Tumor Classification

Conor Schleicher & Han Song
University of Washington
Department of Applied Mathematics
Seattle, WA

Abstract: In this paper, we are going to apply FFT, Haar wavelets decomposition, SVD and LDA on a machine learning algorithm to classify brain MRI images and to detect whether MRI belongs to a healthy person or otherwise. The algorithm is applied on a dataset with over 300 images with a mix of normal brain MRI and brain with tumor MRI. Our algorithm achieved 80% accuracy

Keywords: SVD, FFT, FFTSHIFT, SVM, classification. GitHub Repo: Tumor-Classification

I. Introduction and Overview:

Tumor is an abnormal mass tissue results from either rapid growth of cells or cells do not die when they should. Tumor can be benign (not cancer) or malignant (cancer) [1]. The oldest recorded description about cancer dated back to 3000 BC in Egypt. From this documentation, cancer was called the Edwin Smith Papyrus and the writing also said, "there is no treatment." Fast forward to the twenty first century, human had made remarkable progress in early detection, prevention, and treatment. We would like to emphasize on early detection process as most people who experienced cancer, don't develop symptom until it's late in the game. Being cautious, people start to get annual check-up and X-ray done. In this paper, we are going to develop an AI to determine whether an X-ray photo is showing sign of a tumor. We are going to build a photo library and implement function such as Fast Fourier Transform (FFT), Singular Value Decomposition (SVD), and Linear Discrimination Analysis (LDA).

II. Theoretical Background:

1. SVD

Every matrix **A** has a singular value decomposition and it takes form $\boldsymbol{A} = \boldsymbol{U\Sigma V}^*$. Each of these matrix components has specific meaning to matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$**.** **U** matrix is a unitary matrix consists of n orthonormalized eigenvectors of AA^Twhich will rotate matrix **A** so that its axis coincide the principle coordinates,$\boldsymbol{\Sigma}$ consists of non-negative square roots of the eigenvalues of $\boldsymbol{A^T A}$ in a descending order which will "modify" the rotated axis to scale with the principle coordinates and lastly $\boldsymbol{V}$ matrix is a unitary matrix consists of n orthonormalized eigenvectors of $\boldsymbol{A^T A}$ which will rotate the data point on the principle axis.

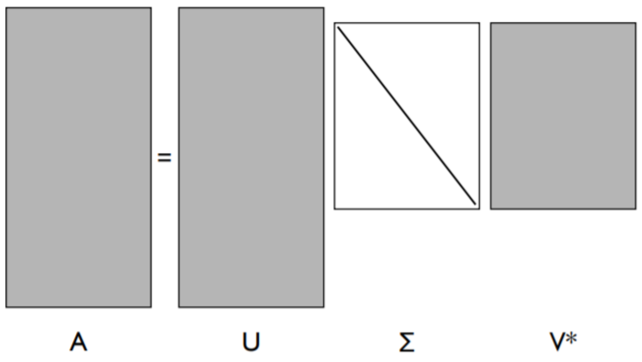

Figure 1: Graphical description of SVD (Credit: Professor Kutz)

2. Wavelets decomposition and LDA:

Wavelets decomposition and LDA are statistical learning (machine learning) techniques. Wavelets is good at detecting edge of an object in images. In this paper, we're using single-level discrete 2-D wavelet transform with respect to Haar wavelets (square shape).

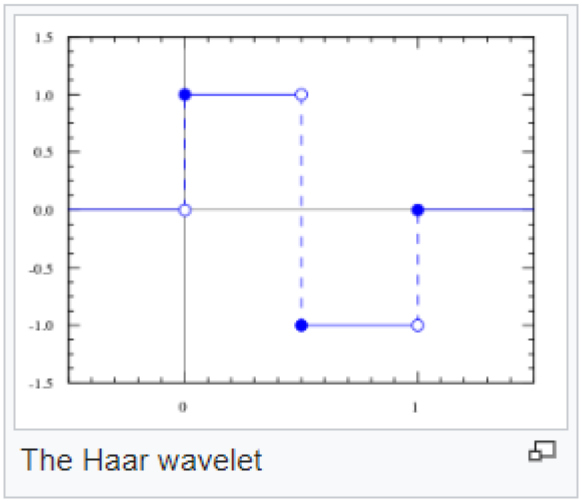

Figure 2: The Haar wavelet

On the other hand, LDA helps to understand the statistical distinction between 2 datasets. LDA can decide which threshold it should project the test iteration. The idea to pick the mode where two dataset present the most distinction.

Reference from professor Kutz [2], the algorithm should follow the following key steps:

Step 1: Decompose images into wavelet basis function
Step 2: Form the wavelet expanded images, find the principle components associatied (PCA).
Step 3: Perform LDA
Step 4: Test efficiency of the algorithm

III. Algorithm Implementation and Development:

1. Functions:

First function is called *tc_wavelet.m*. In this function, we parse in the training set matrix as well as row and column size of all images to extract the Haar wavelets decomposition matrix and produce the edge detection for each image in all matrices.

Second function is *tc_trainer.m*. This function makes statistical decision of which threshold to use based on the distinction obtain from tc_wavelet function.

2. Main algorithm:

X-ray images were collected into a tumor present (yes folder) or a healthy person (no folder). Because all images' sizes are not uniform, we grabbed the smallest image from both folders to be the comparison image:

```
% find name of smallest image
if val < val2
    smallestimage = fullfile(yesfolder,yesfiles(idx).name);
else
    smallestimage = fullfile(nofolder,nofiles(idx2).name);
end
```

We then go through all images in both folders to scale all to the small image above. Since tumor appear on X-ray, it will likely being lit up or have a brighter color than black, transformation to frequency domain is needed to detect those tumors. In frequency domain, all data is put into a big matrix **X** which will go through wavelet transform or SVD transform. A subset of matrix X will become test data and the rest of the columns are consider training data. Once we have uniform size matrices for yes, no, and test dataset, we parse them to tc_wavelet.m and tc_trainer.m and graph our results.

IV. Computational Results and Analysis:

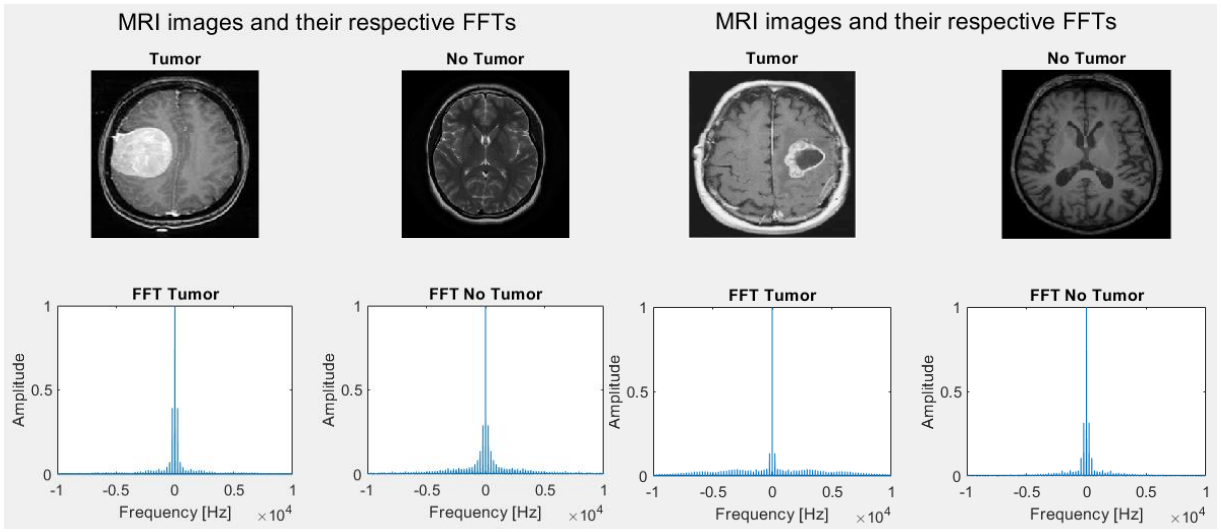

Figure 3: FFT of random samples of tumor and no tumor MRI. Note we can observe that Tumor MRI samples have longer FFT tails.

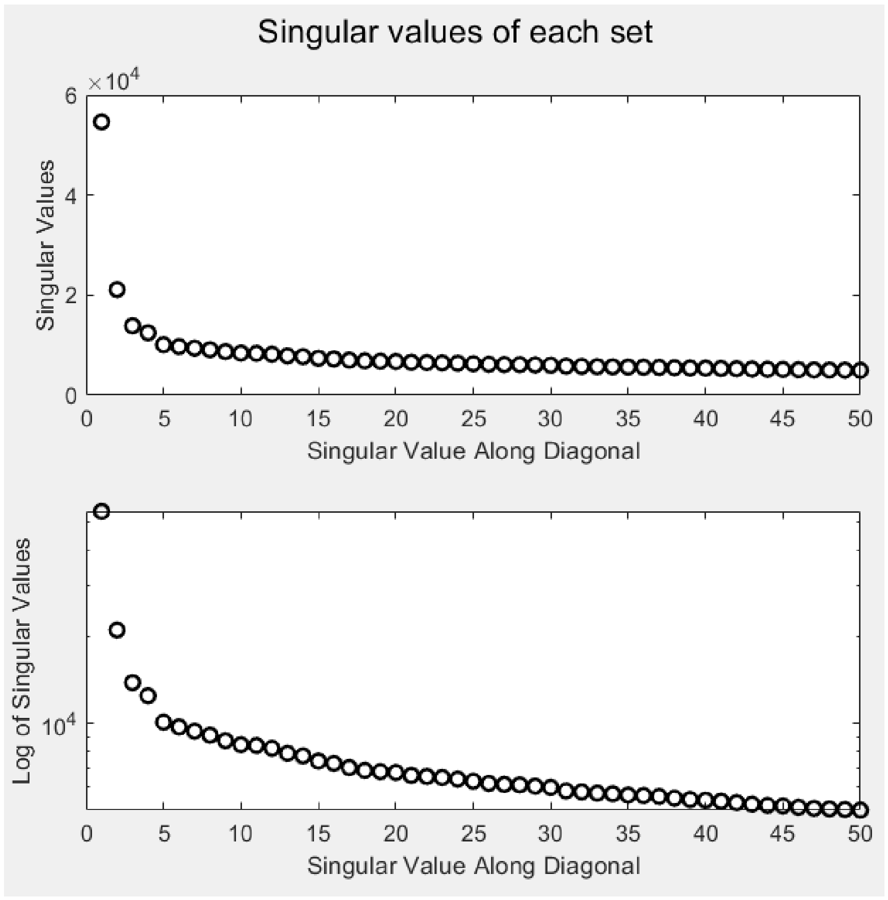

Figure 4: Singular Spectrum of each sets. Normal (top) and log scale (bottom)

There is one mode dominate the singular value and the heavy-tail distribution. The first 4 modes can generate most of the energy of the data set.

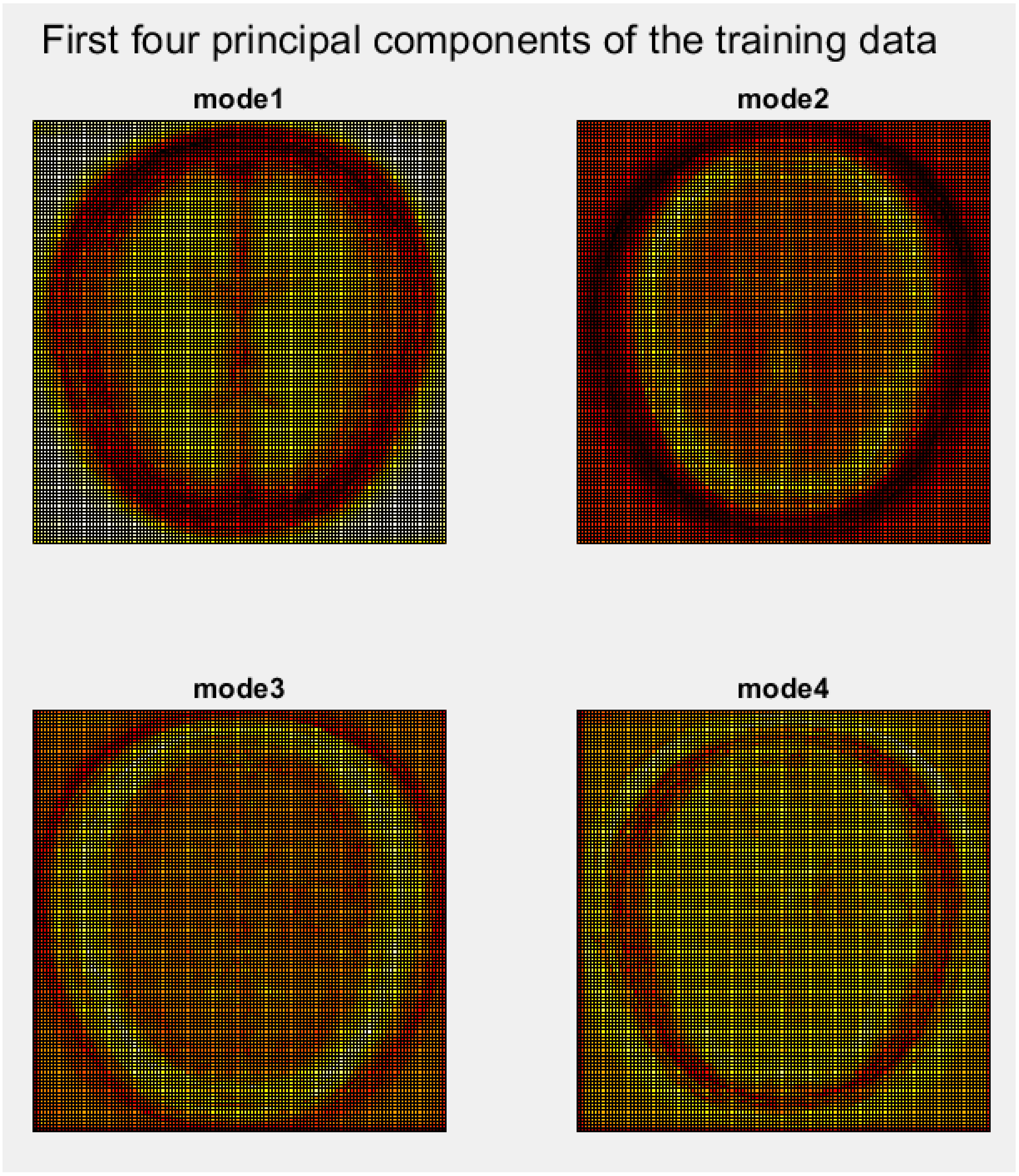

Figure 5: Features highlighted in each dominant mode

Figure 4 showed what feature was detected in each mode. In the case. Mode 1 detected the skull shape and cerebral brain, mode 2 detected the gray matter, mode 3 found gray matter wrinkle and mode 4 detected the region between gray matter and skull. Since tumor grows from gray matter, we can most likely detect it from mode 2.

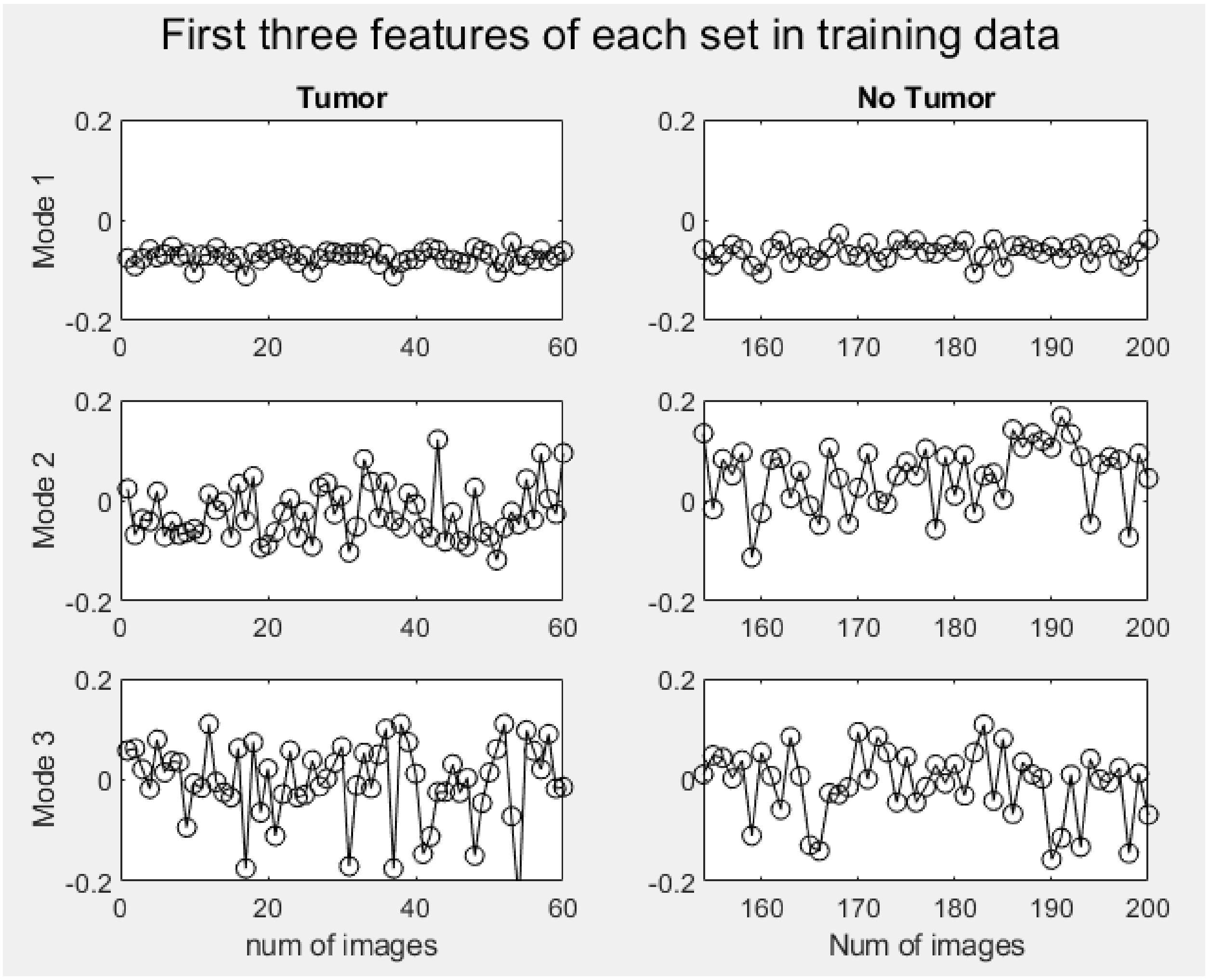

Figure 6: Projections of some images from both training datasets described by the SVD matrix V. Tumor MRI is on the left column and no tumor MRI is on the right

Just like we expect, everyone share mode 1 because of the same skull shape, but mode 2 of tumorous MRI mostly stays in the negative side while no tumor images are mostly positive. We can conclude that our desired POD is mode 2. Applying the LDA algorithm on the MRI images in the training dataset, we obtained an accuracy of 80% after cross validation from 20 new runs. Since most of the data was contained in only the first few primary modes, only 10 features were used. A few items might contribute to error reduction are:

- Images contrast level: just like any form of photograph, different MRI or different MRI technician will produce different color theme.
- Whether pictures are collected straight from MRI, screenshot or a retake from a photo will create a discrepancy in picture angle, information appear on image.
- Image sizes rescaling can distort details afterward.

The wavelet/LDA approach was tested against a FFT/SVM approach and the LDA method performed much better. The SVM algorithm was only able to achieve 50-60% accuracy of detecting the tumor from the FFT'd images.

Overall, this algorithm might work better when installing into an MRI machine as a tumor detector built-in function.

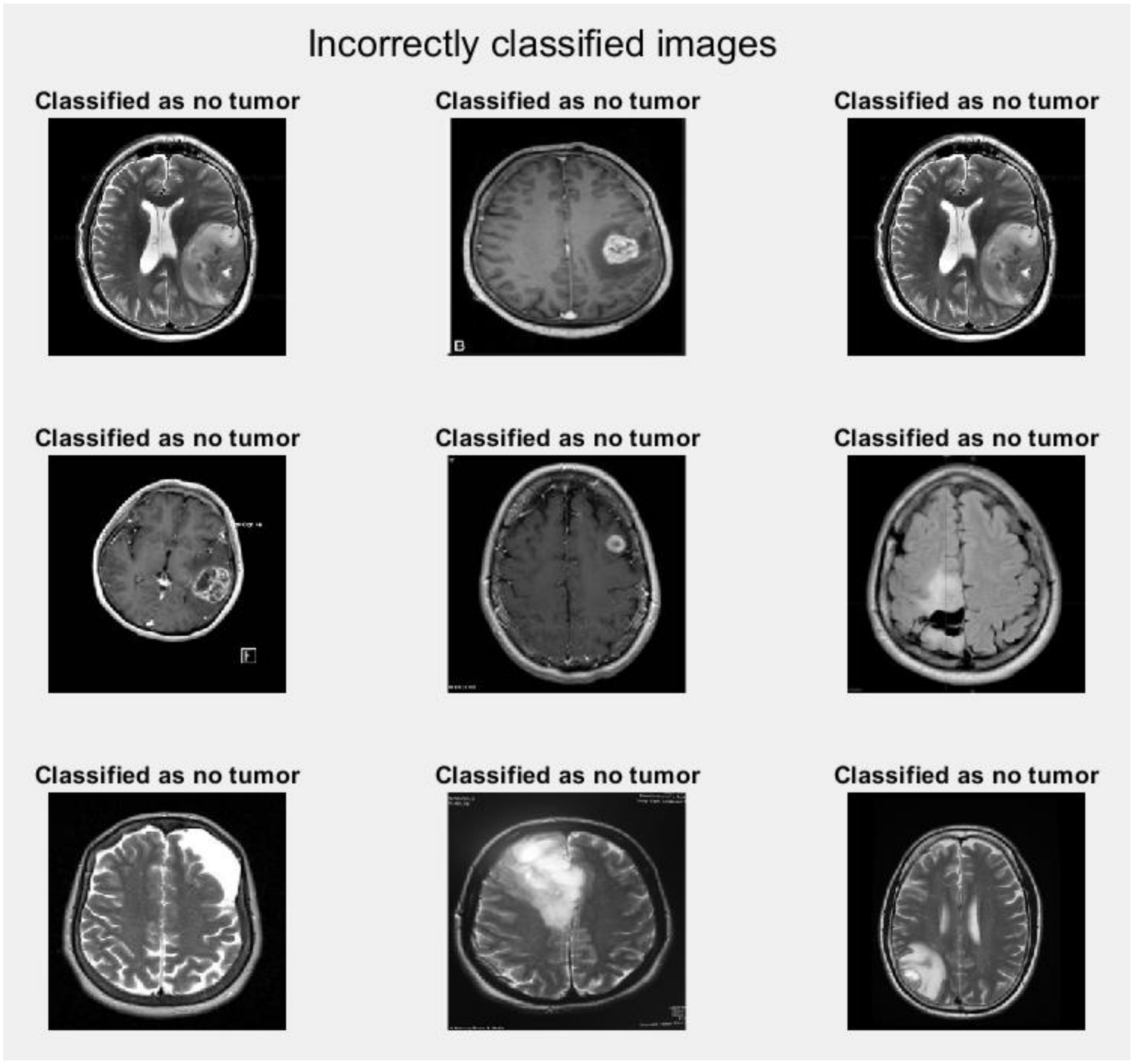

Figure 7: Samples of incorrect classification

V. Summary:

This paper demonstrates that SVD can detect features and LDA can project a random test image onto training dataset. We have over 300 images to train MATLAB what healthy brain and tumorous brain looks like. By applying LDA algorithm, machine can correctly determine if an MRI image belong to which data set. Accuracy yields 80%, and there are a few step we could take to increase the accuracy such as unify the contrast of all images, unify image collecting method (scan, or MRI file). Future exploration could be done with different wavelet transforms, different classification methods, and feature optimization.

Reference

[1] NCI Dictionary of Cancer Terms. National Cancer Institute. www.cancer.gov. Web. Mar 17, 2020

[2] Kutz, Nathan. Data-Driven Modeling & Scientific Computation. Book. Mar 18, 2020.

Appendix A: MATLAB functions

`MODEL=fitcsvm(X,Y) is an alternative syntax that accepts X as an N-by-P matrix of predictors with one row per observation and one column per predictor. Y is the response and is an array of N class labels.`

`[U,S,V] = svd(X,'econ') also produces the "economy size" decomposition. If X is m-by-n with m >= n, then it is equivalent to svd(X,0). For m < n, only the first m columns of V are computed and S is m-by-m.`

`[CA,CH,CV,CD] = dwt2(X,'wname') computes the approximation coefficients matrix CA and details coefficients matrices CH, CV, CD, obtained by a wavelet decomposition of the input matrix X.'wname' is a character vector containing the wavelet name.`

Appendix B: MATLAB Codes

```
%% tc_wavelet function
function tcData = tc_wavelet(tcfile,rows,columns)
[~,n]=size(tcfile); % 4096 x 80

nbcol = size(colormap(gray),1);
for i=1:n
  X=double(reshape(tcfile(:,i),rows, columns));
  [cA,cH,cV,cD]=dwt2(X,'haar');
  cod_cH1 = wcodemat(cH,nbcol);
  cod_cV1 = wcodemat(cV,nbcol);
  cod_edge=cod_cH1+cod_cV1;
  tcData(:,i)=cod_edge(:);
end

%% tc_trainer function
function
[result,w,U,S,V,th,sorttumor,sortcnotumor]=tc_trainer(tumor
,notumor,feature)

nt=length(tumor(1,:));
nnc=length(notumor(1,:));

[U,S,V]=svd([tumor,notumor],0);

images = S*V';
U = U(:,1:feature);
tumors = images(1:feature,1:nt);
notumors = images(1:feature,nt+1:nt+nnc);

md = mean(tumors,2);
mc = mean(notumors,2);

Sw=0;
for i=1:nt
    Sw = Sw + (tumors(:,i)-md)*(tumors(:,i)-md)';
end
for i=1:nnc
    Sw = Sw + (notumors(:,i)-mc)*(notumors(:,i)-mc)';
end

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

[V2,D] = eig(Sb,Sw);  % linear discriminant analysis
dd = diag(D);
```

```
[lambda,ind] = max(abs(dd));
w = V2(:,ind);
w = w/norm(w,2);

vtumor = w'*tumors;
vnotumor = w'*notumors;

result = [vtumor,vnotumor];

if mean(vtumor)>mean(vnotumor)
    w = -w;
    vtumor = -vtumor;
    vnotumor = -vnotumor;
end
% dog < threshold < cat

sorttumor = sort(vtumor);
sortcnotumor = sort(vnotumor);

% figure(4)
% subplot(2,2,1)
% hist(sortdog,30); hold on, plot([18.22 18.h22],[0
10],'r')
% set(gca,'Xlim',[-200 200],'Ylim',[0 10],'Fontsize',[14]),
title('dog')
% subplot(2,2,2)
% hist(sortcat,30,'r'); hold on, plot([18.22 18.22],[0
10],'r')
% set(gca,'Xlim',[-200 200],'Ylim',[0 10],'Fontsize',[14]),
title('cat')

t1 = length(sorttumor);
t2 = 1;
 while sorttumor(t1)>sortcnotumor(t2)
    t1 = t1-1;
    t2 = t2+1;
end
th = (sorttumor(t1)+sortcnotumor(t2))/2;


%% Brain Tumor Classification
% initialize work space
clear all, close all, clc


% First task is to import the image data
```

```
yesfolder = 'TumorDataset/yes';
nofolder = 'TumorDataset/no';

yesfiles = dir(fullfile(yesfolder));
nofiles = dir(fullfile(nofolder));

yesfiles(1:2) = []; % deleting directory elements

% find the smallest image to resize to
% when combining them into one large matrix, they need to
be the same size
[val, idx] = min([yesfiles.bytes]); % get min value and
it's index

% repeat for the no cases
nofiles(1:2) = [];
[val2,idx2] = min([nofiles.bytes]);

% have to check for which is smaller then will resize
images to all that
% size
if val < val2
    smallestimage = fullfile(yesfolder,yesfiles(idx).name);
% find name of smallest image
else
    smallestimage = fullfile(nofolder,nofiles(idx2).name);
end
[rows, columns, colorchannels] =
size(imread(smallestimage)); % get size of smallest image
rows = mean([rows,columns]); % making it square
columns = rows;


% loop through each folder and add images to matrix

% yes dataset
YesData = [];
for i = 1:length(yesfiles)
    basefilename = yesfiles(i).name;
    fullfilename = fullfile(yesfolder,basefilename);
    im = imread(fullfilename);
    if size(im,3) > 1 % checks if image is not gray,
rgb2gray can't work on already gray scale images
        im = rgb2gray(im); % converting to gray scale for
easier analysis
    end
    imr = imresize(im,[rows,columns]);
    YesData(:,i) = double(imr(:));
end

% no dataset
NoData = [];
for j = 1:length(nofiles)
    basefilename = nofiles(j).name;
    fullfilename = fullfile(nofolder,basefilename);
    im = imread(fullfilename);
    if size(im,3) > 1 % checks if image is not gray
        im = rgb2gray(im); % converting to gray scale for
easier analysis
    end
    imr = imresize(im,[rows,columns]);
    NoData(:,j) = double(imr(:));
end

%% Look at FFT of some images

% comparing the yes and the no images along with their
respective FFT's
for i = 1:2
    ys = YesData(:,i);
    ns = NoData(:,i);

    im_y = reshape(ys,rows,columns);
    im_n = reshape(ns,rows,columns);

    L = length(ys)/8400; % i didn't know what to put for
here but I don't think it matters?

    n = length(ys);
    k=(2*pi/L)*[0:n/2 -n/2:-1];
    ks=fftshift(k(1:end-1));

    n2 = length(ns);
    kn = (2*pi/L)*[0:n2/2 -n2/2:-1];
    kns = fftshift(kn);

    ys_f = fft(ys);
    ns_f = fft(ns);

    figure (i)
    sgtitle('MRI images and their respective FFTs')
    subplot(2,2,1)
    imshow(uint8(im_y))
    title('Tumor')

    subplot(2,2,2)
    imshow(uint8(im_n))
    title('No Tumor')

    subplot(2,2,3)
    plot(ks,abs(fftshift(ys_f))/max(ys_f))
    xlim([-10000,10000])
    ylim([0,1])
    title('FFT Tumor')
    ylabel('Amplitude')
    xlabel('Frequency [Hz]')

    subplot(2,2,4)
    plot(kns(1:end-1),abs(fftshift(ns_f))/max(ns_f))
    xlim([-10000,10000])
    ylim([0,1])
    title('FFT No Tumor')
    ylabel('Amplitude')
    xlabel('Frequency [Hz]')
end

%% Split into training test groups
testruns = 20;
percentage = zeros(1,testruns);
errNum = zeros(1,testruns);
for p = 1:testruns

    q1 = randperm(size(YesData,2));
    q2 = randperm(size(NoData,2));

    % split the training/test data by a set amount
    split_yes = floor(0.8*length(q1));
    split_no = floor(0.8*length(q2));

    YesData_train = YesData(:,q1(1:split_yes));
    YesData_test = YesData(:,q1((split_yes + 1):end));

    NoData_train = NoData(:,q2(1:split_no));
    NoData_test = NoData(:,q2((split_no + 1):end));

    numFeat = 10;

    X_test = [YesData_test, NoData_test];
    % This next procedure was used as a comparison to the
professor's method.
    %
    % This first does a FFT on the data then that's fed
through the SVD and
    % classified with MATLAB's classify function.

%      % FFT the images and take SVD
%      X = [YesData_train, NoData_train];
%
%      X_fft = zeros(size(X));
%      for l = 1:size(X,2)
%          X_fft(:,l) = abs(fft(X(:,l)));
%      end
%
%      %X_test_wav = X_test;
%      X_test_wav = zeros(size(X_test));
%      for k = 1:size(X_test,2)
%          X_test_wav(:,k) = abs(fft(X_test(:,k)));
%      end
%
%     [U,S,V] = svd(X_fft,'econ');
%
%     % Run through matlab classify
%
%      U = U(:,1:numFeat);
%      xtrain = V(:,1:numFeat);
%      xtest = U'*X_test_wav;
%      xtest = xtest(:,1:numFeat);
%
%      ctrain =
[repmat({'Tumor'},[size(YesData_train,2),1]);repmat({'NoTum
or'},[size(NoData_train,2),1])];
%      truth =
[repmat({'Tumor'},[size(YesData_test,2),1]);repmat({'NoTumo
r'},[size(NoData_test,2),1])];
%
%      %svm.mod = fitcecoc(xtrain,ctrain);
%      %pre = predict(svm.mod,xtest);
%
%      pre = classify(xtest,xtrain,ctrain);
%
%      num_correct = 0;
%      for k = 1:length(pre)
%         if strcmp(pre{k},truth{k})
%              num_correct = num_correct + 1;
%         end
%      end
%      percentage(p) = (num_correct/length(truth))*100;

% attempt at using the LDA that the professor used for the
dog and cat data

    Yes_wave = tc_wavelet(YesData_train,rows,columns);
    No_wave = tc_wavelet(NoData_train,rows,columns);
    [result,w,U,S,V,threshold,sorttumor,sortnotumor] =
tc_trainer(Yes_wave,No_wave,numFeat);
    Test_Wave = tc_wavelet(X_test, rows, columns);
    Test_Mat = U'*Test_Wave;
    pval = w'*Test_Mat;

    hiddenlabels = [zeros(1,size(YesData_test,2)),
ones(1,size(NoData_test,2))];

    [mt,TestNum] = size(X_test);

    ResVec = (pval>threshold);
    %disp('Number of mistakes');
    errNum(p) = sum(abs(ResVec - hiddenlabels));
    %disp('Rate of success');
    percentage(p) = (1-errNum(p)/TestNum)*100;


end
disp('Average percentage correct')
mean(percentage)

%% Graph some stuff


k = 1;
TestNum = length(pval);
figure()
sgtitle('Incorrectly classified images')
for i = 1:TestNum
  if k > 9
      break
  end
  if ResVec(i)~=hiddenlabels(i)
      imm = reshape(X_test(:,i),rows,columns);
      subplot(3,3,k)
      imshow(uint8(imm))
      if ResVec(i) == 0
          title('Classified as tumor')
      else
          title('Classified as no tumor')
      end
      k = k+1;
  end
end


figure()
sig = sort(diag(S),'Descend');
sgtitle('Singular values of each set')

subplot(2,1,1), plot(sig(1:50),'ko','Linewidth',[1.5])
ylabel('Singular Values')
xlabel('Singular Value Along Diagonal')

subplot(2,1,2), semilogy(sig(1:50),'ko','Linewidth',[1.5])
ylabel('Log of Singular Values')
xlabel('Singular Value Along Diagonal')

figure ()
sgtitle('First four principal components of the training
data')
for j=1:4
  subplot(2,2,j)
  ut1=reshape(U(:,j),sqrt(size(U,1)),sqrt(size(U,1)));
  ut2=ut1(size(ut1,1):-1:1,:);
  pcolor(ut1), %colormap(hot)
  title(['mode' num2str(j)])
  set(gca,'Xtick',[],'Ytick',[])
end

figure()
sgtitle('First three features of each set in training
data')
for j=1:3
  subplot(3,2,2*j-1)
  plot(1:60,V(1:60,j),'ko-')
  ylabel(['Mode ' num2str(j)])
  if j == 3
      xlabel('num of images')
  end
  ylim([-0.2 0.2])
  subplot(3,2,2*j)

plot(size(YesData,2)+1:size(V,1),V(size(YesData,2)+1:end,j)
,'ko-')
  if j == 3
      xlabel('Num of images')
  end
  ylim([-0.2 0.2])
end
subplot(3,2,1), title('Tumor')
subplot(3,2,2), title('No Tumor')
```