Pains & Gains, Prepare list of problem statements.

**Ideation**

In this activity you are expected to list the ideas (at least 4 per each team member) by organizing the brainstorming session and prioritize the top 3 ideas based on the feasibility & importance.

**Project Design Phase -I**

From this milestone you will be starting the project design phase. You are expected to cover the activities given.

**Proposed Solution**

In this activity you are expected to prepare the proposed solution document, which includes the novelty, feasibility of idea, business model, social impact, scalability of solution, etc.

**Problem Solution Fit**

In this activity you are expected to prepare problem - solution fit document and submit for review.

**Solution Architecture**

In this activity you are expected to prepare solution architecture document and submit for review.

**Project Design Phase -II**

From this milestone you will be continue working on the project design phase. You are expected to cover the activities given.

**Customer Journey**

Prepare the customer journey maps to understand the user interactions & experiences with the application (entry to exit).

**Functional Requirement**

In this activity you are expected to prepare the functional requirement document.

**Data Flow Diagrams**

In this activity you are expected to prepare the data flow diagrams and submit for review.

**Technology Architecture**

In this activity you are expected to draw the technology architecture diagram.

**Project Planning Phase**

In this milestone you are expected to prepare milestones & tasks, sprint schedules.

**Prepare Milestone & Activity List**

In this activity you are expected to prepare the milestones & activity list of the project.

**Sprint Delivery Plan**

In this activity you are expected to prepare the sprint delivery plan.

**Project Development Phase**

In this milestone you will start the project development and expected to perform the coding & solutioning, acceptance testing, performance testing based as per the sprint and submit them.

**Project Development - Delivery Of Sprint-1**

In this activity you are expected to develop & submit the developed code by testing it.

**Project Development - Delivery Of Sprint-2**

In this activity you are expected to develop & submit the developed code by testing it.

**Project Development - Delivery Of Sprint-3**

In this activity you are expected to develop & submit the developed code by testing it.

**Project Development - Delivery Of Sprint-4**

In this activity you are expected to develop & submit the developed code by testing it.