Eigenfaces and Music Identification:
SVD Analysis of Extended Yale Faces B Database and Music Samples

Franklin Williams
University of Washington Applied Mathematics
https://github.com/streater512/AMATH-582
March 7, 2020

Abstract
Facial perception is a key skill learned by humans at an early age and is a key area of research within the field of neuroscience. Facial perception is a paramount social skill that allows humans to convey and process information. In this report, a singular value decomposition (SVD) analysis of images containing faces is conducted to identify key features used to detect faces. Not only does SVD analysis give insights into images, but audio signals as well. While birds and other animals can create songs for mating, identifying music and various music genres is uniquely human. The human brain can quickly identify musical features such as pitch, rhythm, and tonality. Using inputs from SVD analysis of song samples, we train various machine learning (ML) classifiers to identify musical artists and genres in three different test cases.

Section I: Introduction

This report begins with an introduction to the practicality of SVD, ML classifiers, as well as an overview of data collected and experiments run. In Section II, a theoretical background of the techniques and models used will be discussed. Section III covers the algorithm implementation and development in MATLAB. Section IV provides the computational results of our tests. To conclude, Section V summarizes the results and ends with final remarks.

Artists have long understood the importance of abstract representation of objects through edges detection of key features. Figure 1 shows an artist's representation of Seattle. Key features of the city, such as the Space Needle and Mount Rainer, are instantly identified by the human brain despite the lack of details or color, due to edge detection. From a mathematical standpoint, wavelets represent an ideal way to represent multi-scale information due to their ability to detect edges within images and key features of audio data (Kutz). Two-dimensional and one-dimensional wavelet decomposition are conducted on facial image data and audio samples, respectively, to visualize the model's edge detection and train ML classifiers.

Figure (1): Artist representation of Seattle skyline
Credit: https://www.pinterest.com/pin/345651340128026752/

Section II: Theoretical Background

One of the most important attributes of a wavelet is its ability to represent a given signal. A wavelet constructs a signal using a single function $\psi \in L^2$, written $\psi_{m,n}$ that represents binary dilations by 2^m and translations $n2^{-m}$ so that

$$\psi_{m,n} = 2^{-m}\psi\left(2^m\left(x - \frac{n}{2^m}\right)\right) = 2^{\frac{m}{2}}\psi(2^m x - n) \tag{1}$$

where *m* and *n* are integers. Additionally, orthogonality of the functions $\psi_{m,n}$ must exist. Orthogonality is defined as

$$(\psi_{m,n}, \psi_{k,l}) = \int_{-\infty}^{\infty} \psi_{m,n}(x)\psi_{k,l}(x)dx = \delta_{m,k}\delta_{n,l} \tag{2}$$

where δ_{ij} is the Dirac delta defined by

$$\delta_{ij} = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

where *i*, *j*, are integers.

Wavelet decomposition is conducted using the Discrete Wavelet Transform (DWT). The DWT is defined by

$$\begin{aligned} \mathcal{W}_\psi(f)(m,n) = (f, \psi_{m,n}) &= \int_{-\infty}^{\infty} f(t)\overline{\psi}_{m,n}(t)dt \\ &= a_0^{-m/2} \int_{-\infty}^{\infty} f(t)\overline{\psi}(a_0^{-\frac{m}{2}} t - n\, b_{0)} dt. \end{aligned} \tag{3}$$

Unlike the Fourier Transform which sacrifices information across the time domain, the DWT gives multi-resolution insight into a signal across time and frequency domains. The DWT of a signal is calculated by passing it through a series of low pass and high pass filters. Low pass filters provide a smoother form of the signal and capture longer frequencies. On the other hand, high pass filters ignore long frequencies and give a sharper form of the signal by capturing high frequencies. If $\psi_{m,n}$ are complete, then the signal can be expanded in the wavelet basis:

$$f(t) = \sum_{m,n=-\infty}^{\infty} (f, \psi_{m,n})\psi_{m,n}(t). \tag{4}$$

The naïve Bayes classifier is based on Bayes' theorem:

$$\mathbb{P}(C_k|\mathbf{X}) = \frac{\mathbb{P}(C_k)\mathbb{P}(\mathbf{X}|C_k)}{\mathbb{P}(\mathbf{X})} = \text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}} \tag{5}$$

Assuming that all **X** are mutually independent, conditional on the category C_k, the conditional distribution over the class variable C is:

$$\mathbb{P}(C_k|x_{\mathbf{1}}, \dots, x_n) = \frac{1}{Z}\mathbb{P}(C_k)\prod_{I=1}^{n}\mathbb{P}(x_{\boldsymbol{i}}|C_k) \tag{6}$$

Where Z is some constant. The classifier adds a decision rule on top of the independent feature model:

$$\hat{y} = \underset{k\in\{1,\dots,K\}}{\operatorname{argmax}}\, \mathbb{P}(C_k)\prod_{I=1}^{n}\mathbb{P}(x_{\boldsymbol{i}}|C_k) \tag{7}$$

The motivation behind a support-vector machine (SVM) is to find the best divisor between sets of data. The SVM constructs a hyperplane or set of hyperplanes in a high-dimension space that can be used for classification. To compute the SVM classifier we utilize quadratic programming to minimize the expression:

$$[\frac{1}{n}\sum_{i=1}^{n} \max\left(0, 1 - y_i(w \cdot x_i - b)\right)] + \lambda \|w\|^2 \tag{8}$$

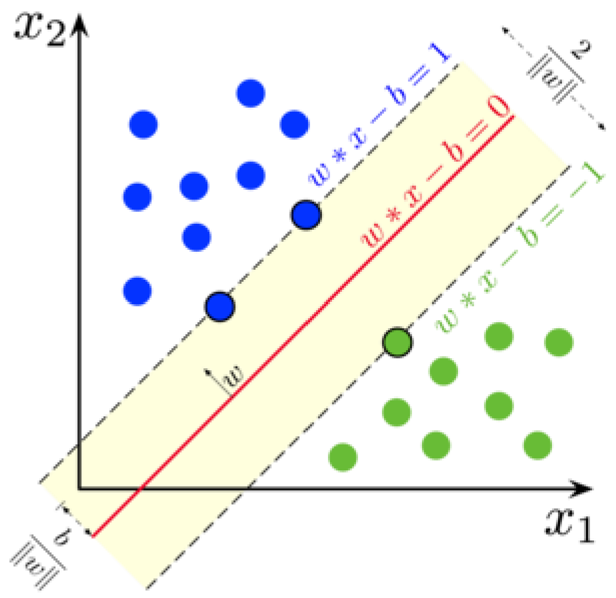


Figure (2): Visual representation of SVM
Credit: https://en.wikipedia.org/wiki/Support-vector_machine

Classification trees are a form of decision trees where the target variable can take a discrete set of values. The classification tree beings at a root node and divides the input data based upon criteria from the data's features and classifies a portion of the input data. Each successive branch further divides the data based on new criteria from the features. This process continues until all the data are classified.

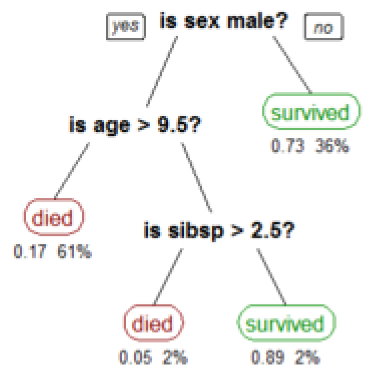


Figure (3): Visualization of Classification Tree from "Titanic" dataset
Credit: https://en.wikipedia.org/wiki/Decision_tree_learning

Section III: Algorithm Implementation and Development

The facial images used in the analysis come from the Extended Yale Faces B Database and included Cropped and Uncropped faces. The Cropped faces data contain 64 photos of 38 participants. Each photo is cropped to contain only the subject's face. Light was cast at different angles to cast shadows on various regions of the participants face. The Uncropped faces data contain 11 photos of 15 participants. In the photo samples, participants are making various facial expressions (sad, wink, smile, etc.).

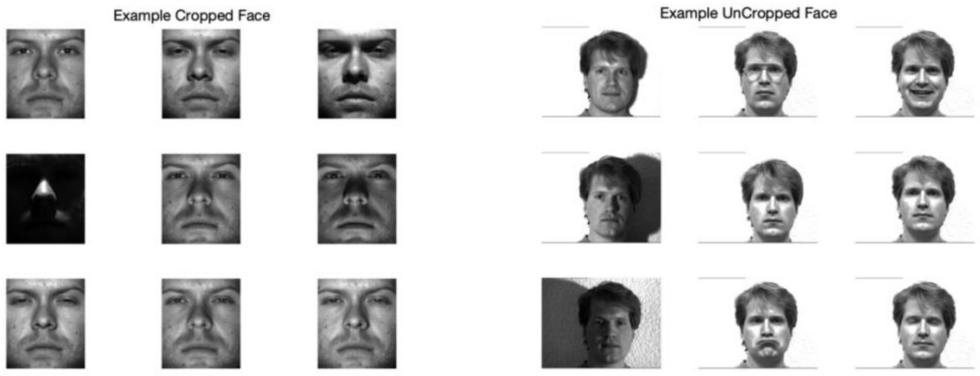


Figure (4) Yale Face Example
Credit: Extended Yale Faces B Database

The two data sets were then processed using wavelet decomposition using various wavelet formations (Haar, Coiflets 1, Symlets 2, Fejé-Krovkin 4, Discrete Meyer). For each of these wavelets, we compare the Horizontal, Vertical and Edge components which are detected by the wavelets. Following, we continue our SVD analysis with the Haar wavelet, for simplicity. Each image is decomposed using the Haar wavelet. The edges detected are then compiled into a matrix with each column representing the edges found from the face. SVD is then performed on the edge matrix and the features are visualized. The results are discussed in the following section.

Three tests were conducted on sampled audio data. In each test, 100 5-second audio clips from various artists were sampled. The samples were then passed through a 1-dimensional DWT, using the Haar Wavelet. The DWT data was then processed using SVD analysis of the time-frequency data, which was used to three ML classifiers: Naive Bayes, SVM, and a classification tree.

In Test 1, we attempt to identify three bands from three genres, New Wave, Rock, and Classical. Test 1 data include samples from the live concert/movie *Talking Heads – Stop Making Sense*, *Widespread Panic – Live 2016-09-15 Tuscaloosa Amphitheater*, and *Mozart – Masters of Classical Music Vol 1*.

In Test 2 we attempt to increase the difficulty for the algorithm to classify artists by choosing music within the same genre. The Grateful Dead pioneered the psychedelic rock scene within the United States throughout the 1960s through the 1990s. To pay tribute to their influence, bands across the nation perform Grateful Dead renditions, "covers", with their own musical styles intermixed. In Test 2 we train the classifiers using samples from *Grateful Dead – Live Cornell*

5/8/77, *Dead & Company 10-29-15 Times Union Center*, and *Joe Russo's Almost Dead 2013-12-27 Live at the Capitol Theatre.*

Finally, in Test 3 we represent various genres with a subset of artists from each genre. Samples from *Greensky Bluegrass – Live 2006-03-17, Billy Strings –Live 2020-01-18*, and *Del McCoury Band 2005-08-31* represented the Bluegrass genre. To represent Rock, the same samples from Widespread Panic, Grateful Dead and Joe Russo's Almost Dead were used. Lastly, a collection of various Hip Hop artists was sampled from *The Greatest 90s Dance Hits*, which include artist such as A Tribe Called Quest and Coolio.

Section IV: Computational Results

Wavelet Representation of Yale Faces:

In the below figures we show the horizontal (far left column), vertical (left center), detected edges (right center), and ideal face (far right) for five wavelets. From top to bottom, the wavelets used in edge detection are Haar, Coiflets 1, Symlets 2, Fejé-Krovkin 4, Discrete Meyer.

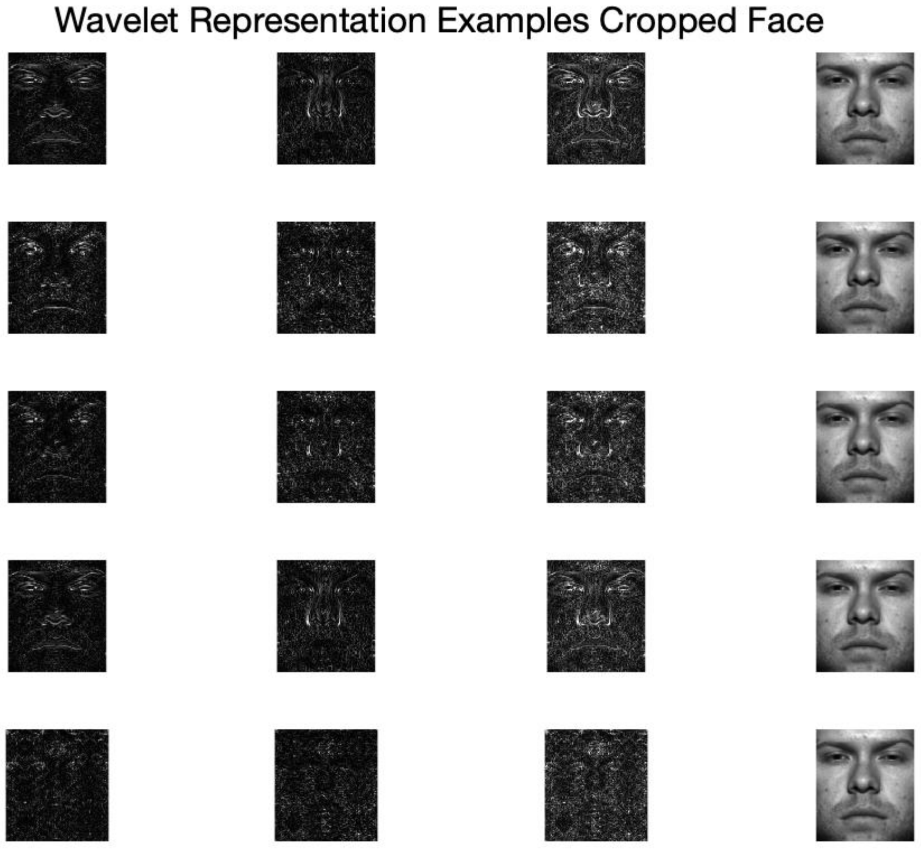


Figure (5) Edge Detection and Ideal face by Wavelet

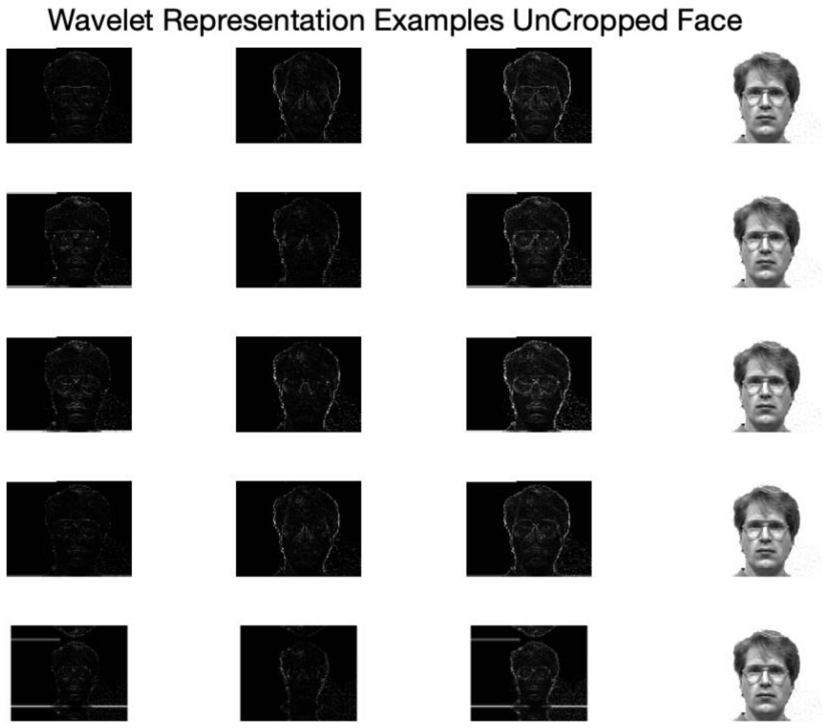


Figure (6) Edge Detection of Yale Faces by Wavelet

In each case the edges detected from the Cropped Face data set give a much more detailed representation of the participants face. Additionally, although it is one of the simpler wavelets, the Haar wavelet appears to do nice job of clearly detecting key facial features like the eyes, nose, and mouth. For the Uncropped sample, the wavelets clearly outline the subject's head and some of the major features. However, detail is missing within the face.

Principal Components of Yale Faces:

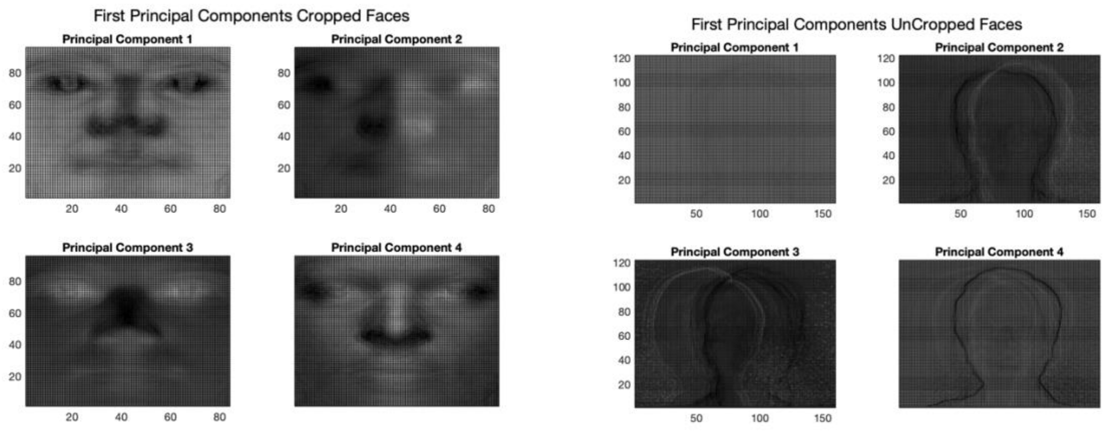


Figure (7) Principal Components of Yale Faces Haar Wavelet

Above we see the first four principal components for both the Cropped and Uncropped data sets. Within the Cropped samples, see strong signatures of eyes, a nose, and the faint outlines of a mouth. We can think of these strong signatures as the “average” of each of the features from the different participants. On the other hand, there appears to be a blank first principal component from the Uncropped data. Upon further inspection this makes sense, as each of the participants were standing in front of a blank background that was common in all photos. The SVD analysis correctly separated the background from foreground from each of the photos. Additionally, we can see a darker outline that appears to be in the “average” shape of the participants heads in the following three principal components. However, no facial features are detected within the first four components. This is unsurprising, as each photo was of a different expression, thus making the details of the face more complex than the Cropped data set.

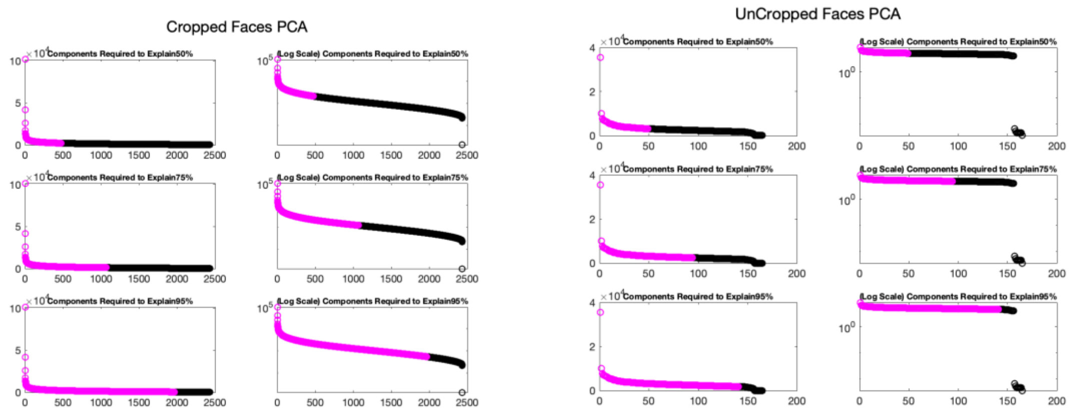


Figure (8) PCA Analysis

Above we plot the singular values from the SVD for the Cropped and Uncropped faces on both a normal and log-scale. In both cases we notice that the first principal component has the largest effect and the singular values exponentially decrease towards zero. The magenta points represent the number of singular values necessary to explain 50-70-90% of the data. Below we plot the percentage of components needed to explain a given amount of data within each dataset.

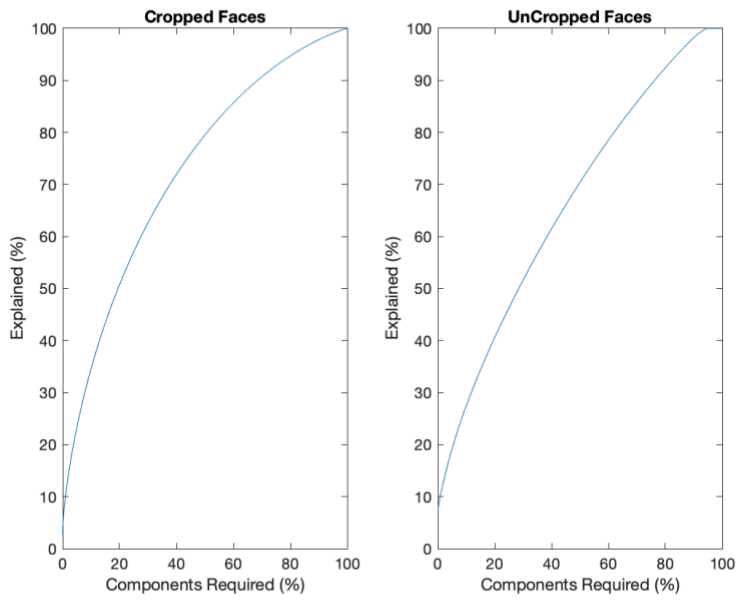
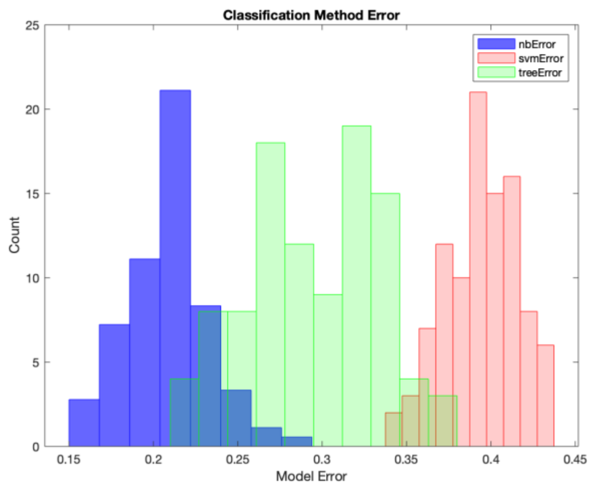


Figure (9) Components Required for Percentage Explained

Band Classification Test 1:

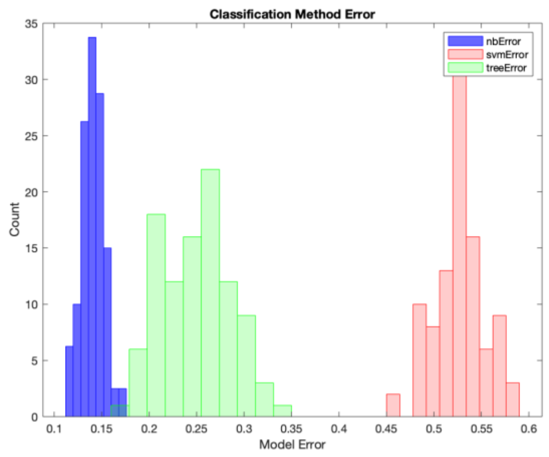


Model	Minimum	Average	Median	Max	Range	Std
Naive Bayes	17.50%	20.40%	20.42%	24.17%	6.67%	1.45%
Support Vector Machine	43.33%	55.53%	55.00%	63.33%	20.00%	3.12%
Decision Tree	25.00%	35.70%	35.42%	47.50%	22.50%	5.08%

Figure (10) Band Classification Test 1 Summary Statistics

Above we show the errors for each of the three models for Test 1 across 100 cross validated trials. We see that the Naïve Bayes model performs the best with an average error of roughly 20%. The Decision Tree Classifier identifies the correct artist roughly 2/3 of the time. Although the SVM classifies the majority of samples incorrectly, it performs better than a random guess, 33% accuracy.

Band Classification Test 2:

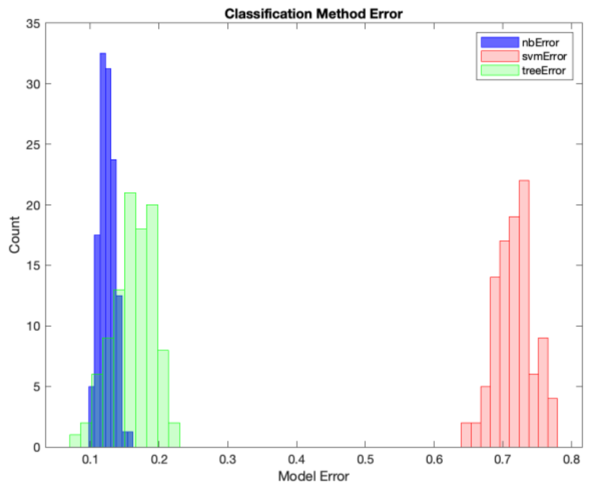


Model	Minimum	Average	Median	Max	Range	Std
Naive Bayes	11.25%	14.03%	14.17%	17.50%	6.25%	1.23%
Support Vector Machine	45.83%	52.64%	52.71%	58.75%	12.92%	2.58%
Decision Tree	16.25%	25.00%	25.00%	34.58%	18.33%	3.57%

Figure (11) Band Classification Test 2 Summary Statistics

Above we show the errors for each of the three models for Test 2 across 100 cross validated trials. We see that the Naïve Bayes model performs the best with an average error of roughly 14%. The Decision Tree Classifier identifies the correct artist roughly 75% of the time. Although the SVM classifies the majority of samples incorrectly, it performs better than a random guess: 33% accuracy. These results are surprising, as we would expect the classifier to have a more difficult time distinguishing similar music samples (each of the artists are playing Grateful Dead Songs). In fact, Dead and Company has three original members of Grateful Dead playing their own hits!

Genre Classification Test 3:



Model	Minimum	Average	Median	Max	Range	Std
Naive Bayes	10.00%	12.57%	12.50%	15.83%	5.83%	1.15%
Support Vector Machine	64.17%	71.73%	71.67%	77.92%	13.75%	2.70%
Decision Tree	7.08%	16.21%	16.25%	22.92%	15.83%	3.02%

Above we show the errors for each of the three models for Test 3 across 100 cross validated trials. We see that the Naïve Bayes model performs the best with an average error of roughly 12%. The Decision Tree Classifier identifies the correct artist roughly 85% of the time. Here we see a large discrepancy in the accuracy of the SVM. In the SVM has a worse than random chance (33%) accuracy when classifying the genre of music from the test data.

Section V: Summary and Conclusions

Within this report, we see the versatility and practicality of DWT combined with SVD analysis for machine learning applications. Use of the Haar wavelet within DWT can identify key features within images of faces, provided the input data is focused on the subject in question. Additionally, using the Haar wavelet DWT in conjunction with SVD can efficiently train some ML classifiers. In each of the three music classification tests we see that the naïve Bayes approach is the most accurate. Further improvements on this model would include more data (various artists, more songs, more genres) and combining models in an ensemble method.

Appendix A

Custom Functions

wavlet_rep takes a matrix of data, reshapes the data, and performs a discrete wavelet transformation corresponding to the name of the wavelet

```
function [cod_cH1,cod_cV1,cod_edge] = wavelet_rep(mat,c,w,x,y,n)
X = double(reshape(mat(:,c), x, y));
[cA,cH,cV,cD]=dwt2(X,w);
nbcol = size(colormap(gray),1);
cod_cH1 = wcodemat(cH,nbcol);
cod_cV1 = wcodemat(cV,nbcol);
cod_edge=cod_cH1+cod_cV1;
end
```

Appendix B

Yale Face Code

```
close all; clear all;

load Cropped.mat
load UnCropped.mat

%% faces sample
figure(1);
 for j=1:9
 subplot(3,3,j);
 face1=uint8(reshape(CroppedMat(:,j), 192, 168));
 imshow(face1);
 end
 sgtitle('Example Cropped Face')
 saveas(figure(1),[pwd '/figures/figure(1).png'], 'png');


 figure(2);
 for j=1:9
 subplot(3,3,j);
 face2=uint8(reshape(UnCroppedMat(:,j), 243, 320));
 imshow(face2);
 end
 sgtitle('Example UnCropped Face')
 saveas(figure(2),[pwd '/figures/figure(2).png'], 'png');


%% Wavelet Representations
wavelets = {'haar', 'coif1', 'sym2', 'fk4', 'dmey'};
figure(3);
for w = 1:length(wavelets)
    [cod_cH1,cod_cV1,cod_edge] = wavelet_rep(CroppedMat,2,wavelets{w},192,168,w);
    subplot(5,4,w*4-3), imshow(uint8(cod_cH1));
    subplot(5,4,w*4-2), imshow(uint8(cod_cV1));
    subplot(5,4,w*4-1), imshow(uint8(cod_edge));
    subplot(5,4,w*4), imshow(uint8(reshape(CroppedMat(:,2), 192, 168)));
end
sgtitle('Wavelet Representation Examples Cropped Face');
saveas(figure(3),[pwd '/figures/figure(3).png'], 'png');

figure(4);
for w = 1:length(wavelets)
    [cod_cH1,cod_cV1,cod_edge] = wavelet_rep(UnCroppedMat,2,wavelets{w},243,320,w);
    subplot(5,4,w*4-3), imshow(uint8(cod_cH1));
    subplot(5,4,w*4-2), imshow(uint8(cod_cV1));
    subplot(5,4,w*4-1), imshow(uint8(cod_edge));
    subplot(5,4,w*4), imshow(uint8(reshape(UnCroppedMat(:,2), 243, 320)));
end
sgtitle('Wavelet Representation Examples UnCropped Face');
saveas(figure(4),[pwd '/figures/figure(4).png'], 'png');

%% Facial SVD
feature = 20;
```

```
figure(5)
for w = 1
    CroppedMat_wave = dc_wavelet(CroppedMat, wavelets{w}, 192, 168, 8064);
    [cU,cS,cV] = svd(CroppedMat_wave, 0);
    cU = cU(:,1:feature);

    subplot(2,2,w*4-3),ut1=reshape(cU(:,1),96,84);ut2=ut1(96:-1:1,:); pcolor(ut2);
    title('Principal Component 1');
    subplot(2,2,w*4-2),ut1=reshape(cU(:,2),96,84);ut2=ut1(96:-1:1,:); pcolor(ut2);
    title('Principal Component 2');
    subplot(2,2,w*4-1),ut1=reshape(cU(:,3),96,84);ut2=ut1(96:-1:1,:); pcolor(ut2);
    title('Principal Component 3');
    subplot(2,2,w*4),ut1=reshape(cU(:,4),96,84);ut2=ut1(96:-1:1,:); pcolor(ut2);
    title('Principal Component 4');
    set(gca,'Xtick',[],'Ytick',[]);
end
sgtitle('First Principal Components Cropped Faces')
saveas(figure(5),[pwd '/figures/figure(5).png'], 'png');

figure(6)
for w = 1
    UnCroppedMat_wave = dc_wavelet(UnCroppedMat, wavelets{w}, 243, 320, 19520);
    [uU,uS,uV] = svd(UnCroppedMat_wave, 0);
    uU = uU(:,1:feature);

    subplot(2,2,w*4-3),ut1=reshape(uU(:,1),122,160);ut2=ut1(122:-1:1,:); pcolor(ut2);
    title('Principal Component 1');
    subplot(2,2,w*4-2),ut1=reshape(uU(:,2),122,160);ut2=ut1(122:-1:1,:); pcolor(ut2);
    title('Principal Component 2');
    subplot(2,2,w*4-1),ut1=reshape(uU(:,3),122,160);ut2=ut1(122:-1:1,:); pcolor(ut2);
    title('Principal Component 3');
    subplot(2,2,w*4),ut1=reshape(uU(:,4),122,160);ut2=ut1(122:-1:1,:); pcolor(ut2);
    title('Principal Component 4');
    set(gca,'Xtick',[],'Ytick',[]);
end
sgtitle('First Principal Components UnCropped Faces');
saveas(figure(6),[pwd '/figures/figure(6).png'], 'png');

%%
cFeature = 1; uFeature = 1;
cExplained = 0; uExplained = 0;
cSigma = diag(cS); uSigma = diag(uS);
percent = [0.5, 0.75, 0.95];

cFeatureCount = []; uFeatureCount = [];

for p = 1:length(percent)
while cExplained < percent(p)
    cExplained = sum(cSigma(1:cFeature))/sum(cSigma);
    cFeature = cFeature+1;
end
cFeatureCount = [cFeatureCount, cFeature];

while uExplained < percent(p)
    uExplained = sum(uSigma(1:uFeature))/sum(uSigma);
    uFeature = uFeature+1;
end
uFeatureCount = [uFeatureCount, uFeature];
end
%%
figure(7);
for i = 1:length(cFeatureCount)

subplot(3,2,i*2-1);
plot(cSigma,'ko','Linewidth',[1]);
sgtitle('Cropped Faces PCA');
title(strcat('Components Required to Explain  ', num2str(percent(i)*100), '%'), 'FontSize', [8])
hold on; plot(cSigma(1:cFeatureCount(i)), 'mo', 'Linewidth',[1]);
hold off;
% set(gca,'Fontsize',[8],'Xlim',[0 length(cSigma)]);

subplot(3,2,i*2);
semilogy(cSigma,'ko','Linewidth',[1]);
title(strcat('(Log Scale) Components Required to Explain  ', num2str(percent(i)*100), '%'), 'FontSize',
[8])
hold on; semilogy(cSigma(1:cFeatureCount(i)),'mo','Linewidth',[1]);
hold off;

% set(gca,'Fontsize',[8],'Xlim',[0 length(cSigma)]);
% sgtitle(strcat('Components Required to Explain  ', num2str(percent*100), '%'))
end
saveas(figure(7),[pwd '/figures/figure(7).png'], 'png');

figure(8);
for i = 1:length(uFeatureCount)

subplot(3,2,i*2-1);
plot(uSigma,'ko','Linewidth',[1]);
sgtitle('UnCropped Faces PCA');
title(strcat('Components Required to Explain  ', num2str(percent(i)*100), '%'), 'FontSize', [8])
hold on; plot(uSigma(1:uFeatureCount(i)), 'mo', 'Linewidth',[1]);
hold off;
% set(gca,'Fontsize',[8],'Xlim',[0 length(cSigma)]);

subplot(3,2,i*2);
semilogy(uSigma,'ko','Linewidth',[1]);
title(strcat('(Log Scale) Components Required to Explain  ', num2str(percent(i)*100), '%'), 'FontSize',
[8])
hold on; semilogy(uSigma(1:uFeatureCount(i)),'mo','Linewidth',[1]);
hold off;

% set(gca,'Fontsize',[8],'Xlim',[0 length(cSigma)]);
% sgtitle(strcat('Components Required to Explain  ', num2str(percent*100), '%'))
end
saveas(figure(8),[pwd '/figures/figure(8).png'], 'png');

%%
figure(9)
subplot(1, 2, 1); plot(100*[0:1/(length(cSigma)-1):1], 100*cumsum(cSigma)/sum(cSigma));
title('Cropped Faces'); ylabel('Explained (%)'); xlabel('Components Required (%)');
subplot(1, 2, 2); plot(100*[0:1/(length(uSigma)-1):1], 100*cumsum(uSigma)/sum(uSigma));
title('UnCropped Faces'); ylabel('Explained (%)'); xlabel('Components Required (%)');
saveas(figure(9),[pwd '/figures/figure(8).png'], 'png');

%%
figure(10)
for j=1:4
    subplot(4,2,2*j-1)
    plot(cV(1:150,j),'ko-')
    subplot(4,2,2*j)
    plot(uV(1:150,j),'ko-')
end
sgtitle('Projection onto First 3 POD Modes')
subplot(4,2,1); ylim([-0.1 0]); title('Cropped Faces');
subplot(4,2,2); ylim([-0.1 0]); title('UnCropped Faces');
saveas(figure(10),[pwd '/figures/figure(10).png'], 'png');

%% functions

function dcData = dc_wavelet(dcfile, w, x, y, nw)
[m,n]=size(dcfile);
nbcol = size(colormap(gray),1);
 for i=1:n
     X=double(reshape(dcfile(:,i), x, y));
     [cA,cH,cV,cD]=dwt2(X,w);
     cod_cH1 = wcodemat(cH,nbcol);
     cod_cV1 = wcodemat(cV,nbcol);
     cod_edge=cod_cH1+cod_cV1;
     dcData(:,i)=reshape(cod_edge,nw,1);
 end
end

function [cod_cH1,cod_cV1,cod_edge] = wavelet_rep(mat,c,w,x,y,n)
X = double(reshape(mat(:,c), x, y));
[cA,cH,cV,cD]=dwt2(X,w);
nbcol = size(colormap(gray),1);
cod_cH1 = wcodemat(cH,nbcol);
cod_cV1 = wcodemat(cV,nbcol);
cod_edge=cod_cH1+cod_cV1;
end
```

Music Code close all; clear all;

```
%% Test 1
%load TalkingHeadsSamples.mat; Artist1 = THsamples;
%load WidespreadPanicSamples.mat; Artist2 = WSPsamples;
%load MozartSamples.mat; Artist3 = Msamples;
%fignum = 1;

%% Test 2
%load GratefulDeadSamples.mat; Artist1 = GDsamples;
%load DeadAndCompanySamples.mat; Artist2 = DCsamples;
%load AlmostDeadSamples.mat; Artist3 = JRADsamples;
%fignum = 2;

%% Test 3

% Bluegrass
load BillyStringsSamples.mat; A1 = BSsamples;
load GreenskySamples.mat; A2 = GSBGsamples;
load DelMcCouryBandSamples.mat; A3 = DMBsamples;
Artist1 = [A1(:,1:33) A2(:, 1:33) A3(:, 1:34)];

% HipHop
load HipHopSamples.mat; Artist2 = HHsamples;

% Jam
load WidespreadPanicSamples.mat; A1 = WSPsamples;
load GratefulDeadSamples.mat; A2 = GDsamples;
load AlmostDeadSamples.mat; A3 = JRADsamples;
Artist3 = [A1(:,1:33) A2(:, 1:33) A3(:, 1:34)];

fignum = 3

%%
Fs = 44100;
rng(512)
ctrain = [ones(80,1); 2*ones(80,1); 3*ones(80,1)];
ttrain = [ones(100,1); 2*ones(100,1); 3*ones(100,1)];
data = [Artist1(:,1:100) Artist2(:,1:100) Artist3(:,1:100)];

L = wmaxlev(length(data), 'haar');
cmat = [];
for i = 1:300
[c,l]=wavedec(data(:,i), L, 'haar');
cmat = [cmat c];
end

[u,s,v]=svd(cmat,0);
nbError = []; svmError = []; treeError = [];
for i = 1:100
Artist1 = v(:,randperm(100));
Artist2 = v(:,randperm(100)+100);
Artist3 = v(:,randperm(100)+200);

% Train and Test Data
train = [Artist1(1:80,:); Artist2(101:180,:); Artist3(201:280,:)];
test = [Artist1(81:100, :); Artist2(181:200,:); Artist3(281:300,:)];

t = templateNaiveBayes();
prenb = fitcecoc(train,ctrain,'CrossVal','on','Learners',t);
Nerr = kfoldLoss(prenb,'LossFun','ClassifErr');
nbError = [nbError Nerr];

t = templateSVM('Standardize',true);
Mdl = fitcecoc(train,ctrain,'Learners',t);
CVMdl = crossval(Mdl);
Serr = kfoldLoss(CVMdl);
svmError = [svmError Serr];

Mdl = fitctree(train,ctrain,'CrossVal','on');
Terr = kfoldLoss(Mdl);
```

```
treeError = [treeError Terr];
end

%% Histogram
figure(fignum)
hist1= histogram(nbError,8,'Normalization','pdf','EdgeColor', 'blue', 'FaceColor',  'blue');
title('Classification Method Error'); ylabel('Count'); xlabel('Model Error');
hold on
hist2 = histogram(svmError,10, 'EdgeColor', 'red', 'FaceColor',  'red', 'FaceAlpha', 0.2);
hist3 = histogram(treeError,10, 'EdgeColor', 'green', 'FaceColor',  'green', 'FaceAlpha', 0.2);
legend();
hold off
saveas(figure(fignum), strcat('Test ', num2str(fignum),' Histograms.png'), 'png')
%% Descriptive Statistics Table
Minimum = [min(nbError); min(svmError); min(treeError)];
Average = [mean(nbError); mean(svmError); mean(treeError)];
Median = [median(nbError); median(svmError); median(treeError)];
Max = [max(nbError); max(svmError); max(treeError);];
Range = [range(nbError); range(svmError); range(treeError);];
Std = [std(nbError); std(svmError); std(treeError)];
Model = {'Naive Bayes'; 'Support Vector Machine'; 'Decision Tree'}
T = table(Model, Minimum, Average, Median, Max, Range, Std)
writetable(T, strcat('Test ', num2str(fignum), 'Descriptive Statistics.xlsx'))
```