Classification of PAF using ANNs

JOSHUA PETER EBENEZER – 15EC10023

DIGITAL SIGNAL PROCESSING PROJECT

ABSTRACT

This report presents a scheme to differentiate between the ECG (Electrocardiographs) signals of patients having healthy hearts and patients showing symptoms of paroxysmal atrial fibrillation (PAF). Different features are extracted and principal component analysis is used to select the best 20 features. Artificial neural networks are trained with different hidden layers and compared to select the optimum number of layers. Different layers gave different results, with the best result being 100% sensitivity and 83.3% specificity on test data with randomly initialized weights and 15 hidden neurons in the hidden layer.

INTRODUCTION

The task is to differentiate between ECG (Electrocardiographs) signals of patients having healthy hearts and patients showing symptoms of paroxysmal atrial fibrillation (PAF). Atrial fibrillation occurs when the atria, or upper chambers of the heart, lose their normal rhythm and beat chaotically. As the population ages, the prevalence is expected to rise; currently approximately 6% of the US population over the age of 65 are diagnosed with this arrhythmia. The development of accurate predictors of the acute onset of PAF is clinically important because of the increasing possibility of electrically stabilizing and preventing the onset of atrial arrhythmias with different atrial pacing techniques.

The report is organized as follows. A brief description of the database is provided, and a few characteristics studied. Feature extraction is described in the next section. Finally, results from ANN training are presented. Appendices at the end include brief descriptions of PCA, ANNs, Cross-entropy loss, and backpropagation.

DATABASE

The database of two-channel ECG recordings was created for use in the Computers in Cardiology Challenge 2001, and is available in PhysioNet. The learning set that was used by the author consists of 50 records. Each record has two signals, both taken of the same patient at the same time with two different electrodes. Each signal is of 5 minutes' duration and sampled at 128 Hz. In training, 32 records were randomly sampled for training, and 8 for validation. 10 were used for testing.

CHARACTERISTICS OF THE DATA

The signals were first normalized with respect to total energy (so that all signals have unit total energy). A normal ECG signal, titled 'n01cm.mat' in the database, is shown in Fig. 1 (taken from training

set), before normalization. An ECG of a patient with PAF, titled ‘p14cm.mat’ in the database, is shown in Fig. 2 before normalization.

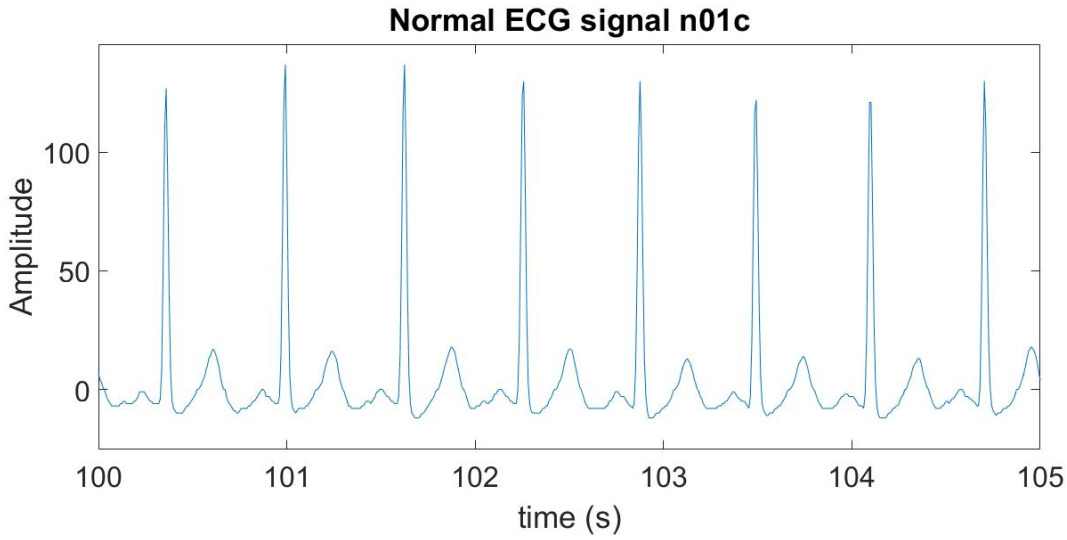

Fig. 1 *Normal ECG signal (time domain)*

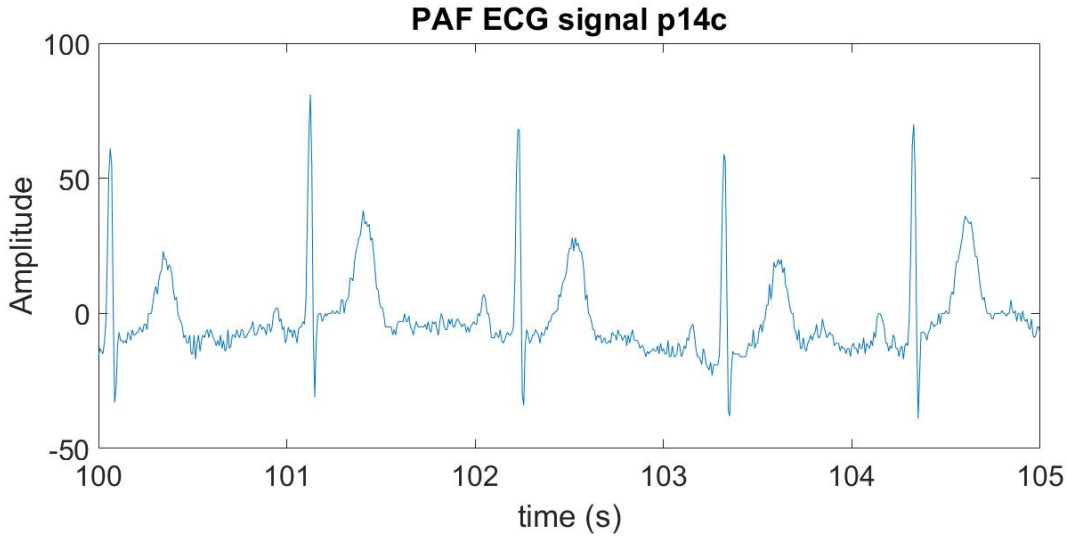

Fig. 2 *ECG signal of patient with PAF (p14c)*

The FFTs of a few normal signals (after energy normalization) are shown in Fig. 3. FFTs of a few abnormal signals (also after energy normalization) are shown in Fig. 4. The spread of the data might be misleading. Abnormal signals actually have a larger peak at 0 Hz than normal signals, hence the spectra near 0 Hz looks smaller in amplitude in comparison to that of normal signals, but it is actually greater. Hence, the summed energy of the lowpass portion of abnormal signals is greater than that of normal signals. This is because the fluctuations in PAF cause a larger, non-zero DC average. Also, while normal signals do not have an envelope, most PAF signals have a very low frequency slowly varying envelope.

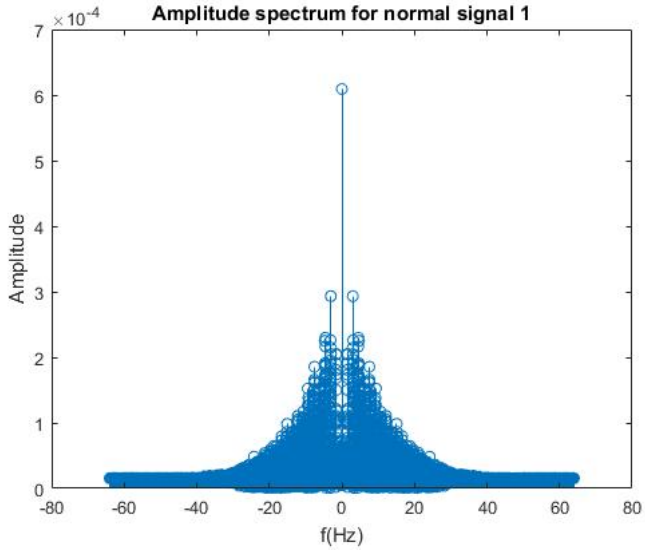

Fig. 3 *Spectrum of normal ECG signal*

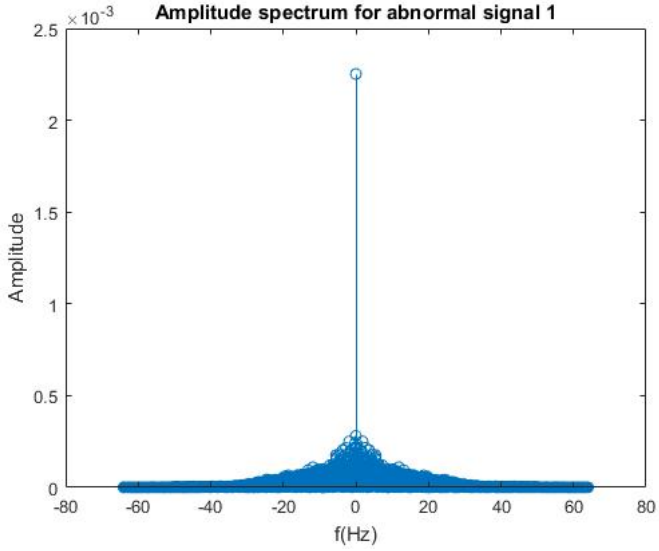

Fig. 4 *Spectrum of abnormal ECG signal*

FEATURE EXTRACTION

The features extracted for this study are shown in Table 1. There are 50 records and hence 50 rows for each feature set. There are 2 channels per record, and hence the additional dimension.

Name of feature	**Size of feature matrix for entire dataset**
Filtered Energy	50 x 2 x 1
Autocorrelation	50 x 2 x 1
Mean	50 x 2 x 1
Standard deviation	50 x 2 x 1

Range	50 x 2 x1
Detailed coefficients of 4th level DWT	50 x 2 x 2047
Skewness	50 x 2 x 1
Kurtosis	50 x 2 x 1

Table 1 *Feature description and dimensionality*

The features extracted for the two channels were concatenated, and all features were flattened and joined to form a 50 x 4824 feature matrix. Principal component analysis was performed on only the training matrix to extract 20 maximum-variance, orthogonal components, and the corresponding features were also extracted from the testing set feature matrix (which is 10 x 4824).

ARTIFICIAL NEURAL NETWORK TRAINING

Since the number of hidden nodes in the Neural network is a heuristic parameter, networks with sizes 12, 15, 20 were trained and tested. The results are shown in the following confusion matrices and ROC curves.

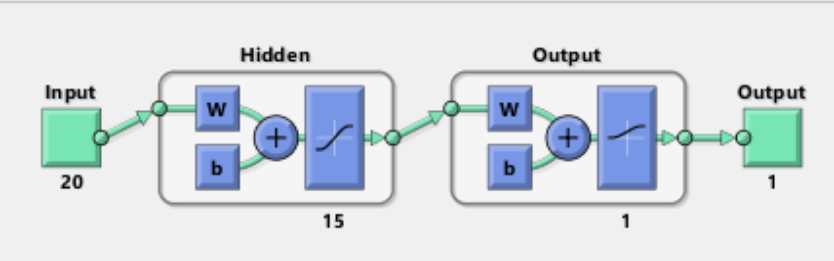

Fig. 5 *View of net architecture in MATLAB®*

Cross entropy is used as the loss function for all nets. The weights are randomly initialized, so each time the net is trained the results are different. Scaled conjugate gradient backpropagation was used. The net was trained for 15 epochs. In summary, the results are as follows:

Number of hidden nodes	Sensitivity	Specificity
12	75%	100%
15	100%	83.3%
20	100%	83.3%

Table 2 *Sensitivity and specificity for various networks*

Fig. 6 *Confusion matrix for 12 hidden nodes*

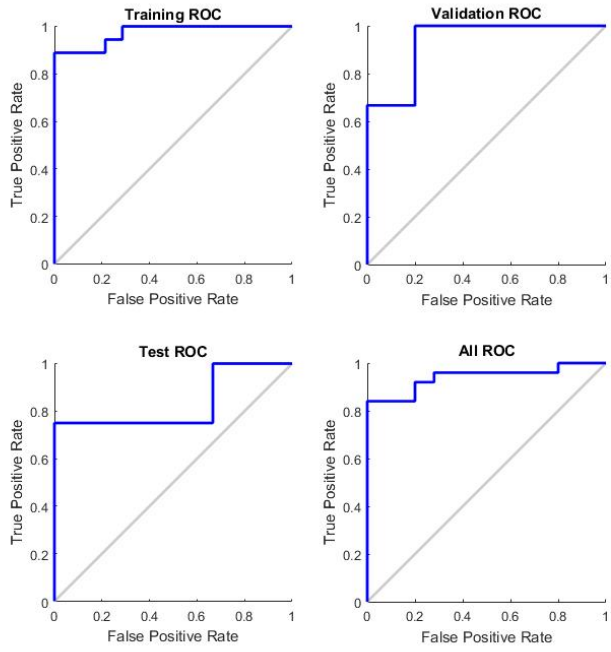

Fig. 7 *ROCs for 12 hidden nodes*

Fig. 8 *Confusion matrices for 15 hidden nodes*

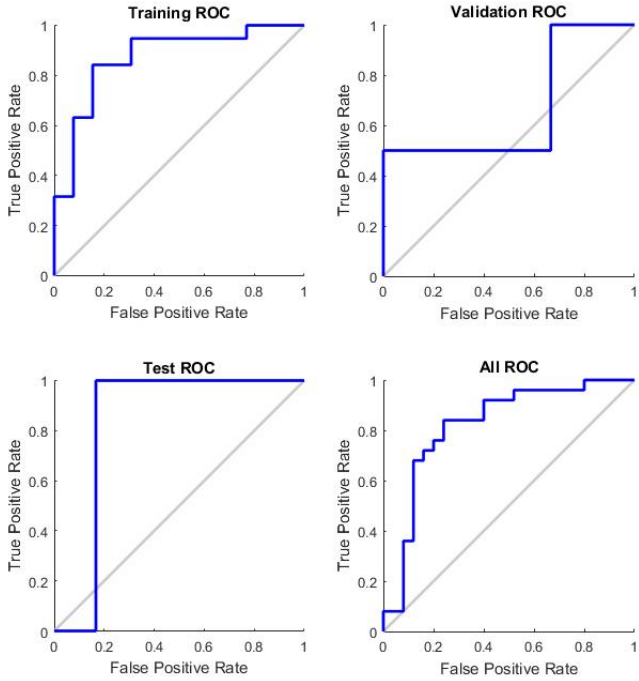

Fig. 9 *ROC for 15 hidden nodes*

Fig. 10 *Confusion matrices for 20 hidden nodes*

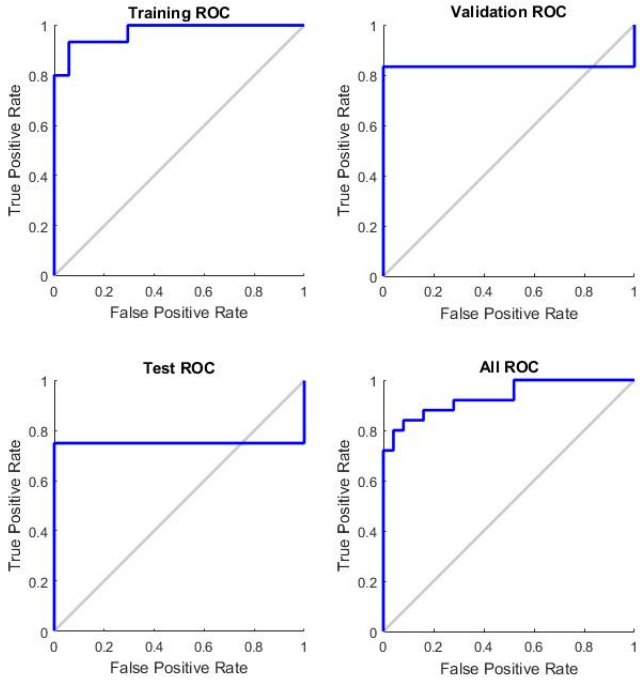

Fig. 11 *ROCs for 20 hidden nodes*

CONCLUSION

The report shows that the artificial neural network was able to use the selected features to successfully classify ECG signals. 15 hidden neurons gave the best result, for a training-test division of 40-10. A larger dataset will probably give better results for the ANN.

APPENDICES

PRINCIPAL COMPONENT ANALYSIS

PCA transforms data to another coordinate system such that the projection of the data on the first component has the greatest variance, the projection on second component has the second greatest variance, and so on. Intuitively, an ellipsoid is being fit to the data. Smaller axes of the ellipsoid have lesser variance, and hence discarding them will not cause us to lose much information.

For the first component, the expression to be maximized is the variance. It can be proved that this comes down to finding the eigenvector of the matrix $X^T X$ corresponding to the largest eigenvalue, where X is the feature matrix. For further components, the projection of the feature matrix along the first component is subtracted from the feature matrix. This creates a set that is orthogonal to the first component. From this matrix, the maximum eigenvalue is again found, and the process continues until the number of reduced features required is found.

ARTIFICIAL NEURAL NETWORK

ANNs consist of an input layer that has number of nodes equal to the number of features, a hidden layer that has a variable number of nodes, and an output layer with nodes equal to the number of classes. Each neuron (node) in the network takes input from all other neurons in the layer preceding it, and computes a linear function of the inputs to generate an output, which is then thresholded or transformed by a non-linear operation (softmax or ReLU). This output indicates how strongly the neuron is firing. The output is fed as the input to all neurons in the next layer. The following figure shows a typical neuron.

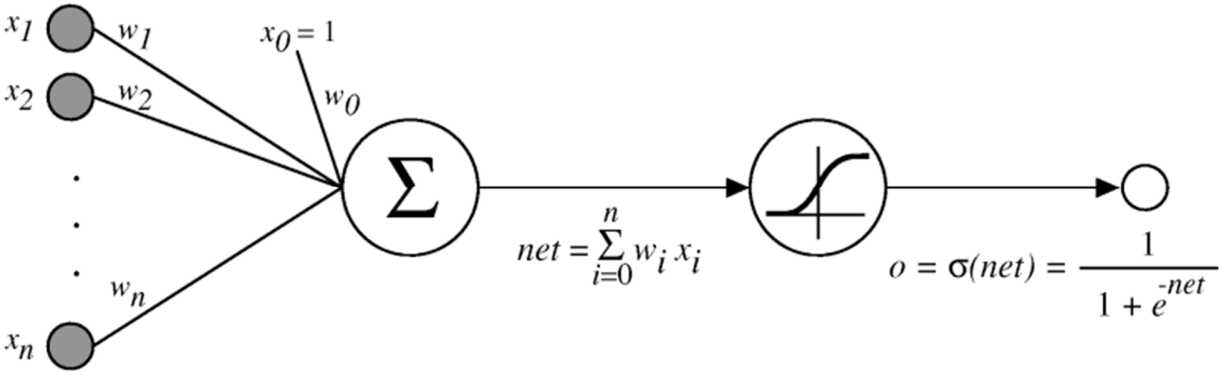

Fig. A1 *Neuron activation and firing*

The activation function at the final layer is a softmax prediction function. The final output is compared with the ground truth quantitatively using a loss function (cross-entropy in this case). The error is fed backward through the network to update the weights so that the prediction better matches the ground truth.

Cross entropy loss for a two-class classification problem is defined as:

$$-(ylog(p) + (1-y)log(1-p))$$

where y is the ground truth and p is the probability that the input belongs to that class from the network softmax function.

Backpropagation is a method to update the weights at each layer using the loss gradients. The objective is to find the weights for which the loss function is at an optimal minimum. At the output layer, the gradient of loss is calculated with respect to the output. Using the chain rule, the derivative of the loss with respect to the weights of the final layer can be written as the derivative of loss with respect to the output, multiplied by the derivative of the output with respect to the weights. This derivative is then fed backward to the previous layer as shown in Fig. A2.

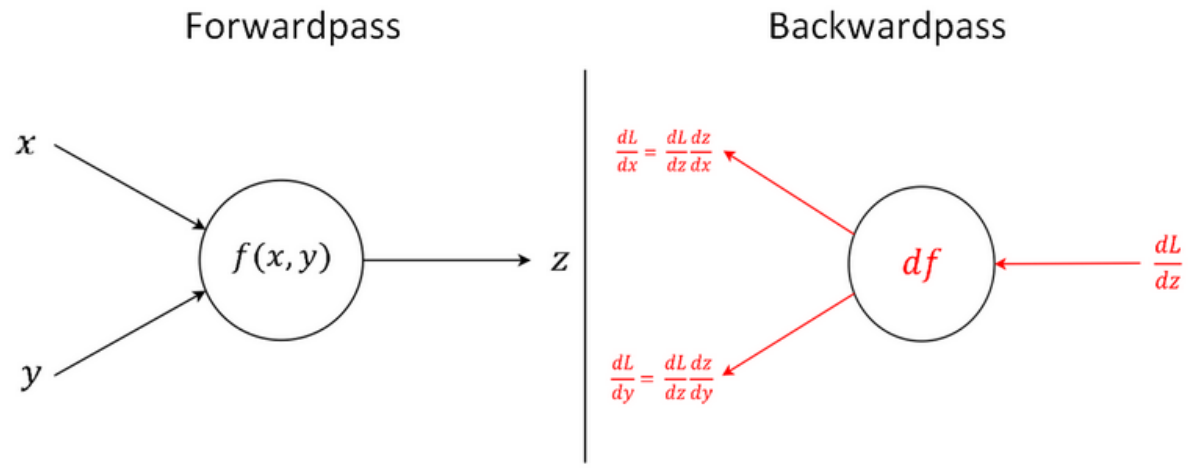

Fig. A2 *Backpropagation*

Each weight in the layer is iteratively updated as

$$w^{t+1} = w^t - \alpha \frac{\partial E}{\partial w^t}$$

where E is the loss function, and α is a heuristic parameter called the learning rate.

MATLAB® IMPLEMENTATION

The following code was used to extract the features from the dataset, apply PCA on the training set features, and extract the corresponding features from the testing set. The MATLAB® neural network toolbox was used to train the neural network and visualize the outputs.

```
clear;
clc;
close all;

fnames=dir('train/*.mat');
normal = zeros(50,2,38400);
arrhy = zeros(50,2,38400);

test_normal = zeros(5,2,38400);
test_arrhy = zeros(5,2,38400);

Fs = 128;
% number of sampling instances
N = 38400;
% frequency base
f = (-N/2:N/2-1)*Fs/N;
% time base
t = 0:1/Fs:5*60-1/Fs;

data_size = length(fnames); % no of samples (+ve and -ve)
val_index = 1:5;    % 5 fold cross validation
data_indices = 1:data_size;    % indices of names of files in database
normal_indices = 1:data_size/2;
abnormal_indices = data_size/2+1:data_size;

%% Butterworth lowpass filter
n = 4;
Wn =12/(Fs/2);
% Zero-Pole-Gain design
[z,p,k] = butter(n,Wn,'low');
sos = zp2sos(z,p,k);

precision = zeros(1,5);
recall = zeros(1,5);

for val_index = 1    % cross validation folds
    test_indices_normal = (val_index-1)*5+1:val_index*5;        % test
indices
    test_indices_abnormal = data_size/2 + (val_index-1)*5+1:data_size/2 +
val_index*5;
    ntest_indices_bool = ismember(normal_indices,test_indices_normal);
    antest_indices_bool = ismember(abnormal_indices,test_indices_abnormal);
    train_indices_normal = normal_indices(~ntest_indices_bool);    % train
indices
    train_indices_abnormal = abnormal_indices(~antest_indices_bool);

    index = 0;
    target = ones(1,50);
    target(1:25)=0;
    norm_energy =zeros(data_size/2-5,2,1);
    abnorm_energy = zeros(data_size/2-5,2,1);

    norm_autocorr = zeros(data_size/2-5,2,1);
    abnorm_autocorr = zeros(data_size/2-5,2,1);
```

```
    norm_mean =zeros(data_size/2-5,2,1);
    abnorm_mean = zeros(data_size/2-5,2,1);

    norm_range =zeros(data_size/2-5,2,1);
    abnorm_range = zeros(data_size/2-5,2,1);

    norm_std =zeros(data_size/2-5,2,1);
    abnorm_std = zeros(data_size/2-5,2,1);

    norm_dwt4 =zeros(data_size/2-5,2,2407);
    abnorm_dwt4 = zeros(data_size/2-5,2,2407);

    %% Training of normal instances
    for ntrain_index = train_indices_normal
        % Read the data and normalize it
        index = index+1;
        file=fullfile('train',fnames(ntrain_index).name);
        s = load(file);
        normal(index,:,:) = L2normalize(s);

        % Find the energy of the filtered signals
        % Find auto-correlations of original signals
        % Define the product(s) as features for each signal

[norm_energy(index,:),norm_autocorr(index,:),norm_mean(index,:),norm_std(inde
x,:),norm_range(index,:),norm_dwt4(index,:,:)] =
feature_extract(normal(index,:,:),sos);
    end
    index = 0;
    norm_features =
cat(1,norm_energy(:,:)',norm_autocorr(:,:)',norm_mean(:,:)',norm_std(:,:)',no
rm_range(:,:)',norm_dwt4(:,:)');
    %% Training of abnormal samples
    for antrain_index = train_indices_abnormal
        index = index+1;
        % Read the data and normalize it
        file=fullfile('train',fnames(antrain_index).name);
        s = load(file);
        arrhy(index,:,:) = L2normalize(s);

        % Find the energy of the filtered signals
        % Find auto-correlations of original signals

[abnorm_energy(index,:),abnorm_autocorr(index,:),abnorm_mean(index,:),abnorm_
std(index,:),abnorm_range(index,:),abnorm_dwt4(index,:,:)] =
feature_extract(arrhy(index,:,:),sos);
    end
    abnorm_features =
cat(1,abnorm_energy(:,:)',abnorm_autocorr(:,:)',abnorm_mean(:,:)',abnorm_std(
:,:)',abnorm_range(:,:)',abnorm_dwt4(:,:)');

    %% ------------------------------- Validation ------------------------
- %%
    tp = 0;
```

```
    tn = 0;
    fp=0;
    fn=0;

    nindex = 0;     % index for normal test cases

    ntest_energy_val = zeros(5,2);
    ntest_auto_corr = zeros(5,2);
    ntest_mean = zeros(5,2);
    ntest_range = zeros(5,2);
    ntest_std = zeros(5,2);
    ntest_dwt4 = zeros(5,2,2407);

    atest_energy_val = zeros(5,2);
    atest_auto_corr = zeros(5,2);
    atest_mean = zeros(5,2);
    atest_range = zeros(5,2);
    atest_std = zeros(5,2);
    atest_dwt4 = zeros(5,2,2407);

    %% Testing for normal test cases
    for ntest_index=test_indices_normal
        file=fullfile('train',fnames(ntest_index).name);
        s = load(file);

        nindex = nindex+1;

        test_normal(nindex,:,:) = L2normalize(s);
        % Find energy of the signal and normalize

[ntest_energy_val(nindex,:),ntest_auto_corr(nindex,:),ntest_mean(nindex,:),nt
est_std(nindex,:),ntest_range(nindex,:),ntest_dwt4(nindex,:,:)]=feature_extra
ct(test_normal(nindex,:,:),sos);

    end

    ntest_features =
cat(1,ntest_energy_val(:,:)',ntest_auto_corr(:,:)',ntest_mean(:,:)',ntest_std
(:,:)',ntest_range(:,:)',ntest_dwt4(:,:)');

    aindex = 0;     % index for abnormal test cases

    %% Testing for abnormal test cases
    for atest_index = test_indices_abnormal

        file=fullfile('train',fnames(atest_index).name);
        s = load(file);

        aindex = aindex+1;

        test_arrhy(aindex,:,:)=L2normalize(s);


[atest_energy_val(aindex,:),atest_auto_corr(aindex,:),atest_mean(aindex,:),at
```

```
est_std(aindex,:),atest_range(aindex,:),atest_dwt4(aindex,:,:)]=feature_extra
ct(test_arrhy(aindex,:,:),sos);

    end

    atest_features =
cat(1,atest_energy_val(:,:)',atest_auto_corr(:,:)',atest_mean(:,:)',atest_std
(:,:)',atest_range(:,:)',atest_dwt4(:,:)');

    features =
cat(2,norm_features,ntest_features,abnorm_features,atest_features);
    train_features = cat(2,norm_features,abnorm_features);

    numberOfDimensions = 20;
    [coeff,score] = pca(train_features');
    reducedDimension = coeff(:,1:numberOfDimensions);
    reducedFeatures = (features' * reducedDimension)';
end



function [normal] = L2normalize(s)
normal=s.val;
% Find energy of the signal and normalize
energy_val = sum(normal(1,:).^2);
energy2 = sum(normal(2,:).^2);
normal(1,:)=normal(1,:)/sqrt(energy_val);
normal(2,:)=normal(2,:)/sqrt(energy2);
end


function [energy,auto_corr,mean_val,std_dev,range_val,dwt4] =
feature_extract(signal,sos)

s1=signal(1,1,:);
s2=signal(1,2,:);

% filter the signals
y1 = sosfilt(sos,s1);
y2 = sosfilt(sos,s2);
energy(1) = sum(y1.^2);
energy(2) = sum(y2.^2);

% Auto-correlation of original signals
% 128 samples corresponds to 10s
auto_corr(1) = sum(s1(1,1,1:end-128).*s1(1,1,129:end));
auto_corr(2) = sum(s2(1,1,1:end-128).*s2(1,1,129:end));

mean_val(1) = mean(s1);
mean_val(2) = mean(s2);

std_dev(1) = std(s1);
std_dev(2) = std(s2);
```

```
range_val(1) = range(s1);
range_val(2) = range(s2);

[C,~] = wavedec(s1(:),4,'db4');
dwt4(1,:)=C(2407:4813);

[C,~] = wavedec(s2(:),4,'db4');
dwt4(2,:)=C(2407:4813);


%energy_val = energy(1)*abs(auto_corr(1));
%auto_corr = energy(2)*abs(auto_corr(2));

end
```