MATLAB CODE:

%% Load Files%%

load('ECGData.mat’)

percent\_train = 70;

[trainData,testData,trainLabels,testLabels] = ...

    helperRandomSplit(percent\_train,ECGData);

helperPlotRandomRecords(ECGData,14)

N = size(ECGData.Data,2);

sn = waveletScattering('SignalLength',N, 'InvarianceScale',150,'SamplingFrequency',128);

[fb,f,filterparams] = filterbank(sn);

subplot(211)

plot(f,fb{2}.psift)

xlim([0 128])

grid on

title('1st Filter Bank Wavelet Filters');

subplot(212)

plot(f,fb{3}.psift)

xlim([0 128])

grid on

title('2nd Filter Bank Wavelet Filters');

xlabel('Hz');

figure;

phi = ifftshift(ifft(fb{1}.phift));

psiL1 = ifftshift(ifft(fb{2}.psift(:,end)));

t = (-2^15:2^15-1).\*1/128;

scalplt = plot(t,phi);

hold on

grid on

ylim([-1.5e-4 1.6e-4]);

plot([-75 -75],[-1.5e-4 1.6e-4],'k--');

plot([75 75],[-1.5e-4 1.6e-4],'k--');

xlabel('Seconds'); ylabel('Amplitude');

wavplt = plot(t,[real(psiL1) imag(psiL1)]);

legend([scalplt wavplt(1) wavplt(2)],{'Scaling Function','Wavelet-Real Part','Wavelet-Imaginary Part'});

title({'Scaling Function';'Coarsest-Scale Wavelet First Filter Bank'})

hold off

scat\_features\_train = featureMatrix(sn,trainData');

Nwin = size(scat\_features\_train,2);

scat\_features\_train = permute(scat\_features\_train,[2 3 1]);

scat\_features\_train = reshape(scat\_features\_train,...

    size(scat\_features\_train,1)\*size(scat\_features\_train,2),[]);

scat\_features\_test = featureMatrix(sn,testData');

scat\_features\_test = permute(scat\_features\_test,[2 3 1]);

scat\_features\_test = reshape(scat\_features\_test,...

    size(scat\_features\_test,1)\*size(scat\_features\_test,2),[]);

[sequence\_labels\_train,sequence\_labels\_test] = createSequenceLabels(Nwin,trainLabels,testLabels);

scat\_features = [scat\_features\_train; scat\_features\_test];

allLabels\_scat = [sequence\_labels\_train; sequence\_labels\_test];

rng(1);

template = templateSVM(...

    'KernelFunction', 'polynomial', ...

    'PolynomialOrder', 2, ...

    'KernelScale', 'auto', ...

    'BoxConstraint', 1, ...

    'Standardize', true);

classificationSVM = fitcecoc(...

    scat\_features, ...

    allLabels\_scat, ...

    'Learners', template, ...

    'Coding', 'onevsone', ...

    'ClassNames', {'ARR';'CHF';'NSR'});

kfoldmodel = crossval(classificationSVM, 'KFold', 5);

%%

% Compute the loss and the confusion matrix. Display the accuracy.

predLabels = kfoldPredict(kfoldmodel);

loss = kfoldLoss(kfoldmodel)\*100;

confmatCV = confusionmat(allLabels\_scat,predLabels)

fprintf('Accuracy is %2.2f percent.\n',100-loss);

classes = categorical({'ARR','CHF','NSR'});

[ClassVotes,ClassCounts] = helperMajorityVote(predLabels,[trainLabels; testLabels],classes);

CVaccuracy = sum(eq(ClassVotes,categorical([trainLabels; testLabels])))/162\*100;

fprintf('True cross-validation accuracy is %2.2f percent.\n',CVaccuracy);

MVconfmatCV = confusionmat(categorical([trainLabels; testLabels]),ClassVotes);

MVconfmatCV

model = fitcecoc(...

     scat\_features\_train, ...

     sequence\_labels\_train, ...

     'Learners', template, ...

     'Coding', 'onevsone', ...

     'ClassNames', {'ARR','CHF','NSR'});

predLabels = predict(model,scat\_features\_test);

[TestVotes,TestCounts] = helperMajorityVote(predLabels,testLabels,classes);

testaccuracy = sum(eq(TestVotes,categorical(testLabels)))/numel(testLabels)\*100;

fprintf('The test accuracy is %2.2f percent. \n',testaccuracy);

confusionchart(categorical(testLabels),TestVotes)

%% Supporting Functions

% \*helperPlotRandomRecords\* Plots four ECG signals randomly chosen from

% |ECGData|.

function helperPlotRandomRecords(ECGData,randomSeed)

% This function is only intended to support the XpwWaveletMLExample. It may

% change or be removed in a future release.

if nargin==2

    rng(randomSeed)

end

M = size(ECGData.Data,1);

idxsel = randperm(M,4);

for numplot = 1:4

    subplot(2,2,numplot)

    plot(ECGData.Data(idxsel(numplot),1:3000))

    ylabel('Volts')

    if numplot > 2

        xlabel('Samples')

    end

    title(ECGData.Labels{idxsel(numplot)})

end

end

%%

% \*helperMajorityVote\* Finds the mode in the predicted class labels for

% each set of scattering time windows. The function returns both the class

% label modes and the number of class predictions for each set of

% scattering time windows. If there is no unique mode, |helperMajorityVote|

% returns a class label of "error" to indicate that set of scattering

% windows is a classification error.

function [ClassVotes,ClassCounts] = helperMajorityVote(predLabels,origLabels,classes)

% This function is in support of ECGWaveletTimeScatteringExample. It may

% change or be removed in a future release.

% Make categorical arrays if the labels are not already categorical

predLabels = categorical(predLabels);

origLabels = categorical(origLabels);

% Expects both predLabels and origLabels to be categorical vectors

Npred = numel(predLabels);

Norig = numel(origLabels);

Nwin = Npred/Norig;

predLabels = reshape(predLabels,Nwin,Norig);

ClassCounts = countcats(predLabels);

[mxcount,idx] = max(ClassCounts);

ClassVotes = classes(idx);

% Check for any ties in the maximum values and ensure they are marked as

% error if the mode occurs more than once

modecnt = modecount(ClassCounts,mxcount);

ClassVotes(modecnt>1) = categorical({'error'});

ClassVotes = ClassVotes(:);

%-------------------------------------------------------------------------

function modecnt = modecount(ClassCounts,mxcount)

modecnt = Inf(size(ClassCounts,2),1);

for nc = 1:size(ClassCounts,2)

    modecnt(nc) = histc(ClassCounts(:,nc),mxcount(nc));

end

end

% EOF

end