

# Heart sound classification

<sup>1</sup>G. Gowtham, <sup>2</sup>V. Shravan Kumar, <sup>3</sup>N. Jaya Sai Kumar Reddy, <sup>4</sup>B. Manoj Kumar,  
<sup>5</sup>Ms. S. Nivethalakshmi

<sup>1,2,3,4</sup>Student, <sup>5</sup>Assistant Professor  
Bharath University

**Abstract-** Automatic type of four forms of beats Normal (N), premature ventricular beat (PVC), supraventricular premature or ectopic beat (SVPB), and ventricular-ordinary fusion (FUSION) is carried out the use of the Multiclass Support Vector System (MSVM). . Urban Algorithm Support Vector Machine (CSVM). Pre-processing and R top detection is now achieved inside the WFDB software package deal, which reads the annotation and R (pinnacle) neighborhoods. The feature R (top) is used to detect the peaks in the diverse waveforms collectively with P and T extracted from the ECG. Discrete cosine and discrete transforms or Discrete Fourier Transforms (DFTs) had been used to extract the feature and discount measurement of the input vector classifier. The motive of the study was to enhance the multi-magnificence SVM by way of using an extension of the traditional SVM set of rules for complex spaces as a manner to degree 4 forms of coronary heart beats simultaneously. This approach can also don't forget the adjustment of segments within a character. Accuracy among 86% and 94% is considered for MSVM and CSVM lessons respectively. This tool indicates that the proposed pulse classifier may be a completely dependable and beneficial device for detecting arrhythmia conditions and robotically producing the class.

## INTRODUCTION:

ECGs offer a image illustration of the electric activity of the heart muscle. This take a look at evaluates ECG class the usage of preprocessing, characteristic extraction, and choice tactics, and the class of ECG contracts the usage of MSVM and CSVM. The DCT and DST coefficients had been used to extract the capabilities contained within the MSVM classifier vector, whereas the DFT coefficients have been used to generate the input vector for the CSVM classifier. Classifying MSVM and CSVM was used to differentiate between the four kinds of ECG arrhythmias using DCT, DST and DFT coefficients as the characteristic vector within the classifier vector. A Sequence Minimal Optimization (SMO) technique is used to set up the CSVM and reply to the complicated hyperplane environment.

## LITERATURE SURVEY:

### 1. ECG analysis and classification using CSVM, MSVM and SIMCA classifiers

Jannah, Najlaa, 2017

Certain ECG classification has the capability to introduce better detection techniques and improved accuracy in the analysis of arrhythmias, consequently improving the first-class of care. This dissertation explored the use of category algorithms, CSVM and SIMCA, and their overall performance in reading ECG beats. The mission aimed to introduce a brand new manner to interactively support affected person care in and out of the health facility, and to develop new category algorithms for arrhythmia detection and analysis. Waveform detection (P-QRS-T) became executed the usage of the WFDB software package and waveforms with extraordinary resolutions.

**Logging:** less accuracy rate

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

Amin Ullah, Syed Muhammad Anwar ,Muhammad Bilal and Raja Majid Mehmood, 2020

The electrocardiogram (ECG) is one of the most extensively used indicators in the analysis and analysis of cardiovascular sickness (CVD). ECG indicators can locate abnormal heart rhythms, usually referred to as arrhythmias. Careful have a look at of ECG signals is essential for correct diagnosis of acute and continual coronary heart disease patients. In this examine, we recommend a - dimensional (2-D) convolutional neural community (CNN) model for analyzing ECG alerts into eight instructions.

**Logging:** They used CNN classification only

### 3. Cardiac Arrhythmia Detection and Classification From ECG Signals Using XGBoost Classifier Saroj Kumar Pandeyz, Rekh Ram Janghel, Vaibhav Gupta, 13 August 2021

In this text, we suggest an green method for reading electrocardiogram (ECG) signals using the XG Boost classifier. ECG signals undergo 4 steps: facts acquisition, noise filtering, characteristic extraction, and segmentation. In step one, a records set is collected from the MIT-BIH arrhythmia database. In the second one level, the noise is removed the use of a fuzzy baseline elimination clear out. The next step makes use of forty five descriptors from the 4 foremost capabilities which have accomplished nicely in previous paintings, specifically waves, higher order information (HOS), morphological descriptors and R-R durations.

**Logging:** Used xg boost classifier

### 4. Arrhythmia detection using multi-lead ECG spectra and Complex Support Vector Machine Classifiers

Najlaa Jannah,\*, Sillas Hadjiloucasb , Jameel Al-Malki

Electrocardiograms (ECGs) are widely used to diagnose cardiac arrhythmias. This paper explores using system gaining knowledge of class algorithms for ECG analysis and arrhythmia detection. Four forms of beats: normal (N), premature ventricular contraction (PVC), untimely atrial contraction (APC), and proper package branch block (RBBB) are simultaneously offered through a complex vector machine (CSVM) classifier.

**Logging:** used only CSVM

## 5. "Complex Support Vector Machines for Regression and Quaternary Classification"

**P. Bouboulis, S. Theodoridis, C. Mavroforakis and L. Evaggelatos-Dalla 2014**

Automatic category of the four varieties of beats Normal (N), premature ventricular contraction (PVC), supraventricular untimely or ectopic contraction (SVPB), and ventricular-regular fusion (FUSION) is implemented the usage of a multiclass support vector gadget (MSVM). Complicated vector device (CSVM) set of rules.

**Logging:** They use only DFT

## 6. "On the application of optimal wavelet filter banks for ECG signal classification"

**S. Hadjiloucas, N. Jannah, F. Hwang and R. K. H. Galvão, 2014**

Future paintings on wavelet preprocessing to similarly compress the type input space by means of producing wavelets in better order moment standards [3] in addition to an method to make bigger CSVM input and spaces to arbitrary size the use of SVM Clifford algebra [4] may be mentioned. . To the interview.

**Logging:** used only CSVM less accuracy

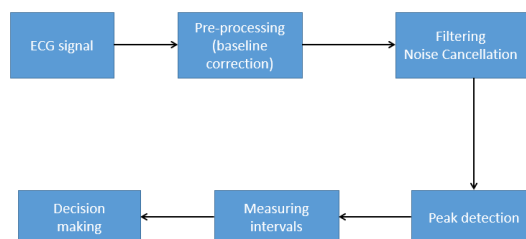
## EXISTING SYSTEM:

- Ischemic coronary heart disease (CHD), additionally known as cardiovascular disease (CVD), consequences from inadequate blood supply to the coronary heart organ. CAD is related to many sorts of arrhythmias, usually described as dissociated, slow, or rapid. Acute myocardial infarction (MI) can be fatal, but mortality commonly relies upon on the severity of the arrhythmia.
- Delayed mortality from those cardiac complications requires early analysis of coronary heart sickness, before it progresses to acute MI and ends in dying. The most common approach of diagnosing specific varieties of arrhythmia is the electrocardiogram (ECG).
- Careful ECG analysis is receiving plenty interest from researchers for correct and powerful analysis of arrhythmia and critical heart conditions. Traditional diagnostics consists of the recording of an ECG on the patient's bedside and the presence of a doctor to analyze and circumstance the patient.
- However, these strategies are time-eating, and in extreme cardiac cases, time is of the essence in diagnosing this situation.

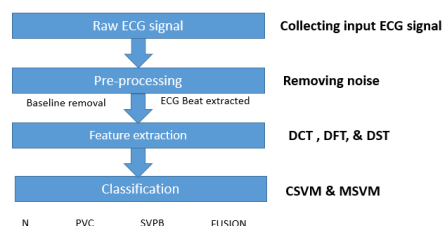
## PROPOSED SYSTEM

- The proposed paintings provides a beneficial methodology for multi-lead ECG evaluation and arrhythmia category based totally on CSVM category after a pre-described DFT signal. The examine used four forms of abbreviations derived from the INCARTDB database, although different databases with nicely-annotated snap shots may also be used.
- Classification of unmarried hits turned into accomplished. SMO become used to calculate hyperplane parameters and CSVM thresholds all through the education section. The results show that the proposed heart fee classifier is very reliable and may be used for automated detection and classification of arrhythmia conditions. Using CSVM, you may quickly insert a category IV characteristic by means of splitting it into separate SVM capabilities.
- The CSVM algorithm tested the high-quality overall performance by showing that ECG lines have been extracted in multidimensional area the use of complicated hyperplanes. Using CSVM algorithm suggests huge improvement in multiclass category as compared to MSVM. The average accuracy the use of ECG pulses extracted from the six precords and limbs became 98.25%.
- In terms of the use of CSVM, the principle issue is that best four types of ECG pulses can be certain, which gives a motivation for the similarly development of ECG type evaluation based on Clifford algebra.

## BLOCK DIAGRAM:



## FLOWCHART:



## Introduction

In this educational, we will display you how to create an photo using Python language. We aren't restricted to at least one library or framework; however, there is one that we can use most often - the Open CV library. We'll begin with a bit talk about image processing after which circulate directly to various packages/situations where picture processing can are available in accessible. So permit's get began!

## What is Image Processing?

It is crucial to know precisely what photo processing is and what its function is within the huge photograph earlier than entering into its standards. Image processing is maximum generally referred to as "digital imaging" and the region in which it's far maximum generally used is "computer vision". Don't get confused - we'll speak approximately both terms and how they relate. Both photo processing algorithms and laptop imaginative and prescient (CV) algorithms take an photo as enter; but, in image processing, the output is likewise an image, even as in pc imaginative and prescient, the output may be some capabilities/records about the picture.

## Why do we need it?

The information we acquire or generate is in the main uncooked records, no longer appropriate for direct use in programs for a number of reasons. It is consequently vital to first examine it, entire the important pre-processing, after which use it.

Suppose we're trying to create a cat classifier. Our program will take an picture as enter and then inform us if the photo contains a cat or now not. The first step in creating this class is to gather hundreds of pix of cats. One of the commonplace troubles is that each one the cleaned images will now not have the same length/dimensions, so we need to resize/preprocess them to a widespread length earlier than passing them to the model.

This is one of the many motives why photo processing is essential to any computer vision application.

## Prerequisites

Before we pass any similarly, allow's speak what we want to understand which will follow this manual effortlessly. First, you need to have primary knowledge of any programming language. Second, you want to realize what machine studying is and the way it works as we will use a few gadget learning algorithms to describe the process in this newsletter. As an advantage, it'll be useful if you have any knowledge of OpenCV or simple expertise of it before intending with this tutorial. But that is optional.

One issue you want to know virtually with a purpose to comply with this educational is how the photo is represented in reminiscence. Each image is represented by using pixels, i.E. Matrix of pixel values. For a grayscale picture, the pixel values variety from zero to 255 and constitute the depth of that pixel. For example, when you have a 20 x 20 photo, it will be represented through a 20 x 20 matrix (four hundred pixel values in general).

If we're talking approximately the coloration of the photograph, you need to understand that it has three channels - crimson, inexperienced and blue (RGB). Therefore there may be three such matrices for one picture.

## Installation

Note: Since we are going to use OpenCV via Python, it is an implicit requirement that you already have Python (version 3) already installed on your workstation.

### Windows

```
$ pip install opencv-python
```

### MacOS

```
$ brew install opencv3 --with-contrib --with-python3
```

### Linux

```
$ sudo apt-get install libopencv-dev python-opencv
```

To check if your installation was successful or not, run the following command in either a Python shell or your command prompt:

```
import cv2
```

## Some Basics You Should Know

Before we continue to use photo processing within the software, it's miles critical to have an concept of what falls into this class and the way to carry out the ones operations. These operations could be used with others in our applications later. That's how you start.

For this article we'll be using the following image:



**Note** The photo in this newsletter has been scaled for show, however the unique length we are the usage of is round 1180x786.

You may also be aware that the photograph is now in color, because of this that it is represented by means of 3 shade channels ie. Crimson, green and blue. We will convert the image to grayscale in addition to separate the image into separate channels the usage of the code underneath.

### Finding Image Details

When loading an photograph with the print feature, we are able to get some simple homes, consisting of the wide variety of pics and dimensions:

```
import cv2

img = cv2.imread('rose.jpg')

print("Image Properties")

print("- Number of Pixels: " + str(img.size))

print("- Shape/Dimensions: " + str(img.shape))
```

### Output:

```
Image Properties
- Number of Pixels: 2782440
- Shape/Dimensions: (1180, 786, 3)
```

### Splitting an Image into Individual Channels

Now we'll split the image in to its red, green, and blue components using OpenCV and display them:

```
from google.colab.patches import cv2_imshow

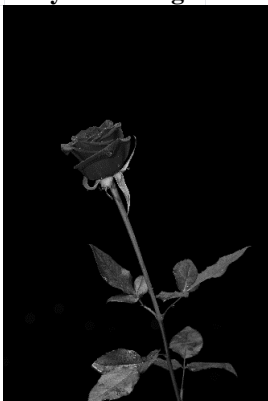
blue, green, red = cv2.split(img) # Split the image into its channels

img_gs = cv2.imread('rose.jpg', cv2.IMREAD_GRAYSCALE) # Convert image to grayscale

cv2_imshow(red) # Display the red channel in the image
cv2_imshow(blue) # Display the red channel in the image
cv2_imshow(green) # Display the red channel in the image
cv2_imshow(img_gs) # Display the grayscale version of image
```

For brevity, we'll just show the grayscale image.

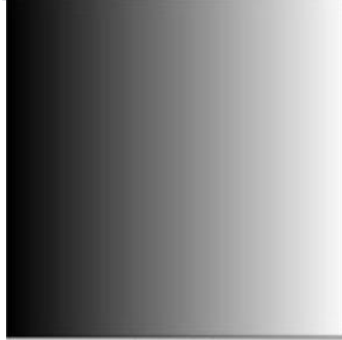
### Grayscale Image:



### Image Thresholding

The idea of a threshold is quite simple. As cited above for photo illustration, pixel values can be from zero to 255. Let's say we need to convert an picture to a binary picture, i.E. To give a pixel value of zero or 1. To try this we can perform thresholds. For instance, if the Threshold (T) is 125, then all elements with values extra than 125 may be assigned the cost 1, and all factors with values

much less than or identical to, might be assigned the fee zero. Let us apprehend higher through the code. **Image used for Thresholding:**



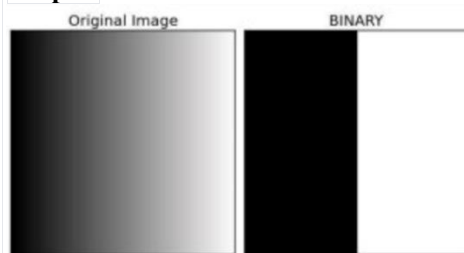
```
import cv2

# Read image
img = cv2.imread('image.png')

# Perform binary thresholding on the image with T = 125
threshold = cv2.threshold(img, 125, 255, cv2.THRESH_BINARY)

cv2.imshow(threshold)
```

#### Output:



As you could see, the two regions inside the resulting photo i.e. They are positioned within the black region (pixel value zero) and in the white place (pixel value 1). It turns out that we positioned the threshold proper inside the middle of the photo, so the black and white items are separated there.

#### Applications

#1: remove noise from the picture

Now which you have a wellknown idea of what a process picture is and what it's used for, allow's pass and find out about some specific applications.

In most cases, the raw information accrued incorporates noise, i.e. Useless capabilities that make the picture hard to perceive. Although these snap shots may be without delay used for characteristic extraction, the accuracy of the algorithm will go through greatly. This is because picture processing is carried out to the picture before it's miles exceeded to the accuracy algorithm.

There are many exclusive kinds of noise, which includes Gaussian noise, salt and pepper noise, and many others. We can dispose of this noise from the photo with the aid of making use of a filter that removes this noise, or at least reduces its impact. There are many filter out options, each with extraordinary strengths and therefore pleasant appropriate for a particular form of noise.

To get this proper, we're going to upload "salt and pepper" noise to the grayscale version of the rose image we looked at above, then try and put off that noise from our image by using the use of extraordinary filters and see what works. The quality . Appropriate for this kind.

```
import numpy as np

# Adding salt & pepper noise to an image
def salt_pepper(prob):
    # Extract image dimensions
    row, col = img_gs.shape

    # Declare salt & pepper noise ratio
    s_vs_p = 0.5
    output = np.copy(img_gs)

    # Apply salt noise on each pixel individually
    num_salt = np.ceil(prob * img_gs.size * s_vs_p)
```

```

coords = [np.random.randint(0, i - 1, 1), int(num_salt)]
for i in img_gs.shape:
    output[coords] = 1

# Apply pepper noise on each pixel individually
num_pepper = np.ceil(prob * img_gs.size * (1 - s_vs_p))
coords = [np.random.randint(0, i - 1, 1), int(num_pepper)]
for i in img_gs.shape:
    output[coords] = 0
cv2_imshow(output)

return output

# Call salt & pepper function with probability = 0.5
# on the grayscale image of rose
sp_05 = salt_pepper(0.5, 1)

# Store the resultant image as 'sp_05.jpg'
cv2.imwrite('sp_05.jpg', sp_05)

```

Well, we have introduced some noise to our rose image, which now looks like this:

#### Noisy Image:



Now permit's apply numerous filters to it and observe your observations, i.E. How a whole lot every filter reduces the noise.

#### Arithmetic Filter with Sharpening Kernel

# Create our sharpening kernel, the sum of all values must equal to one for uniformity

```

kernel_sharpening = np.array([[1, 1, 1, 1, 1],
                                [-1, 1, 9, 1, -1],
                                [-1, 1, 1, 1, 1]])

```

# Applying the sharpening kernel to the grayscale image & displaying it,

```

print("\n\n--- Effects on S&P Noise Image with Probability 0.5 ---\n\n")

```

# Applying filter on image with salt & pepper noise

```

sharpened_img = cv2.filter2D(sp_05, 1, kernel_sharpening)

cv2_imshow(sharpened_img)

```

An photograph from the application of an arithmetic clear out to an picture with salt and pepper noise is shown beneath. When compared to the unique grey photo, we can see that the photo is too bright and does now not spotlight the rose. Hence we can conclude that the salt and pepper arithmetic filter out cannot cast off the noise.

#### Arithmetic Filter Output:



### Midpoint Filter

```
from scipy.ndimage import maximum_filter, minimum_filter
```

```
def midpoint(img):
    maxf = maximum_filter(img, kernel_size=3)
    minf = minimum_filter(img, kernel_size=3)
    midpoint = (maxf + minf) / 2
    cv2.imshow('midpoint', midpoint)
```

```
print("\n\n---Effects on S&P Noise Image with Probability 0.5---\n\n")
```

```
midpoint(sp_05)
```

The resulting image from making use of a medium filter out to an picture with salt and pepper noise is shown below. When compared to the authentic grayscale photograph, we see that, as above the center mode, the picture is simply too bright; however, it may carry bright spots at the rose. Hence it could be stated that it's miles a better choice than the arithmetic filter out, but still the original photograph isn't completely restored.

### Midpoint Filter Output:



### Contraharmonic Mean Filter

**Note:** Implementations of these threads can effortlessly be determined on-line, and the way exactly they paintings is beyond the scope of this manual. We will have a look at programs from an abstract/standard angle.

```
def contraharmonic_mean(img, size, Q):
    num = np.power(img, Q + 1)
    denom = np.power(img, Q)
    kernel = np.full(size, 1.0)
    result = cv2.filter2D(num, -1, kernel) / cv2.filter2D(denom, -1, kernel)
    return result
```

```
print("\n\n--- Effects on S&P Noise Image with Probability 0.5 ---\n\n")
```

```
cv2.imshow('contraharmonic_mean(sp_05, kernel_size=3, Q=0.5)')
```

An photo from the software of a median counterharmonic filter to an photo with salt and pepper noise is proven below. When compared to the unique grayscale picture, we see that it reproduces nearly the same picture as the authentic. The intensity/brightness of the field is the same and highlights in rose. From this we can conclude that the average counterharmonic filter may be very powerful in preventing the sound of salt and pepper.



**Contraharmonic Mean Filter Output:**

Now that we've got determined the great clear out to restore the unique photograph from the noise, we can flow on to the subsequent utility.

**#2: Detect edges with the Canny Edge Detector**

The rose photograph she used only has a steady history, i.E. Black, so we are able to use another image for this application to better show the skills of the algorithm. The purpose is that if the concern is steady, it makes detecting the acute case quite clean, and we do not need it.

We pointed out the cat classifier earlier on this tutorial, permit's take this example and notice how picture processing performs an necessary position in it.

In a type algorithm, the picture is first scanned for "gadgets", i.E. Whilst you input an photograph, the algorithm reveals the entirety in that picture and compares them to the characteristics of the object you are attempting to find. When classifying a cat, it compares all items discovered within the photo to the features of the cat picture, and if a suit is found, it tells the input image to incorporate a cat.

When you use the cat classification example, it's miles handiest fair to use the cat picture inside the following. We will use the image below:

**Image used for Edge Detection:**

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
```

```
# Declaring the output graph's size
```

```
plt.figure(figsize=(16, 16))
```

```
# Convert image to grayscale
```

```
img_gs = cv2.imread('cat.jpg', cv2.IMREAD_GRAYSCALE)
```

```
cv2.imwrite('gs.jpg', img_gs)
```

```
# Apply canny edge detector algorithm on the image to find edges
```

```
edges = cv2.Canny(img_gs, 100, 200)
```

```
# Plot the original image against the edges
```

```
plt.subplot(121), plt.imshow(img_gs)
```

```
plt.title('Original Gray Scale Image')
```

```
plt.subplot(122), plt.imshow(edges)
```

```
plt.title('Edge Image')
```

```
# Display the two images
```



plt.show()

**Edge Detection Output:**

As you can see, the part of the photo that incorporates the object, which in this situation is the cat, is divided with the aid of factors in the mouth detection. Now you need to be wondering what's Canny Edge Detector and how did it paintings; now allow us to speak To understand the above, we need to address three key steps. First, it does the noise reduction of the picture in a comparable manner to what become said before. Second, it makes use of the primary spinoff at each pixel to discover the edges. The purpose at the back of this is that at the factor where it's far at the threshold, the change is extremely sharp, which reasons the cost of the first spinoff to spike, making that pixel the "area pixel".

In the cease, the hysteresis threshold physical activities; We have said above that there may be a spike within the value of the primary spinoff to the margin, however we have no longer defined "how excessive" the spike must be so as for it to be indicated as a margin - that is called the "threshold". ! Earlier on this tutorial, we discussed what a simple threshold is. Threshold hysteresis is an improvement, it uses thresholds instead of one. The motive for that is that if the edge price is too excessive, we can also pass over some actual edges (real negatives), and if the price is just too low, we can have many points marked as edges that do not without a doubt have edges (false positives). . . One threshold is about high, the other is diminished. All points above the "excessive threshold" are marked, then all points above the lower threshold but below the excessive threshold are evaluated; factors near or adjoining to factors marked as edges also are marked edges, and the relaxation are rejected.

These are the main concepts / techniques that the Canny Edge Detector algorithm uses to stumble on edges in an image.

**CONCLUSION**

In this newsletter, we discovered the way to set up OpenCV, the most popular Python image processing library, on different systems including Windows, MacOS, and Linux, and how to successfully install it.

We mentioned what picture processing is and its use inside the field of system studying for laptop imaginative and prescient. We talked about a few not unusual sorts of noise and how we can dispose of it from our photographs using diverse filters earlier than the use of the snap shots in our applications.

In addition, we learned how photo processing plays an integral part in huge-scale packages, consisting of object detection or class. Note that this article is handiest the end of the iceberg, and Digital Image Processing has lots more to offer that can not be included in one manual. Reading this can will let you delve deeper and find out about different superior principles related to photo processing.

**REFERENCES:**

1. Übeyli ED. "Combining recurrent neural networks with eigenvector methods for classification of ECG beats". Digital Signal Processing 2009; 19(2): 320–329.
2. Froese T, Hadjiloucas S, Galvão RKH, et al. " Comparison of extrasystolic ECG signal classifiers using discrete wavelet transforms". Pattern Recognition Letters 2006; 27(5): 393–407.
3. Wang J-S, Chiang W-C, Hsu Y-L, et al. " ECG arrhythmia classification using a probabilistic neural network with a feature reduction method". Neurocomputing 2013; 116: 38–45.
4. Gacek A. "Preprocessing and analysis of ECG signals - A self-organizing maps approach". Expert Systems with Applications 2011; 38(7): 9008–9013.
5. Yu SN, and Chou KT. " Selection of significant independent components for ECG beat classification". Expert Systems with Applications 2009; 36(2): 2088–2096.
6. Jannah N, Hadjiloucas S. "A Comparison between ECG Beat Classifiers Using Multiclass SVM and SIMCA with Time Domain PCA Feature Reduction". In: Proceedings - 2017 UKSim-AMSS 19th International Conference on Modelling and Simulation, UKSim 2017. Cambridge, 2017, pp. 126–131.
7. Rahhal MM Al, Bazi Y, Alhichri H, et al. " Deep learning approach for active classification of electrocardiogram signals". Information Sciences 2016; 345: 340–354.
8. Kanani P, Padole M. ECG Heartbeat Arrhythmia Classification Using Time-Series Augmented Signals and Deep Learning Approach. Procedia Computer Science 2020; 171: 524–531.
9. Kanani P, Padole M. " ECG Heartbeat Arrhythmia Classification Using Time-Series Augmented Signals and Deep Learning Approach". Procedia Computer Science 2020; 171: 524–531.
10. Latif G, Al Anezi FY, Zikria M, et al. " EEG-ECG Signals Classification for Arrhythmia Detection using Decision Trees". In: Proceedings of the Fourth International Conference on Inventive Systems and Control (ICISC 2020). Coimbatore, 2020, pp. 192–196.
11. Wu J, Li F, Chen Z, et al. " Patient-specific ECG classification with integrated long short-term memory and convolutional neural networks". IEICE Transactions on Information and Systems 2020; E103D(5): 1153–1163.
12. Monasterio V, Laguna P, and Martinez JP. "Multilead estimation of T-wave alternans in the ECG using principal component analysis". In: 16th European Signal Processing Conference (EUSIPCO 2008). 2008, pp. 1–5.

13. Goldberger AL, Amaral LAN, Glass L, et al. "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals". *Circulation* 2000; 101(23): e215-e220.
14. Zhang J, Tian J, Cao Y, et al. "Deep time–frequency representation and progressive decision fusion for ECG classification". *KnowledgeBased Systems*.2020;190:105402.
15. Bouboulis P, Theodoridis S, Mavroforakis C, et al. "Complex Support Vector Machines for Regression and Quaternary Classification". *IEEE Transactions on Neural Networks and Learning Systems* 2015; 26(6): 1260–1274.