IMAGE RECOGNITION FOR CATS AND DOGS

HYO JIN CHUNG AND MINH N. TRAN

Abstract. In this project, we are given a training set of 80 images of cats and 80 images of dogs to classify a testing set of 38 images of cats and dogs. Two different methods are used to detect the edges of the given images. First method is to apply a 2-D Discrete Haar Wavelet Transform (DWT). Second method is to apply a Laplacian filter. Then, Fisher's Linear Discriminant Analysis (FDA) is used to classify the testing images to their appropriate categories. The two methods result in classification rates of 95% and 97%, respectively.

1. Introduction and Overview

It is intriguing how we can recognize people, animals, and places. It is even more intriguing to develop a mathematical algorithm that mimics the mechanism of image recognition. To begin, let us look at how we do that biologically.

In our daily life, we take in a huge amount of information from many sources around us. Next, we form a preconception. To visually recognize images, we compare the image at hand with the ones that we have accumalated through our experience.

Let us look at Figure 1 as an example. We can easily tell that these pictures are of a dog and a bird since we already have a set of these two animals in our minds.

(a) A dog (b) A bird

Figure 1

Although these images are only composed of a few very simple lines, we are still able to recognize them since these minimally represent the two animals. They are sufficient for us to see the main features of a dog and a bird. In general, our brains find

types of images that have the same features that the unknown image has. Thus, the edges of the images take an important part of our own biological image recognition process.

In this project, we develop a mathematical algorithm to recognize the images of cats and dogs, just like what our minds do to recognize images. We are given with a training set of 80 images of cats and 80 images of dogs so that we train the computer to know what a cat and a dog should look like. Then our goal is to have the computer classify each of 38 probe images as either a cat or a dog by comparing those unknown images with the training set.

2. Theoretical Background

Before we introduce the image recognition algorithm, we review the mathematical methods that are going to be used for the algorithm.

2.1. **Discrete Wavelet Transform.** DWT is based on wavelet basis which has varying frequency and limited duration. The key advantage of wavelet transform over Fourier transform is that it captures both frequency and location information. This means that features that go undetected in one resolution can be captured in the other.

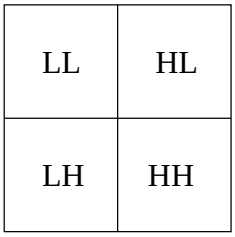


Figure 2. 2-D wavelet transform

Generally, 2-D wavelet transform takes the original image and creates four sub-images. The image in the top left is the result of reducing the resolution of the original image. This is achieved by applying the low pass filter (filter that passes low frequencies) to both rows and columns (LL). The image in the top right is the result of applying the high pass filter (filter that passes high frequencies) to the columns and low pass filter to the rows (HL). The image in bottom left is obtained via L columns and H rows (LH), and in bottom right via H columns and H rows (HH). See Figure 2.

2.2. **Average Filter.** Average filter is a smoothing spatial filter. It reduces the sharp transitions in intensities for an easier detection of edges. 2-D 3×3 smoothing spatial filter used here is the weighted average filter

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{1}$$

2.3. **Laplacian filter.** Laplacian filter is a sharpening spatial filter. Its objective is to highlight transitions in intensities. This is accomplished by taking derivatives of the intensities. The simplest isotropic derivative operator is the Laplacian

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

In discrete form, in the x-direction, we have

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y).$$

In the y-direction

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y).$$

Thus, the discrete Laplacian of two variables is

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y).$$

This gives the 2-D 3×3 sharpening spatial filter mask

$$\begin{bmatrix} & (x, y-1) & \\ (x-1, y) & (x, y) & (x+1, y) \\ & (x, y+1) & \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

The diagonal directions can be incorporated by rotating the above filter mask.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix}.$$

Another Laplacian mask is formed by adding the diagonal directions to the 2-D 3×3 sharpening spatial filter mask using the definition of the digital Laplacian

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 1 \\ 0 & -4 & 0 \\ 1 & 0 & 1 \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Thus, the 2-D 3×3 sharpening spatial filter mask used here is

$$\begin{bmatrix} (x-1, y-1) & (x, y-1) & (x+1, y-1) \\ (x-1, y) & (x, y) & (x+1, y) \\ (x-1, y+1) & (x, y+1) & (x+1, y+1) \end{bmatrix} \longrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

By multiply this mask by -1, we have our Laplacian filter mask

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{2}$$

2.4. **Principal Component Analysis.** The main step of PCA is the extraction of the eigenvalues and eigenvectors usually by Singular Value Decomposition (SVD) to form KL (Karhunen-Loéve) basis, which is well-known optimal basis that achieve dimensionality reduction of the data matrix. See Figure 3

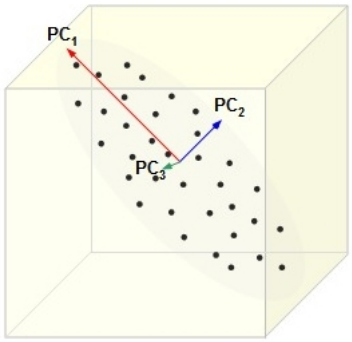


FIGURE 3. Principal Component Analysis

2.5. **Fisher's Linear Discriminant Analysis.** A simple idea of FDA is in the context of a two-class classification problem. Namely, we will consider two classes of data set and project them onto new bases. The goal is to find a suitable projection that maximizes the distance between the inter-class data while minimize the intra-class data. See Figure 4.

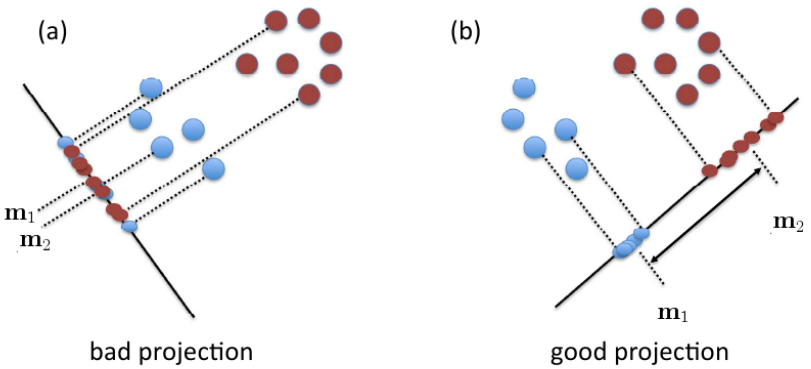


FIGURE 4. Two-class FDA

We want to construct a projection $\mathbf{w}$ such that

$$\mathbf{w} = \arg\max_{\mathbf{w}} \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

where S_B is the between-class scatter matrix and S_W is the within-class scatter matrix given by

$$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$
$$S_W = \sum_{i=1}^{2} \sum_{x \in D_i} (x - \mathbf{m}_i)(x - \mathbf{m}_i)^T$$

where $\mathbf{m}_1$ is the mean of the first class, $\mathbf{m}_2$ is the mean of the second class, and D_1, D_2 are two classes of the data set.

3. Algorithm Implementation and Development

The main idea of the image recognition process is that we first separate the training set of images into cats and dogs, and then we see which group each of the probe images belongs to.

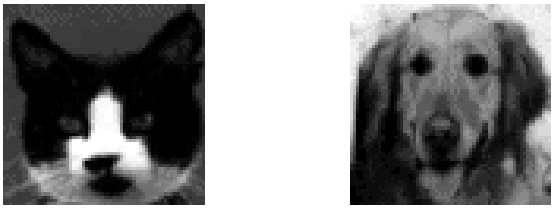
We develop an algorithm which classifies the images in the training set into two classes of cats and dogs, and the similar algorithm would be used to find the membership of the probe images.

As we have seen before, the detection of edges is an important integral part of image recognition process. Hence, the first step of the 2-class classification algorithm is to extract the edges from the original images so that it is easier to recognize the main features of the images. In this project, we introduce two different methods for edge detection which are DWT and Laplacian Filter. Then, to consider only the main features of the images for the image recognition process, the second step is to do the dimensionality reduction using PCA. Finally, FDA is used for 2-class classification.

Edge Detection
$\Downarrow$
Dimension Reduction
$\Downarrow$
Classification

Once we find a threshold value using the training set, which best separates the two classes of data, we move on to the next part of the image recognition process. We now take the probe images and go through the same classification algorithm that we used for the training set. Then, for example, using the threshold, any probe image that is placed in the class of cat would be recognized as an image of cat.

To illustrate the algorithm implementation, let us consider the first picture of a cat and the first picture of a dog from the cat and dog data set. See Figure 5.



(A) Cat00 (B) Dog01

FIGURE 5. Original images

3.1. **Method 1: Using 2-D Discrete Haar Wavelet Transform.**

2-D Discrete Haar Wavelet Transform
$\Downarrow$
Principal Componant Analysis
$\Downarrow$
Fisher's Linear Discriminant Analysis

3.1.1. *2-D Discrete Haar Wavelet Transform.* To extract the edges, we take the Haar wavelet transform of the cat and dog. See Figure 6 and 7. The figure on the top left is the reduced resolution of the original image. The one on the top right is the fine scales in the horizontal direction, bottom left is fine scales in the vertical direction, and bottom right is the fine scales in the diagonal direction.

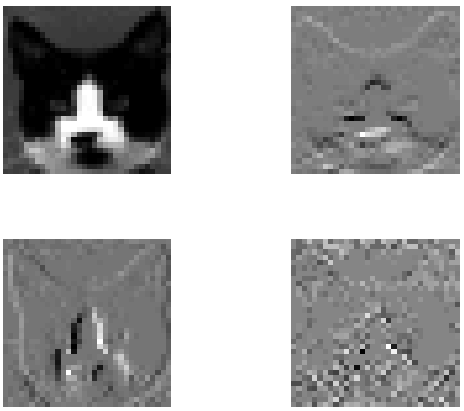


FIGURE 6. Haar wavelet transform of Cat00

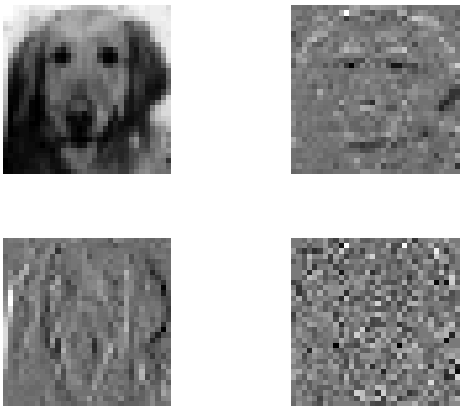
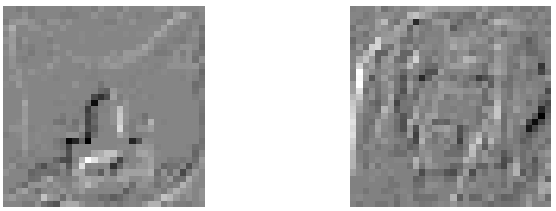


FIGURE 7. Haar wavelet transform of Dog01

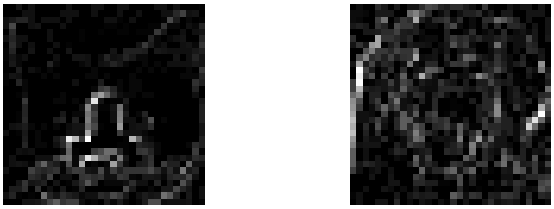
After decomposing the image using Haar wavelet transform, we extract the edges by adding the fine scales in the horizontal and vertical direction. Adding these two details is good enough since horizontal fine scales are obtained by applying HL and vertical fine scales are obtained by applying LH. The frequencies that go undetected in one direction are captured in the other direction. See Figure 8. The original images are 64-by-64. We notice that the images after DWT are now 32-by-32.



(A) Cat00 (B) Dog01

FIGURE 8. After adding vertical and horizontal details

The details in the images are not standing out very clearly at this stage since the color intensities have not been rescaled yet. After rescaling the color of the horizontal and vertical fine scales of the cat and dog to the range from 0 to 255, the edges are now more apparent. See Figure 9. Now we can better see the cat features with pointed ears in the image on the left and the dog features with rounded head in the image on the right.



(A) Cat00 (B) Dog01

FIGURE 9. Vertical and horizontal details after color mapping

3.1.2. *Principal Component Analysis.* Before we move on to the classification step of the algorithm, we perform SVD on the set of images to find the principal components of the two animals. See Figure 10. The first four principal components of cat and dog data give the four main features of the animals. We see that there is a strong cat resemblance with the pointed ears in the components. Thus, there will be more chances that we will classify cats correctly.

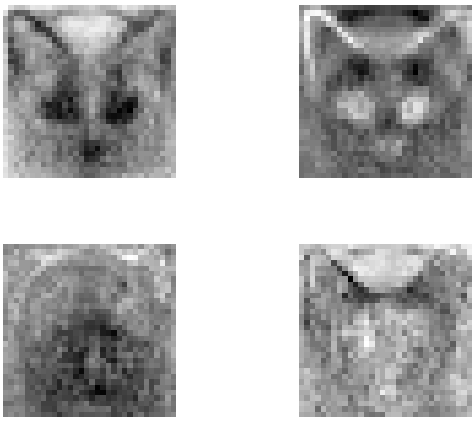


FIGURE 10. First four principal components of cat and dog data

A plot of the eigenvalues after SVD shows a heavy tail distribution. See Figure 11. To capture the essential components of the cats and dogs, we only need to retain a few features. In this case, we choose our number of features to be 20.

From the matrices S and V produced by SVD, we can lower the dimension of the cat and dog data to efficiently classify them. And we can use the eigenvectors U to project the probes onto this lower dimension.

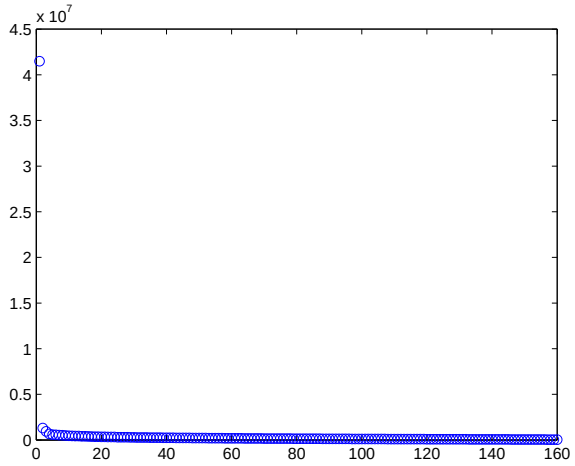
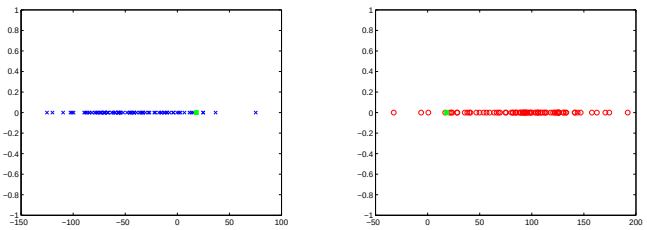


FIGURE 11. Eigenvalues of the cat and dog data

3.1.3. *Fisher's Linear Discriminant Analysis.* We now turn our attention to the statistical aspect of the project. Using the method of FDA, we find the within class variance matrix and between class variance matrix. From here we find the optimal projection w to project the data onto the real line where we actually do the classification. To do that, we find a threshold value that best separates the two classes of data.

Figure 12 shows the projection of the cat and dog data down onto the real line and the threshold value. With the training data, we have about 3-4 misclassifications for each cats and dogs.



(A) Dog data (B) Cat data

FIGURE 12. Projection of cat and dog data onto the real line

To have a better understanding of the distribution of the data, we generate the histogram of the cats and dogs with the separation indicator line in red. See Figure 13.

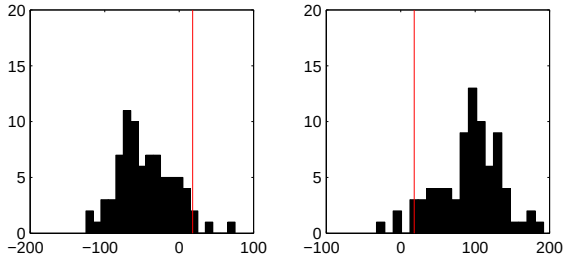
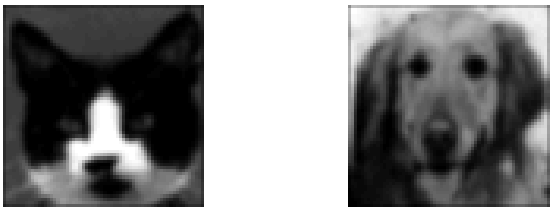


FIGURE 13. Histogram of cat and dog data after being projected onto the real line

3.2. **Method 2: Using Laplacian Filter.**

Laplacian Filter
$\Downarrow$
Principal Componant Analysis
$\Downarrow$
Fisher's Linear Discriminant Analysis

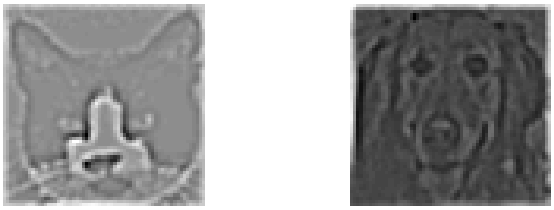
3.2.1. *Laplacian Filter.* The second method uses the Laplacian filter for edge detection. Before we apply the Laplacian filter on the image, we first smooth out the image by applying the average filter in Equation (1). This process blends any small objects with the background and makes the larger objects easy to detect. See Figure 14.



(A) Cat00 (B) Dog01

FIGURE 14. After applying average filter

We then extract the edges by applying the Laplacian filter in Equation (2). See Figure 15. We can see that edges are more apparent now. The entire images are very gray so that the edges now stand out better. The ears of the cat are staning out better from the background and the whisker of the cat is more defined. Also, the ears and the mouth of the dog are better shown now. Since the Laplacian filter does not change the resolution, the images after edge detection look as detailed as the original ones.



(A) Cat00 (B) Dog01

FIGURE 15. After applying Laplacian mask

The resulting figure in the right is darker than the one in the left. To have consistency in color, again we map the color intensity of this figure to the scale from 0 to 255. See Figure 16.

(A) Cat00 (B) Dog01

FIGURE 16. After color rescaling

3.2.2. *Principal Component Analysis.* Before classifying the images into cats and dogs, we use the idea of PCA to find the best basis to project the data onto a lower dimension for less computations. We perform SVD to find the best D value, which is a new dimension of the data. The first four principal components after applying SVD is shown in Figure 17. This best D value provide an even lower dimension than the one that we first project the original data onto. Classifying in this dimension is more advantageous since most of the important details are retained and the computational

cost is further reduced. In this case, we set the energy level to about 95% to find the best D value.

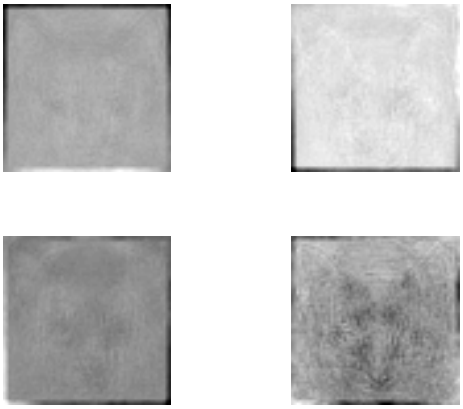


FIGURE 17. First four principal components of cat and dog data

The best D value found here is 121. In Figure 18 the plot of the eigenvalues after applying SVD shows that we need to use most of the eigenvalues. However, comparing to the dimension of 160, the dimension of 121 is more efficient for further computation and still retains the important components of the data.

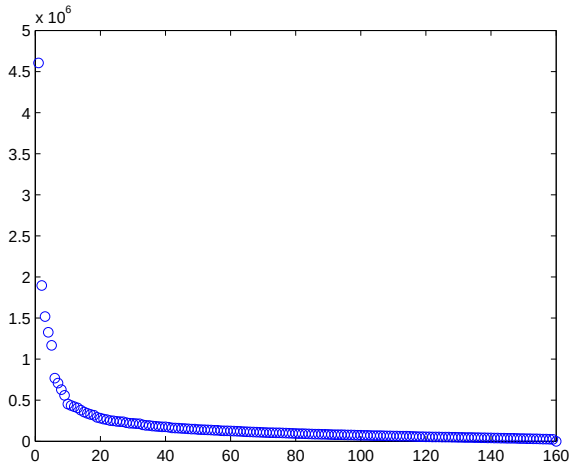
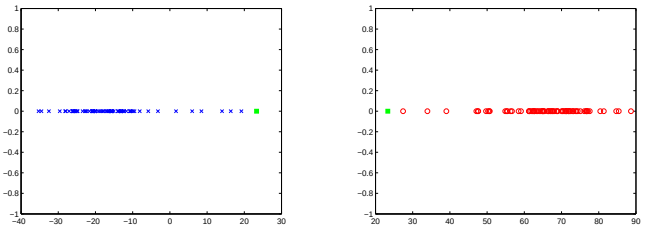


FIGURE 18. Eigenvalues of the cat and dog data

Similar to the PCA step from the previous method, the first 121 eigenvectors U are used to project the data onto this lower dimension.

3.2.3. *Fisher's Linear Discriminant Analysis.* Like the first method, we use FDA for the classification step. Using the method of FDA we find the optimal projection direction w to project the data onto the real line.

After projecting the cat and dog data down onto the real line and obtaining the threshold value, we generate a plot. See Figure 19. All of the cats and the dogs in the data are correctly classified. We generate the histogram of the cats and dogs with the indicator line in red. See Figure 20.



(A) Dog data (B) Cat data

FIGURE 19. Projection of cat and dog data onto the real line

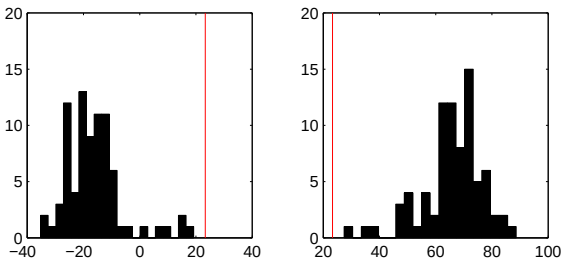
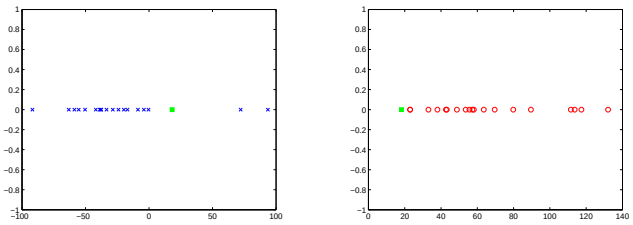


FIGURE 20. Histogram of cat and dog data after being projected onto the real line

4. Computational Results

4.1. **Result of Method 1 using 2-D Discrete Haar Wavelet Transform.** After running the cat and dog probes through the algorithm and projecting them onto the real line, we get the results below. See Figure 21.



(A) Dog probes (B) Cat probes

FIGURE 21. Projection of cat and dog probes onto the real line

A histogram of the probes are displayed below. See Figure 22.

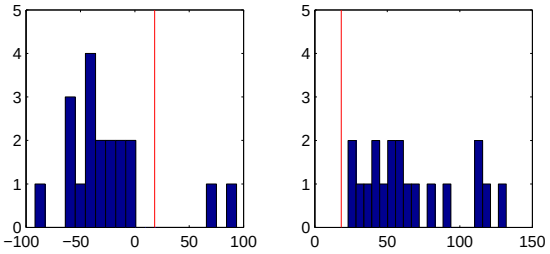


FIGURE 22. Histogram of cat and dog probes after being projected onto the real line

The confusion matrix is shown below. The classification rate is 94.74%.

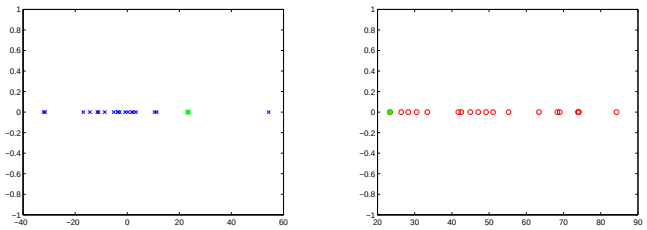
		Actual	
		cat	dog
Classified	cat	19	2
	dog	0	17

The misclassified dogs are shown below. See Figure 23. These two dogs are classified as cats because they have the pointed ears which is a feature of cats.

(A) (B)

FIGURE 23. Misclassified Dogs

4.2. **Result of Method 2 using Laplacian Filter.** After running the cat and dog probes through the algorithm and projecting them onto the real line, we get the results below. See Figure 24. We see that only one dog is misclassified out of 38 cats and dogs.



(A) Dog probes (B) Cat probes

FIGURE 24. Projection of cat and dog probes onto the real line

A histogram of the probes are displayed below. See Figure 25.
The confusion matrix is shown below. The classification rate is 97.37%.

		Actual	
		cat	dog
Classified	cat	19	1
	dog	0	18

The misclassified dog is shown below. See Figure 26. This dog is classified as a cat because it has the pointed ears feature of cats. This dog is one of the misclassified dogs from Method 1.

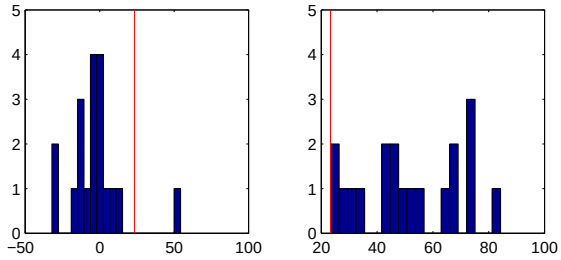


FIGURE 25. Histogram of cat and dog probes after being projected onto the real line

FIGURE 26. Misclassified Dog

5. SUMMARY AND CONCLUSIONS

In summary, to classify the images of cats and dogs, we used two different methods for edge detection which are 2-D Discrete Wavelet Transform and Laplacian filter. Then FDA was used to recognize the images as either cats or dogs. The first method using 2-D Discrete Wavelet Transform resulted in the classification rate of 95%. The second method using Laplacian filter produced the classification rate of 97%. The higher classification rate of the second method is probably because in the edge detection process, the original resolution is maintained (64-by-64), whereas the one in the first method, the resolution is reduced (32-by-32). Comparing Figure 8 and 15, we can see that Laplacian mask does a better job at detecting the image edges. Also, the second method retains a higher number of dimensions (121) than the first method (20). However, due to the fact that the first method takes in a lower dimension and retains smaller number of dimension, it is faster than the second method. Thus, depending on the situation, if high classification rate is desired, then the second method

should be used. If time is an issue, then the first method should be used. Moreover, depending on the situation one method might work better than the other. Thus, deciding which method to use is very important.

We should note that this algorithm only works for two classes of images. For multiclass recognition, a more sophisticated algorithm should be devised. Also, we should keep in mind that a hundred percent correct classification is very rare especially when we encounter dogs that share features of cats and vice versa.

Appendix A. MATLAB functions used and brief implementation explanation

A.1. **colormap.** colormap(gray) sets the current figure's colormap to gray. colormap(gray) is used to set the colors in black and white.

A.2. **conv2.** conv2 performs the 2-D convolution of the input matrices. conv2(A,B,'same') is used to apply the average filter and Laplacian filter where A is a matrix of an image and B is a filter.

A.3. **dwt2.** dwt2 performs a single-level 2-D wavelet decomposition with respect to a particular wavelet. dwt2 is used to decompose the images of cats and dogs into approximation and details.

A.4. **repmat.** B = repmat(A,M,N) creates a large matrix B consisting of an M-by-N tiling of copies of A. The size of B is [size(A,1)*M, size(A,2)*N]. repmat is used to subtract the mean of the data from each column of the data matrix.

A.5. **reshape.** reshape(X,M,N) returns the M-by-N matrix whose elements are taken columnwise from X. reshape is used to store a matrix form of an image into a column or vice versa.

A.6. **svd.** [U,S,V] = svd(X) produces a diagonal matrix S, of the same dimension as X and with nonnegative diagonal elements in decreasing order, and unitary matrices U and V so that X = U*S*V'. [U,S,V] = svd(X,0) is used for Singular Value Decomposition which produces the "economy size" decomposition. svd is used for dimensionality reduction.

A.7. **wcodemat.** wcodemat is an extended pseudocolor matrix scaling. Y = wcodemat(X,NBCODES) where X is a matrix and NBCODES is the color range, rescales the values in matrix X to the value range from NBCODES. wcodemat is used to rescale colors to appropriate pseudocolor scaling after edge detection.

Appendix B. MATLAB codes

B.1. **MATLAB codes for the method of using Discrete Haar Wavelet Transform and Fisher's Linear Discriminant Analysis.**

B.1.1. *wavelet.*

```
function [AA] = wavelet(A,n)
%--------------------------------------------------------------------------
% Purpose:
% To decompose an image into an approximation and 3 detail components in an
% arrangement
%  LL  HL
%  LH  HH
% where L=low, H=high
%
% Inputs:
% A - matrix form of an image
% n - level of decomposition
%
% Output:
% AA - matrix form of new image
%--------------------------------------------------------------------------

nbcol = size(colormap(gray),1);
cA = A;

for i=1:n
 [cA,cH,cV,cD] = dwt2(cA,'haar');

 % rescale to appropriate pseudocolor scaling
 cod_cH = wcodemat(cH,nbcol);
 cod_cV = wcodemat(cV,nbcol);
end
AA = cod_cH+cod_cV;
```

B.1.2. *FDA.*

```
function [U,vcat,vdog,threshold,w] = FDA(cat0,dog0,feature)
%--------------------------------------------------------------------------
% Purpose:
% To classify using 2-class Fisher Discriminant Analysis (FDA)
%
% Inputs:
% cat0 - data matrix of cats
```

```
% dog0 - data matrix of dogs
% feature - number of components to be considered
%
% Outputs:
% U - eigenvectors to reduce dimension of probe
% vcat - projection of cats onto the real line
% vdog - projection of dogs onto the real line
% threshold - separation indicator between cats and dogs
% w - projection vector
%--------------------------------------------------------------------------

nc = length(cat0(1,:));
nd = length(dog0(1,:));

[U,S,V] = svd([cat0 dog0],0);

coeff = S*V';
U = U(:,1:feature);
cats = coeff(1:feature,1:nc);
dogs = coeff(1:feature,nc+1:nc+nd);

mc = mean(cats,2);
md = mean(dogs,2);

SW = 0;  % within class variances
for i=1:nc
 SW = SW + (cats(:,i)-mc)*(cat~\ref{Fi:Mask cat dog}s(:,i)-mc)';
end
for i=1:nd
 SW = SW + (dogs(:,i)-md)*(dogs(:,i)-md)';
end

SB = (md-mc)*(md-mc)';  % between class variances
[V2,D] = eig(SB,SW);
[lambda,ind] = max(abs(diag(D)));
w = V2(:,ind);
w = w/norm(w,2);

vdog = w'*dogs;
vcat = w'*cats;

if mean(vdog)>mean(vcat)
```

```
 w = -w;
 vdog = -vdog;
 vcat = -vcat;
end

sortdog = sort(vdog);
sortcat = sort(vcat);
 very simple. They are only
t1 = length(sortdog);
t2 = 1;
while sortdog(t1)>sortcat(t2)
 t1 = t1-1;
 t2 = t2+1;
end
threshold = (sortdog(t1)+sortcat(t2))/2;
```

B.1.3. *waveFDA.*

```
function [clsfy] = waveFDA(cats,dogs,probes,feature)
%-------------------------------------------------------------------------
% Purpose:
% To classify cats and dogs
%
% Inputs:
% cats - matrix of images of cats
% dogs - matrix of images of dogs
% probes - matrix of images of cats and dogs for testing
% feature - number of components to retain
%
% Outputs:
% clsfy - classification vector
%
% Functions called:
% wavelet
% FDA
%-------------------------------------------------------------------------

nc = length(cats(1,:));
nd = length(dogs(1,:));
np = length(probes(1,:));

for i=1:nc
 cat0(:,:,i) = reshape(cats(:,i),64,64);  % cats are of size 64x64
```

```
end
for i=1:nd
 dog0(:,:,i) = reshape(dogs(:,i),64,64);  % dogs are of size 64x64
end
for i=1:np
 probe0(:,:,i) = reshape(probes(:,i),64,64);  % probes are of size 64x64
end

% decomposition using wavelet
n = 1;  % level of decomposition
for i=1:nc
 data = wavelet(cat0(:,:,i),n);
 cat0_res(:,i) = reshape(data,size(data,1)*size(data,2),1);
end
for i=1:nd
 data = wavelet(dog0(:,:,i),n);
 dog0_res(:,i) = reshape(data,size(data,1)*size(data,2),1);
end
for i=1:np
 data = wavelet(probe0(:,:,i),n);
 probe0_res(:,i) = reshape(data,size(data,1)*size(data,2),1);
end

% 2-class classification
[U,vcat,vdog,threshold,w] = FDA(cat0_res,dog0_res,feature);

test = U'*probe0_res;
pval = w'*test;

clsfy = zeros(1,np);
for i=1:np
 if pval(i)>threshold  % dogs<threshold<cats
  clsfy(i) = 1;  % classify as cats
 end
end
```

B.1.4. *testcode.*

```
clear

% Read in data
cd TIFFtraining
fname = ls;
```

```
N = length(fname);
for i=3:N
 data = imread(fname(i,:));
 data = double(data);
 data = data(:,:,1);
 Dmat(:,i-2) = reshape(data,size(data,1)*size(data,2),1);
end
cd ..

% Training Set
cats = Dmat(:,1:80);
dogs = Dmat(:,81:160);

% Testing Set
load PatternRecAns
probes = TestSet;

% Number of features
feature = 20;

% Classify the probes as cats or dogs
clsfy = waveFDA(cats,dogs,probes,feature);
counter = abs(clsfy - hiddenlabels);
```

B.2. **MATLAB codes for the method of using Laplacian Filter and Fisher's Linear Discriminant Analysis.**

B.2.1. *mask.*

```
function [AA] = mask(A)
%-------------------------------------------------------------------------
% Purpose:
% To extract the edges(outline) of the image
%
% Input:
% A - matrix of image
%
% Output:
% AA - matrix of image after filter and mask
%-------------------------------------------------------------------------

% The weighted average filter
filter = (1/16)*[1 2 1 ; 2 4 2 ; 1 2 1];
```

```
% The Laplacian mask
mask = [-1 -1 -1 ; -1 8 -1 ; -1 -1 -1];

Af = conv2(A,filter,'same');
Afm = conv2(Af,mask,'same');

nbcol = size(colormap(gray),1);
AA = wcodemat(Afm,nbcol);
```

B.2.2. *maskFDA.*

```
function[clsfy] = maskFDA(cats,dogs,probes,energy)
%-------------------------------------------------------------------------
% Purpose:
% To classify cats and dogs
%
% Inputs:
% cats - matrix of images of cats
% dogs - matrix of images of dogs
% probes - matrix of images of cats and dogs for testing
% energy - energy level for PCA
%
% Outputs:
% clsfy - classification vector
%
% Functions called:
% mask
%-------------------------------------------------------------------------

nc = size(cats,2);
nd = size(dogs,2);
np = size(probes,2);

% Reshape the images back to 64*64
for i=1:nc
    cat(:,:,i) = reshape(cats(:,i),64,64);
end
for i=1:nd
    dog(:,:,i) = reshape(dogs(:,i),64,64);
end
for i=1:np
    probe(:,:,i) = reshape(probes(:,i),64,64);
end
```

```
% Apply the filter and the mask to extract the edges
for i=1:nc
    data = mask(cat(:,:,i));
    cats(:,i) = reshape(data,size(data,1)*size(data,2),1);
end
for i=1:nd
    data = mask(dog(:,:,i));
    dogs(:,i) = reshape(data,size(data,1)*size(data,2),1);
end
for i=1:np
    data = mask(probe(:,:,i));
    probes(:,i) = reshape(data,size(data,1)*size(data,2),1);
end

D = [cats dogs];
D = D - repmat(mean(D,2),[1,nc+nd]); % mean subtracted
[U S V] = svd(D,0);

% Find the best d value to use
Svec = diag(S);
totalenergy = dot(Svec,Svec);
currentenergy = 0;
for d = 1 : length(Svec)
    currentenergy = currentenergy + ((Svec(d))^2) / totalenergy;
    if currentenergy >= energy
        break
    end
end
d
% PCA
Ud = U(:,1:d);
qc = Ud'*cats;
qd = Ud'*dogs;
qp = Ud'*probes;

% Compute the optimal projection direction, w
A = S(1:d,1:d)*V(:,1:d)';
cat = A(:,1:nc);
dog = A(:,nc+1:nc+nd);

mc = mean(cat,2);
```

```
md = mean(dog,2);

SB = (md-mc)*(md-mc)';

M1 = repmat(mc,[1,nc]);
M2 = repmat(md,[1,nd]);

SW1 = (A(:,1:nc) - M1) * (A(:,1:nc) - M1)';
SW2 = (A(:,nc+1:nc+nd) - M2) * (A(:,nc+1:nc+nd) - M2)';

SW = SW1 + SW2;

[V2,d] = eig(SB,SW);
[l,ind] = max(abs(diag(d)));
w = V2(:,ind);
w = w/norm(w,2);

% Project the data and the probe onto a line
Cat = w'*qc;
Dog = w'*qd;
Probe = w'*qp;

if mean(Dog) > mean(Cat)
    w = -w;
    Dog = -Dog;
    Cat = -Cat;
    Probe = -Probe;
end

sortDog = sort(Dog);
sortCat = sort(Cat);

t1 = length(sortDog);
t2 = 1;
while sortDog(t1) > sortCat(t2)
    t1 = t1 - 1;
    t2 = t2 + 1;
end

threshold = (sortDog(t1)+sortCat(t2))/2;

clsfy = zeros(1,length(Probe));
```

```
for i = 1 : length(Probe)
    if Probe(i) > threshold
        clsfy(i) = 1;
    end
end
```

B.2.3. *testcode.*

```
clear

% Read in data
cd TIFFtraining
fname = ls;
N = length(fname);
for i = 3:N
    data = imread(fname(i,:));
    data = double(data);
    data = data(:,:,1);
    Dmat(:,i-2) = reshape(data,size(data,1)*size(data,2),1);
end
cd ..

% Training Set
cats = Dmat(:,1:80);
dogs = Dmat(:,81:160);

% Testing Set
load PatternRecAns
probes = TestSet;

% Energy level for PCA
energy = 0.95;

% Classify the probes as cats or dogs
clsfy = maskFDA(cats,dogs,probes,energy);
counter = abs(clsfy - hiddenlabels);
```

References

[1] J. M. Chang, *Matrix Methods for Geometric Data Analysis and Pattern Recognition*, classnotes.
[2] J. Nathan Kutz, *AMATH 582, Computation Methods for Data Analysis*, classnotes.
[3] M. Kirby, *Geometric Data Analysis: An Empirical Approach to Dimensionality Reduction and the Study of Patterns*, Wiley & Sons, 2001.