# %% clear

clc

close all

clear all

# %% generate matrix to store all data about images

data=[];

% neural network data

nn\_features=[];

nn\_classes=[];

nn\_reduced\_features=[];

% SVM data

svm\_features=[];

svm\_classes=[];

svm\_reduced\_features=[];

# %% acquisition and loading images

% load image matrix

load training\_database.mat;

% get the length of image matrix(number of images)

number\_of\_photos=length(photos);

# %% prosess all image in loop

for photo\_num=1:number\_of\_photos

%%

im=photos(photo\_num).photo;

%figure, imshow(im);

%store to data matrix

data(photo\_num).photos.im\_original=im;

# %% image resize to 512\*512

im=imresize(im,[512, 512]);

%figure, imshow(im);

im\_original=im; % to be used in crop after detectiong thr lesion

# %% convert image from RGB to graylevel

im\_gray=rgb2gray(im);

%figure, imshow(im\_gray);

% from ISSN: 2249-7951

# %% contrast and brightness adjustment, gamma correction

% H = vision.ContrastAdjuster;

% im\_gray=step(H,im\_gray);

% %figure, imshow(im\_gray);

% from 1877-0509

im\_gray=imadjust(im\_gray);

%figure, imshow(im\_gray);

# %% median filter

im\_gray=medfilt2(im\_gray);

%figure, imshow(im\_gray);

% store to data matrix

data(photo\_num).photos.im\_gray=im\_gray;

% from ISSN(p): 2249-684x

# %% Segmentation to detect a lesion region

% maximun entropy

% [level im\_bw]=max\_entropy(im\_gray);

% %figure, imshow(im\_bw);

% outso method

im\_bw=im2bw(im\_gray);

%figure, imshow(im\_bw);

# %% complement of the image

im\_bw=imcomplement(im\_bw);

%figure, imshow(im\_bw);

# %% filled the holes in the objects

im\_filled=imfill(im\_bw,'holes');

%figure, imshow(im\_filled);

# %% opening

im\_opened = imopen(im\_filled, strel('disk', 10));

%figure, imshow(im\_opened);

%store to data matrix

data(photo\_num).photos.im\_bw=im\_opened;

# %% labeled image to get informations about objects

[im\_labeled,num\_of\_objects]=bwlabel(im\_opened,8);

%store to data matrix

data(photo\_num).photos.im\_labeled=im\_labeled;

# %% extract properties of objects in labeled image

im\_info=regionprops(im\_labeled,'all');

%store to data matrix

data(photo\_num).im\_info=im\_info;

# %% using centroid to detect the central object(lesion)

% size of image

[n,m]=size(im\_opened);

% contre of the image and round to nearest integer

centre\_bw=[round(m/2),round(n/2)];

if (im\_labeled(round(m/2),round(n/2))>0)

central\_object\_index=im\_labeled(round(m/2),round(n/2));

else

% minimum distance to the centre of the image

min\_distance=inf;

% for loop to get the minimum didtance of objects

for obj=1:num\_of\_objects

distance=sqrt((im\_info(obj).Centroid(1)-centre\_bw(1))^2+(im\_info(obj).Centroid(2)-centre\_bw(2))^2);

if distance<min\_distance

min\_distance=distance;

central\_object\_index=obj;

end

end

end

# %% return the central object (lesion)

central\_bounding\_box=im\_info(central\_object\_index).BoundingBox;

% crop the central object

im\_bw\_croped=imcrop(im\_labeled,central\_bounding\_box);

%figure, imshow(im\_bw\_croped);

% store to data

data(photo\_num).photos.im\_bw\_croped=im\_bw\_croped;

data(photo\_num).central\_bounding\_box=central\_bounding\_box;

# %% calculate labeled image for lesion

[im\_bw\_croped\_labeled, num\_of\_objects]=bwlabel(im\_bw\_croped,8);

lesion\_info=regionprops(im\_bw\_croped\_labeled,'all');

% store to data matrix

data(photo\_num).lesion\_info=lesion\_info;

# %% determine the largest area b/t objects (lesion)

% calculate all areas of objects in one matrix after filling the holes;

objects\_areas=cat(1,lesion\_info.Area);

%store to data matrix

data(photo\_num).objects\_areas=objects\_areas;

# %% determine the largest object in the image

max\_area=0;

large\_object\_index=0;

for i=1:length(objects\_areas)

if(objects\_areas(i)>max\_area)

max\_area=objects\_areas(i);

large\_object\_index=i;

end

end

# %% delete the objects that has areas smaller than the largest one

im\_bw\_croped=bwareaopen(im\_bw\_croped,max\_area);

%figure, imshow(im\_bw\_croped);

%store to data matrix

data(photo\_num).photos.im\_bw\_croped=im\_bw\_croped;

# %% crop lesion area from original image

% crop the lesion area from the original image after detection the lesion

% by using binary image and objects region properties;

im\_original\_croped=imcrop(im\_original,central\_bounding\_box);

%figure, imshow(im\_original\_croped);

%store to data matrix

data(photo\_num).photos.im\_original\_croped=im\_original\_croped;

# %% remove any things except the tumor object

% [rows,columns]=size(im\_bw\_croped);

% for x=1:rows

% for y=1:columns

% if(im\_bw\_croped(x,y)==0)

% im\_original\_croped(x,y,1)=255;

% im\_original\_croped(x,y,2)=255;

% im\_original\_croped(x,y,3)=255;

% end

% % end

% % end

% %figure, imshow(im\_original\_croped);

% %store to data matrix

% data(photo\_num).photos.im\_original\_croped=im\_original\_croped;

r=im\_original\_croped(:,:,1);

g=im\_original\_croped(:,:,2);

b=im\_original\_croped(:,:,3);

rs=(r).\*uint8(im\_bw\_croped);

gs=(g).\*uint8(im\_bw\_croped);

bs=(b).\*uint8(im\_bw\_croped);

im\_original\_croped=uint8(zeros(size(im\_original\_croped)));

im\_original\_croped(:,:,1)=rs;

im\_original\_croped(:,:,2)=gs;

im\_original\_croped(:,:,3)=bs;

%figure, imshow(im\_original\_croped)

# %% return croped image to gray levels

im\_gray\_croped=rgb2gray(im\_original\_croped);

%figure, imshow(im\_gray\_croped);

%store to data matrix

data(photo\_num).photos.im\_gray\_croped=im\_gray\_croped;

# %% ABCD features

%% A: asymmetry index (AI)

lesion\_area=bwarea(im\_bw\_croped);

[height, width]=size(im\_bw\_croped);

image\_area=height\*width;

dA=image\_area-lesion\_area;

%area indes

AI=(dA/image\_area)\*100;

% store to data

data(photo\_num).features.A=AI;

# %% B: border irregularity (campact index CI)

%perimeter oa the lesion(number of edge pixles)

im\_bw\_bounadaries = bwperim(im\_bw\_croped);

%figure, imshow(im\_bw\_bounadaries);

% perimeter=bwarea(im\_bw\_bounadaries);

perimeter=0;

[x,y]=size(im\_bw\_bounadaries);

for i=1:x

for j=1:y

if(im\_bw\_bounadaries(i,j)==1)

perimeter=perimeter+1;

end

end

end

%compaxt index (CI)

CI=(power(perimeter,2))/(4\*pi\*lesion\_area);

% store to data

data(photo\_num).B=CI;

%% C: color variation (CV)

% colors values

white=[255, 255, 255];

black=[0, 0, 0];

red=[255, 0, 0];

light\_broun=[205, 133, 63];

dark\_broun=[101, 67, 33];

blue\_gray=[0, 134, 139];

%convert RGB image to double

im\_double=im2double(im\_original\_croped);

% calculate the distance from any pixle to each color-value

% get im\_bw\_croped size to check the object pixle just

[r,c]=size(im\_bw\_croped);

color\_count=zeros(1,6);

for x=1:r

for y=1:c

if(im\_bw\_croped(x,y)==1)

%distance

%white

dist1=sqrt((im\_double(x,y,1)-white(1))^2+(im\_double(x,y,2)-white(2))^2+(im\_double(x,y,3)-white(3))^2);

if(dist1==0)

color\_count(1,1)=1;

end

%black

dist2=sqrt((im\_double(x,y,1)-black(1))^2+(im\_double(x,y,2)-black(2))^2+(im\_double(x,y,3)-black(3))^2);

if(dist2==0)

color\_count(1,2)=1;

end

%red

dist3=sqrt((im\_double(x,y,1)-red(1))^2+(im\_double(x,y,2)-red(2))^2+(im\_double(x,y,3)-red(3))^2);

if(dist3==0)

color\_count(1,3)=1;

end

%light broun

dist4=sqrt((im\_double(x,y,1)-light\_broun(1))^2+(im\_double(x,y,2)-light\_broun(2))^2+(im\_double(x,y,3)-light\_broun(3))^2);

if(dist4==0)

color\_count(1,4)=1;

end

%dark broun

dist5=sqrt((im\_double(x,y,1)-dark\_broun(1))^2+(im\_double(x,y,2)-dark\_broun(2))^2+(im\_double(x,y,3)-dark\_broun(3))^2);

if(dist5==0)

color\_count(1,5)=1;

end

%blue gray

dist6=sqrt((im\_double(x,y,1)-blue\_gray(1))^2+(im\_double(x,y,2)-blue\_gray(2))^2+(im\_double(x,y,3)-blue\_gray(3))^2);

if(dist6==0)

color\_count(1,6)=1;

end

end

end

end

color\_feature=sum(color\_count);

% store to data

data(photo\_num).C=color\_feature;

%% D: diameter (D) (equivalent diameter)

diameter\_pixles=[lesion\_info(1).EquivDiameter];

diameter=diameter\_pixles/3.779527559055;

% store to data

data(photo\_num).D=diameter;

# %% TDS features

TDS=(AI\*1.3)+(CI\*0.1)+(color\_feature\*0.5)+(diameter\*0.5);

% store to data

data(photo\_num).TDS=TDS;

# %% boundaries and circulation

boundaries= bwboundaries(im\_bw\_croped,'noholes');

%Display the label matrix and draw each boundary

%figure, imshow(im\_bw\_croped);

hold on

for k = 1:length(boundaries)

boundary = boundaries{k};

plot(boundary(:,2), boundary(:,1), 'r', 'LineWidth', 2)

end

% stats = regionprops(L,'Area','Centroid');

% threshold = 0.94;

%

% % loop over the boundaries

for k = 1:length(boundaries)

% obtain (X,Y) boundary coordinates corresponding to label 'k'

boundary = boundaries{k};

% compute a simple estimate of the object's perimeter

delta\_sq = diff(boundary).^2;

perimeter = sum(sqrt(sum(delta\_sq,2)));

% obtain the area calculation corresponding to label 'k'

area = lesion\_info(k).Area;

% compute the roundness metric

metric = 4\*pi\*area/perimeter^2;

% display the results

circulation\_string = sprintf('%2.2f',metric);

% mark objects above the threshold with a black circle

% if metric > threshold

% centroid = stats(k).Centroid;

% plot(centroid(1),centroid(2),'ko');

% end

text(boundary(1,2)-35,boundary(1,1)+13,circulation\_string,'Color','y',...

'FontSize',14,'FontWeight','bold');

end

circulation=str2double(circulation\_string);

% store to data

data(photo\_num).features.circulation=circulation;

%%

im\_histogram = histeq(im\_gray\_croped);

%figure, imagesc(im\_histogram), colormap gray, title('histogram eq for the image')

% mean

M = mean2(im\_histogram(:));

% standard deviation

SD =std(double(im\_histogram(:)));

% skewness

im\_double= im2double(im\_histogram);

S=skewness(im\_double(:));

% kurtosis

K=kurtosis(double(im\_histogram(:)));

% energy

im\_histogram\_info=graycoprops(im\_histogram,{'energy'});

Enr=im\_histogram\_info(1).Energy;

% entropy

Ent=entropy(im\_histogram(:));

%% features using GLCM

%% creates a gray-level co-occurrence matrix (GLCM)

GLCM = graycomatrix(im\_gray\_croped);

%% statistical properties of the GLCM.

% GLCM main features

GLCM\_features = graycoprops(GLCM,'all');

% GLCM 22-features

% glcm\_features=GLCM\_features(GLCM);

%store to data matrix

data(photo\_num).info.glcm=GLCM;

% data(photo\_num).features=[stats];

data(photo\_num).glcm\_features=GLCM\_features;

% //note

% GLCM in binary image, give us the same result as in graySale

% from ISSN: 2249-7951

%% set all features in one matrix

% features

nn\_features(1,photo\_num)=GLCM\_features.Contrast;

nn\_features(2,photo\_num)=S; % skewness

nn\_features(3,photo\_num)=K; % kurtosis

nn\_features(4,photo\_num)=Ent; %entropy

nn\_features(5,photo\_num)=M; % mean

nn\_features(6,photo\_num)=SD;% dtandard deviation

nn\_features(7,photo\_num)=circulation;

nn\_features(8,photo\_num)=Enr; % energy

nn\_features(9,photo\_num)=GLCM\_features.Correlation;

nn\_features(10,photo\_num)=GLCM\_features.Homogeneity;

nn\_features(11,photo\_num)=TDS;

% store to data

data(photo\_num).features=nn\_features(:,photo\_num);

%% SVM data

% cconvert to SVM features

svm\_features=[];

[r,c]=size(nn\_features);

for i=1:r

svm\_features(:,i)=nn\_features(i,:);

end

%convert classes to svm\_classes

svm\_classes=[];

[r,c]=size(nn\_classes);

for i=1:c

svm\_classes(i,1)=nn\_classes(1,i);

end

% store to data

data(photo\_num).svm\_features=svm\_features(photo\_num,:);

%% set the class to the image

% in neural network

% nn\_classes=[1,0]: abnormal

% nn\_classes=[0,1]: normal

% in support vector machine

% svm\_classes=1 : abnormal

% svm\_classes=0 : normal

nn\_classes(:,photo\_num)=photos(photo\_num).class;

% store to data

data(photo\_num).class=nn\_classes(:,photo\_num);

close all

end % end of for loop

%% convert features to column-features

[r,c]=size(nn\_features);

features\_to\_pca=[];

for i=1:r

features\_to\_pca(:,i)=nn\_features(i,:);

end

%% PCA

reduced\_features\_index=PCA(features\_to\_pca)

%% select features after PCA-code

nn\_reduced\_features=[];

svm\_reduced\_features=[];

i=length(reduced\_features\_index);

while i>=(length(reduced\_features\_index)-4)

nn\_reduced\_features=[nn\_reduced\_features; nn\_features(reduced\_features\_index(i),:)];

svm\_reduced\_features=[svm\_reduced\_features, svm\_features(:,reduced\_features\_index(i))];

i=i-1;

end

%% saving all data

save('data.mat','data');

% neural network data

save('nn\_features.mat','nn\_features');

save('nn\_classes.mat','nn\_classes');

save('nn\_reduced\_features.mat','nn\_reduced\_features');

% support vector machine data

save('svm\_classes.mat','svm\_classes');

save('svm\_features.mat','svm\_features');

save('svm\_reduced\_features.mat','svm\_reduced\_features');

% PCA data

save('features\_to\_pca.mat','features\_to\_pca');

save('reduced\_features\_index.mat','reduced\_features\_index');