

## SPRINT 2

### Classification of Arrhythmia by Using Deep Learning With 2-D ECG Spectral Image Representation

**Team ID:** PNT2022TMID39864

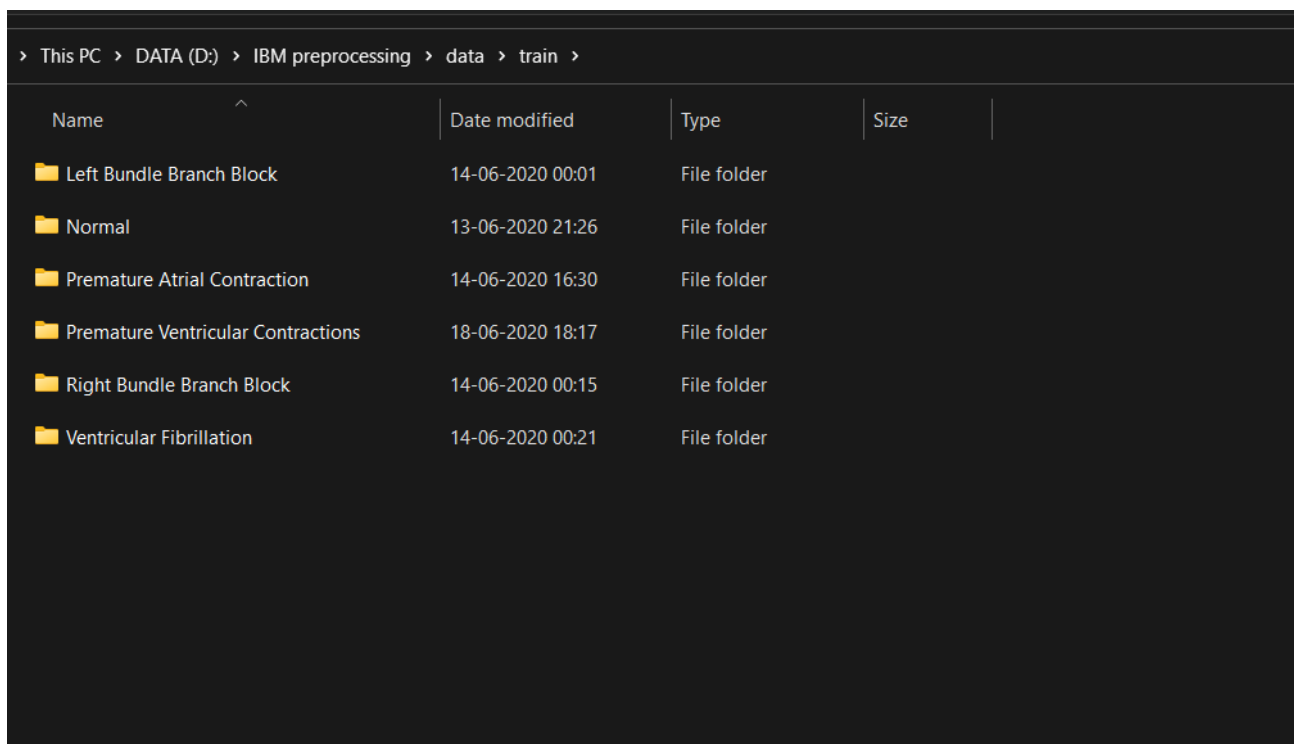
**Team Members:** BHAVANA ,BHAVANI ,JANANI , LAVANYA

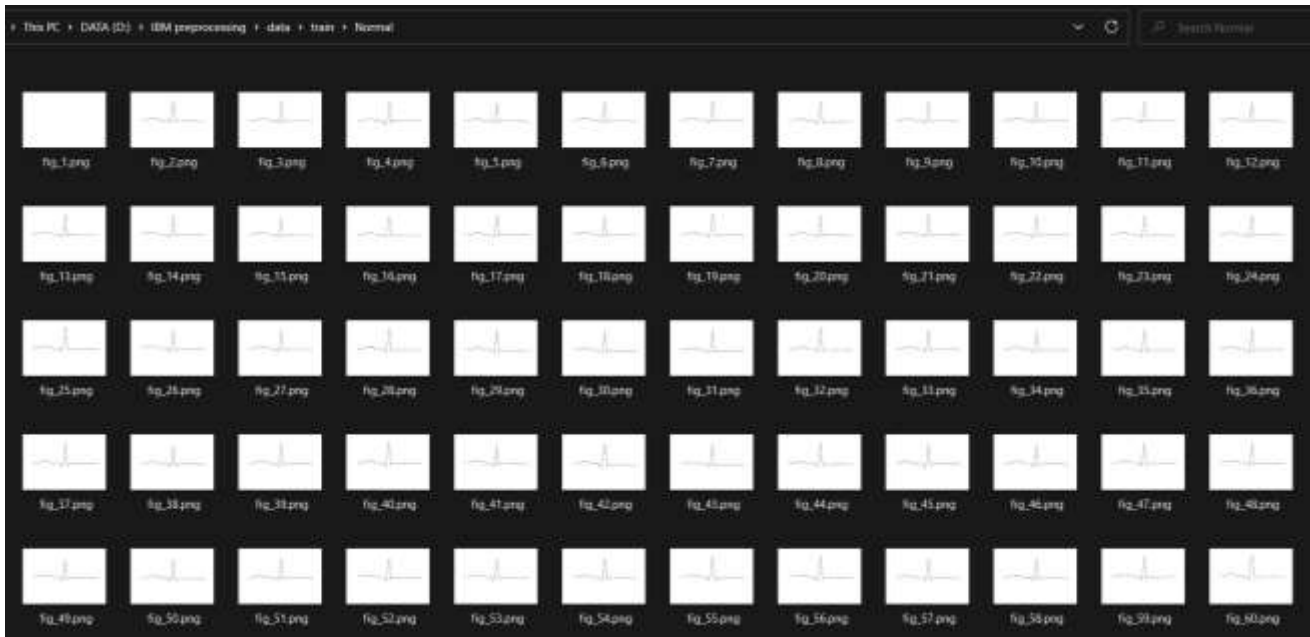
### Description of USN and Screenshots:

#### USN-3:

As a user, I want quality data to be collected for the purposes of training the model. Also, image processing methods must be employed to pre-process the dataset.

#### Screenshot:





## **Image Split:**

**Left Bundle Branch Block – 504 images**

**Normal – 7436 images**

**Premature Atrial Contraction – 2054 images**

**Premature Ventricular Contractions – 2759 images**

**Right Bundle Branch Block – 2239 images**

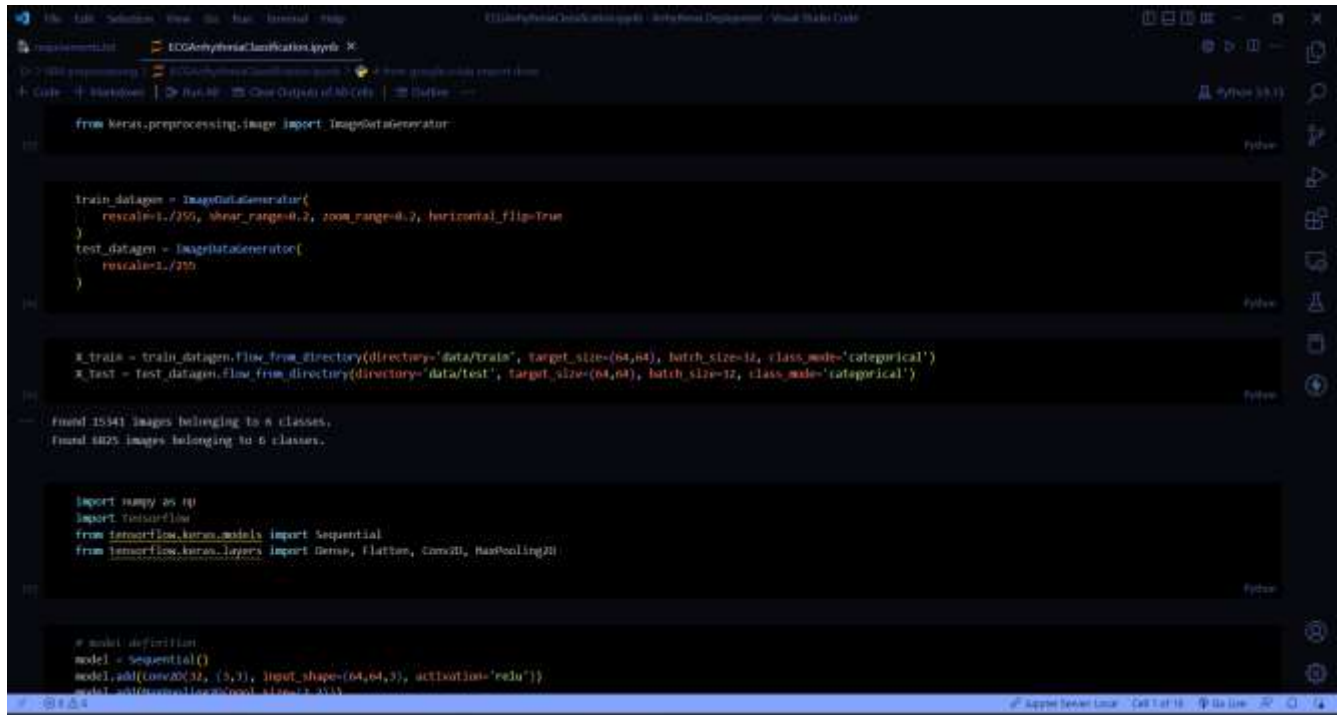
**Ventricular Fibrillation – 439 images**

For reducing skewness in the dataset, ImageDataGenerator class was used for both processing and handling with data imbalance.

## USN-4:

As a user, I want the ML model to be as accurate as possible.

## Screenshot:



```
from keras.preprocessing.image import ImageDataGenerator

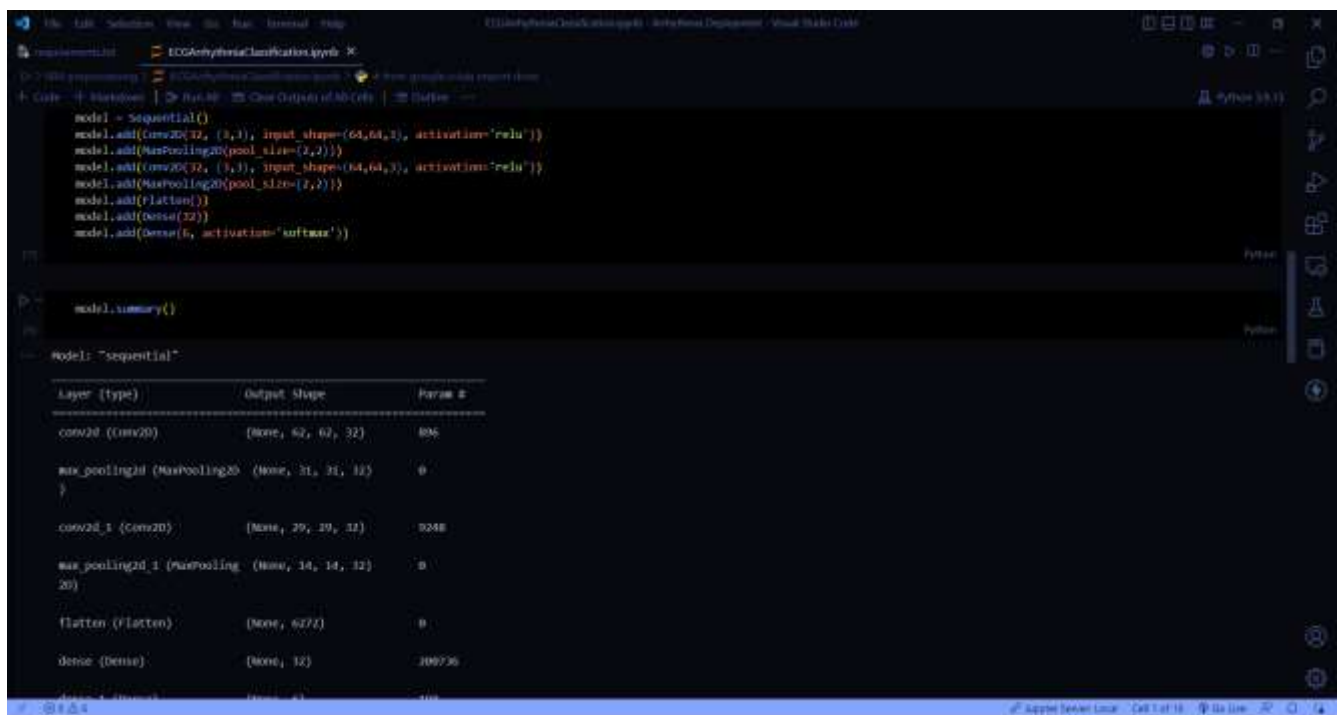
train_datagen = ImageDataGenerator(
    rescale=1./255, shear_range=0.2, zoom_range=0.2, horizontal_flip=True
)
test_datagen = ImageDataGenerator(
    rescale=1./255
)

x_train = train_datagen.flow_from_directory(directory='data/train', target_size=(64,64), batch_size=32, class_mode='categorical')
x_test = test_datagen.flow_from_directory(directory='data/test', target_size=(64,64), batch_size=32, class_mode='categorical')

found 15941 images belonging to 8 classes,
found 4825 images belonging to 6 classes.

import numpy as np
import tensorflow
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

# model definition
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape=(64,64,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
```



```
model = Sequential()
model.add(Conv2D(32, (3,3), input_shape=(64,64,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
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(32))
model.add(Dense(6, activation='softmax'))

model.summary()
```

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, 672)	0
dense (Dense)	(None, 32)	216736
dense_1 (Dense)	(None, 6)	206

# Model Architecture:

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, 32)	200736
dense_1 (Dense)	(None, 6)	198
=====		

Total params: 211,078

Trainable params: 211,078

Non-trainable params: 0

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

model.fit(X_train, steps_per_epoch=len(X_train), epochs=10, validation_data=X_test, validation_steps = len(X_test))
```

```
Epoch 1/10
400/400 [=====] - 37s 53ms/step - loss: 0.8551 - accuracy: 0.7090 - val_loss: 0.7419 - val_accuracy: 0.7771
Epoch 2/10
400/400 [=====] - 27s 55ms/step - loss: 0.3563 - accuracy: 0.8958 - val_loss: 0.4946 - val_accuracy: 0.8541
Epoch 3/10
400/400 [=====] - 25s 53ms/step - loss: 0.2785 - accuracy: 0.9185 - val_loss: 0.4826 - val_accuracy: 0.8722
Epoch 4/10
400/400 [=====] - 25s 52ms/step - loss: 0.2387 - accuracy: 0.9388 - val_loss: 0.4956 - val_accuracy: 0.8789
Epoch 5/10
400/400 [=====] - 25s 52ms/step - loss: 0.2188 - accuracy: 0.9392 - val_loss: 0.3386 - val_accuracy: 0.8753
Epoch 6/10
400/400 [=====] - 26s 55ms/step - loss: 0.1881 - accuracy: 0.9426 - val_loss: 0.3988 - val_accuracy: 0.8856
Epoch 7/10
400/400 [=====] - 25s 52ms/step - loss: 0.1618 - accuracy: 0.9589 - val_loss: 0.3494 - val_accuracy: 0.9017
Epoch 8/10
400/400 [=====] - 25s 53ms/step - loss: 0.1445 - accuracy: 0.9559 - val_loss: 0.2847 - val_accuracy: 0.9121
Epoch 9/10
400/400 [=====] - 26s 53ms/step - loss: 0.1354 - accuracy: 0.9576 - val_loss: 0.3341 - val_accuracy: 0.9017
Epoch 10/10
400/400 [=====] - 27s 56ms/step - loss: 0.1242 - accuracy: 0.9617 - val_loss: 0.3153 - val_accuracy: 0.9221

keras.callbacks.History at 0x7f942f328208
```

```
File Edit Selection View Go Run Terminal Help
ECGAnomalyClassification.ipynb - Anomaly Detection - Visual Studio Code

requirements.txt ECGAnomalyClassification.ipynb K
D:\> pip install tensorflow==2.10.0 tensorflow.keras==2.10.0 keras==3.0.0 keras.preprocessing==2.10.0 model==0.0.0
+ Code + Run and Debug + Check Output of All Cells + Outline

model.save('ECG.h5')
Python

def predict_image(img):
    from tensorflow.keras.models import load_model
    from keras.preprocessing import image
    model = load_model('ECG.h5')

    img = image.load_img('uploads/PAC.jpg', target_size=(64,64))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    pred = model.predict_classes(x)

    pred
Python

index = ['left Bundle Branch Block', 'Normal', 'Premature Atrial Contraction', 'Premature Ventricular Contraction', 'Right Bundle Branch Block', 'Ventricular Fibrillation']
result = str(index[pred[0]])

result
Python
```