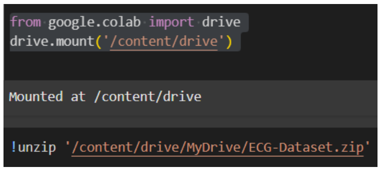
Data Collection and Preprocessing Phase

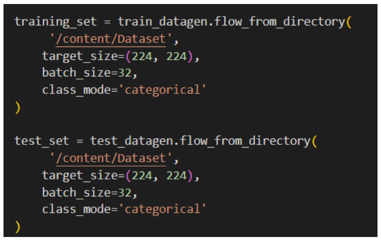
Date	24 SEPTEMBER 2024
Team ID	SWTID1727151090
Project Title	Classification of Arrhythmia by Using Deep Learning with 2-D ECG Spectral Image Representation
Maximum Marks	6 Marks

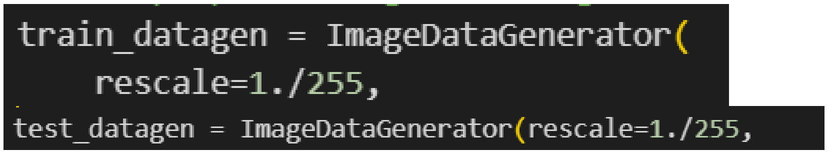
Preprocessing Template

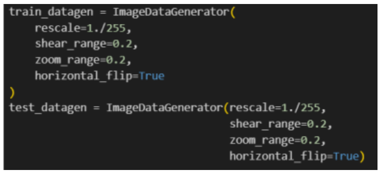
The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

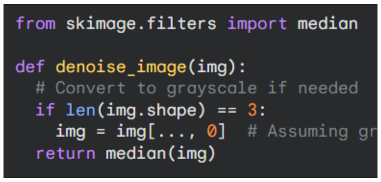
Section	**Description**
Data Overview	The dataset contains test and train elements. These test and train elements each contain data of ➢ Left Bundle Branch Block ➢ Normal ➢ Premature Atrial Contraction ➢ Premature Ventricular Contractions ➢ Right Bundle Branch Block ➢ Ventricular Fibrillation
Resizing	Resize images to a specified target size.
Normalization	Normalize pixel values to a specific range.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.

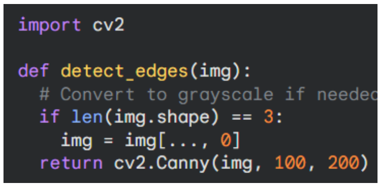
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.
Color Space Conversion	Convert images from one color space to another.
Image Cropping	Crop images to focus on the regions containing objects of interest.
Batch Normalization	Apply batch normalization to the input of each layer in the neural network.
Data Preprocessing Code Screenshots	
Loading Data	 from google.colab import drive drive.mount('/content/drive') Mounted at /content/drive !unzip '/content/drive/MyDrive/ECG-Dataset.zip'

Resizing	
	training_set = train_datagen.flow_from_directory(
	    '/content/Dataset',
	    target_size=(224, 224),
	    batch_size=32,
	    class_mode='categorical'
	)
	
	test_set = test_datagen.flow_from_directory(
	    '/content/Dataset',
	    target_size=(224, 224),
	    batch_size=32,
	    class_mode='categorical'
	)

Normalization	
	train_datagen = ImageDataGenerator(
	    rescale=1./255,
	test_datagen = ImageDataGenerator(rescale=1./255,

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

Denoising	
	from skimage.filters import median
	
	def denoise_image(img):
	  # Convert to grayscale if needed
	  if len(img.shape) == 3:
	    img = img[..., 0] # Assuming gr
	  return median(img)

Edge Detection	
	import cv2
	
	def detect_edges(img):
	  # Convert to grayscale if needec
	  if len(img.shape) == 3:
	    img = img[..., 0]
	  return cv2.Canny(img, 100, 200)

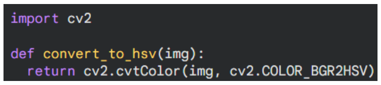
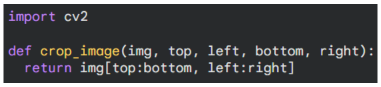
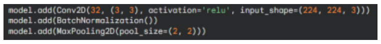
Color Space Conversion	
	import cv2
	
	def convert_to_hsv(img):
	  return cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

Image Cropping	
	import cv2
	
	def crop_image(img, top, left, bottom, right):
	  return img[top:bottom, left:right]

Batch Normalization	
	model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
	model.add(BatchNormalization())
	model.add(MaxPooling2D(pool_size=(2, 2)))