SCHOOL OF COMPUTING

FACULTY OF ENGINEERING

UNIVERSITY TECHNOLOGY MALAYSIA

FUNDAMENTAL OF IMAGE PROCESSING

(SCSV3213)

SEMESTER 1 2019/2020

PROJECT REPORT

TITLE: HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

By

LIM BAO JING	(A17CS0076) (970106-04-5002)
LOW CHIA JING	(A17CS0083) (970713-02-5767)
TAN SEE JOU	(A17CS0218) (970205-26-5254)

SECTION 01

LECTURER:

DR MD. SAH BIN HJ. SALAM

Contents

BACKGROUND

A neural network is a series of algorithms that try to stimulate the learning function of the human brain. It recognizes the underlying relationships in a set of data through a series of process that is similar to the way the human brain operates. The neural network typically composed of interconnected units or artificial neurons which are collected and classified information according to a specific architecture.

The objective of the neural network is to transform the inputs into meaningful outputs. A neural network contains several layers of interconnected nodes. The input layer collects the input patterns while the output layer has classifications or output signals to which the patterns may map. The hidden layer between the input and output layers is where the calculations take place.

1. INTRODUCTION

The main objective of this project is applying Neural Network in the recognition of handwritten Arabic numbers, which is from 0 to 9. The human brain can recognize and differentiate between the number effortlessly through experiences and memories. However, for a computer, it is not that easy to differentiate especially handwritten.

A large number of handwritten Arabic number data set had been used for training purposes and then the data will be processed into several phases such as image enhancement, preprocessing, feature extraction, recognition and so on. Along with the processes, we have developed a system that can learn and automatically infer rules for differentiating and recognizing the handwritten numbers instead of memories from those training.

2. OBJECTIVES

- ➢ To develop a system which capable to recognize different handwritten Arabic numbers.
- ➢ To understand the concept of neural networks and apply the neural network in the system.

3. PROCESS

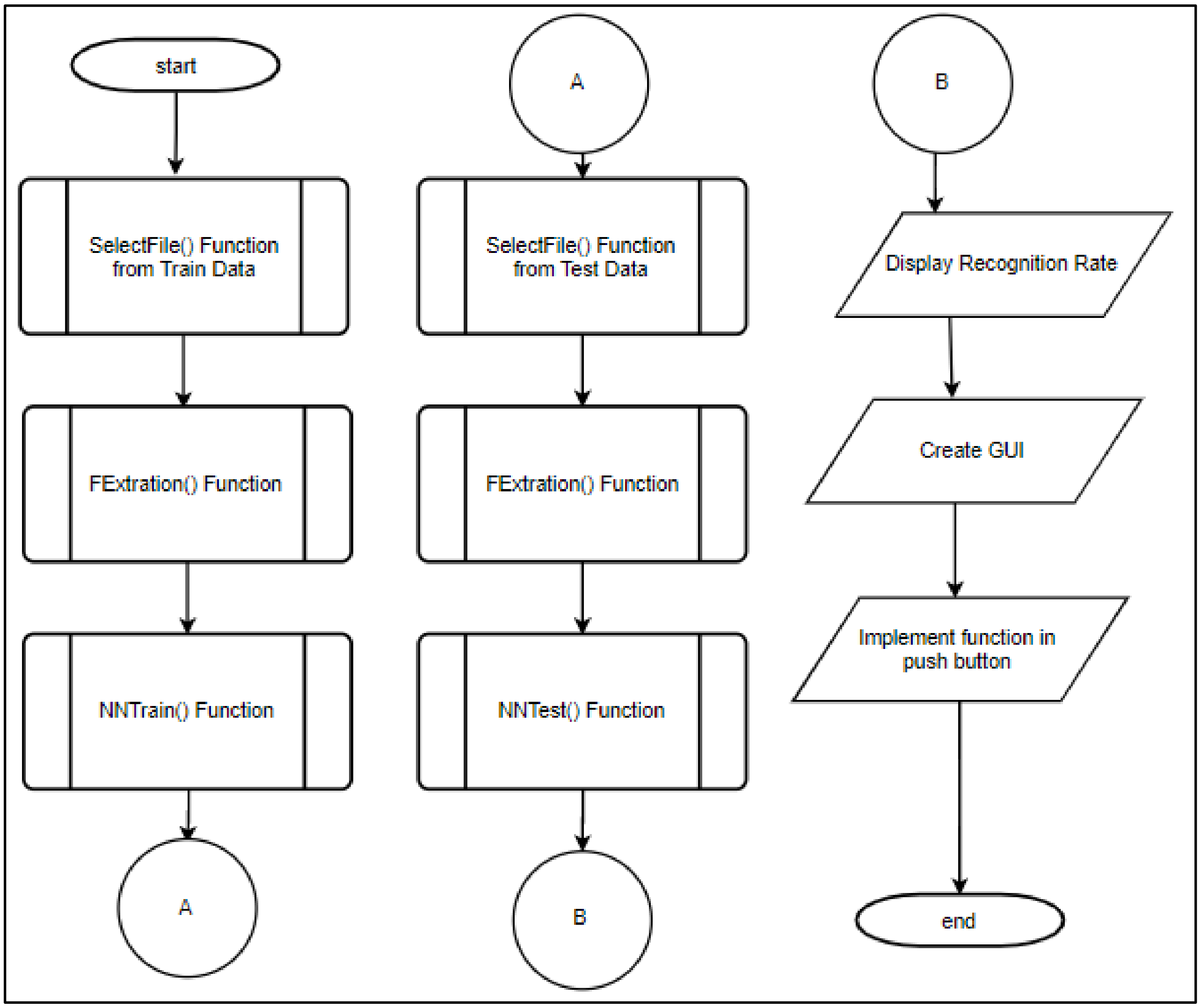


*Figure 1: **Workflow** to solve this project*

3.1 The Pattern Database

Topic: Handwritten Arabic Number

Number of Class: 10 (0 to 9)

Training patterns: 10,000 (each patterns 1000)

Testing patterns: 5,000 (each patterns 500)

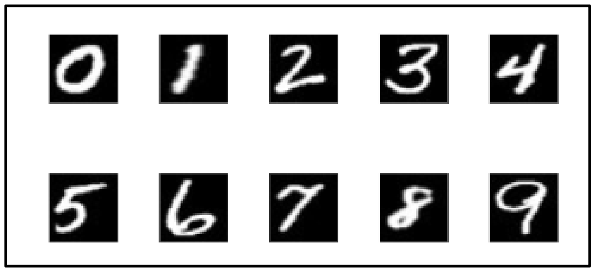
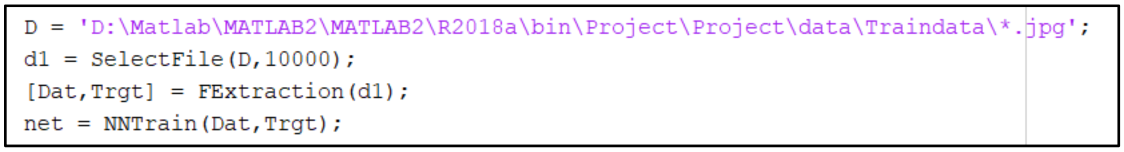


Figure 2: 10 different patterns which are from 0 to 9.

3.2 Pre-Processing Steps

3.2.1 Select and Import the Patterns

SampleCallDigit Program



```
D = 'D:\Matlab\MATLAB2\MATLAB2\R2018a\bin\Project\Project\data\Traindata\*.jpg';
d1 = SelectFile(D,10000);
[Dat,Trgt] = FExtraction(d1);
net = NNTrain(Dat,Trgt);
```

*Figure 3: **SampleCallDigit()** function*

This function is used to train and test the dataset.

1. SelectFile(D,N) will select the files to be trained.

2. FExtraction(d) will do the pre-processing and features extraction of the selected patterns with the target. The output will be matric for training and target output.

SelectFile() Function

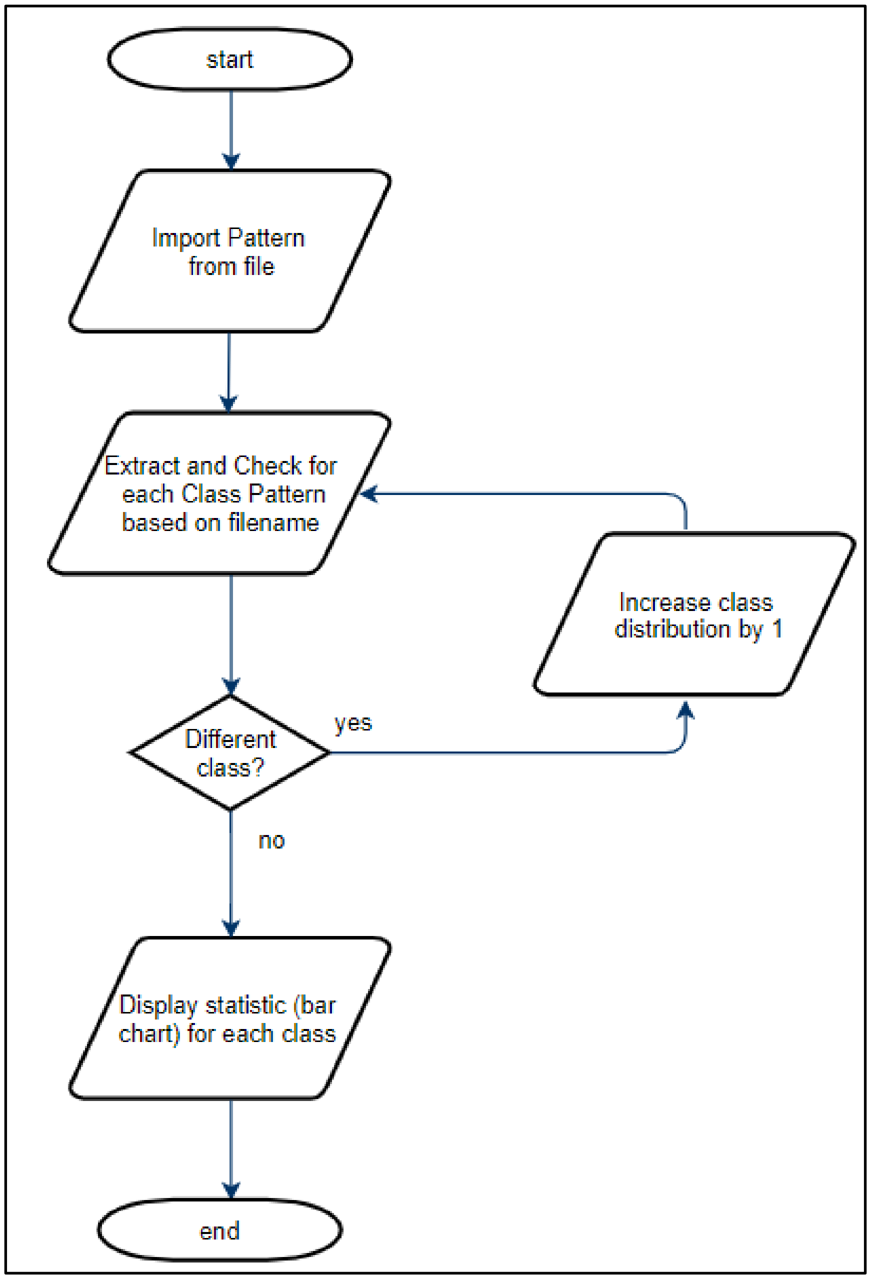
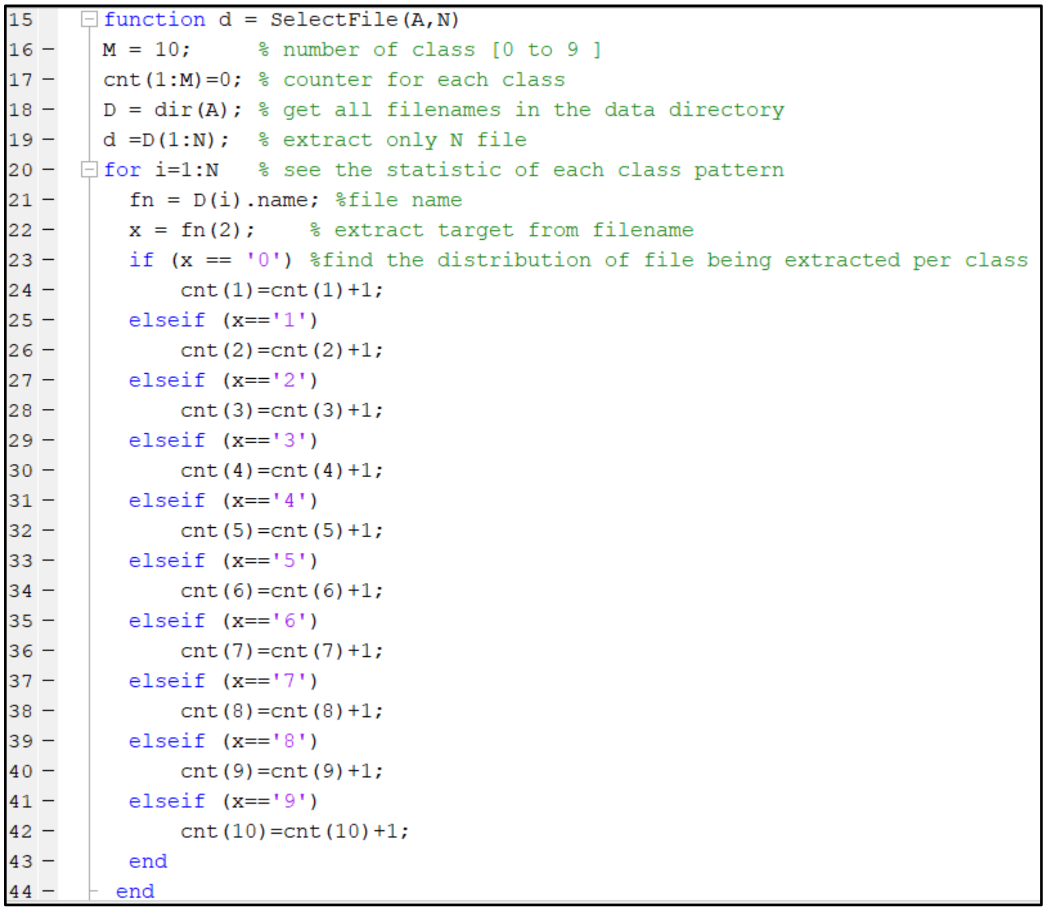

start
Import Pattern from file
Extract and Check for each Class Pattern based on filename
Increase class distribution by 1
Different class?
yes
no
Display statistic (bar chart) for each class
end

*Figure 4: Workflow for **SelectFile()** function.*

This function is used to select and import the dataset from a different path. This function will also show the statistic of the patterns imported. We have chosen the same number of each pattern for training (1000 each pattern) and testing (500 each pattern) to ease the training and testing process.



```matlab
function d = SelectFile(A,N)
M = 10;      % number of class [0 to 9 ]
cnt(1:M)=0; % counter for each class
D = dir(A); % get all filenames in the data directory
d =D(1:N);  % extract only N file
for i=1:N   % see the statistic of each class pattern
  fn = D(i).name; %file name
  x = fn(2);     % extract target from filename
  if (x == '0') %find the distribution of file being extracted per class
      cnt(1)=cnt(1)+1;
  elseif (x=='1')
      cnt(2)=cnt(2)+1;
  elseif (x=='2')
      cnt(3)=cnt(3)+1;
  elseif (x=='3')
      cnt(4)=cnt(4)+1;
  elseif (x=='4')
      cnt(5)=cnt(5)+1;
  elseif (x=='5')
      cnt(6)=cnt(6)+1;
  elseif (x=='6')
      cnt(7)=cnt(7)+1;
  elseif (x=='7')
      cnt(8)=cnt(8)+1;
  elseif (x=='8')
      cnt(9)=cnt(9)+1;
  elseif (x=='9')
      cnt(10)=cnt(10)+1;
  end
 end
```

*Figure 5: **SelectFile()** function.*

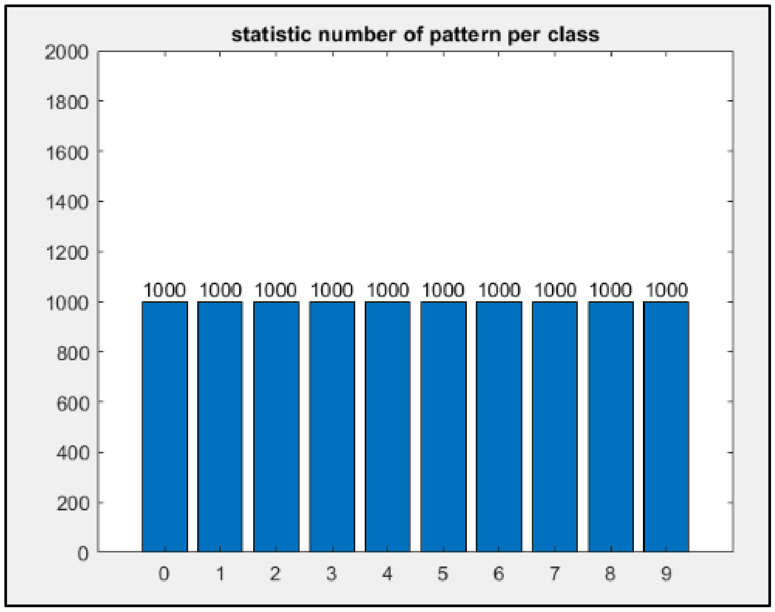


Figure 6:Statistic pattern per class.

3.2.2 Image enhancement

***FExtration()* Function**

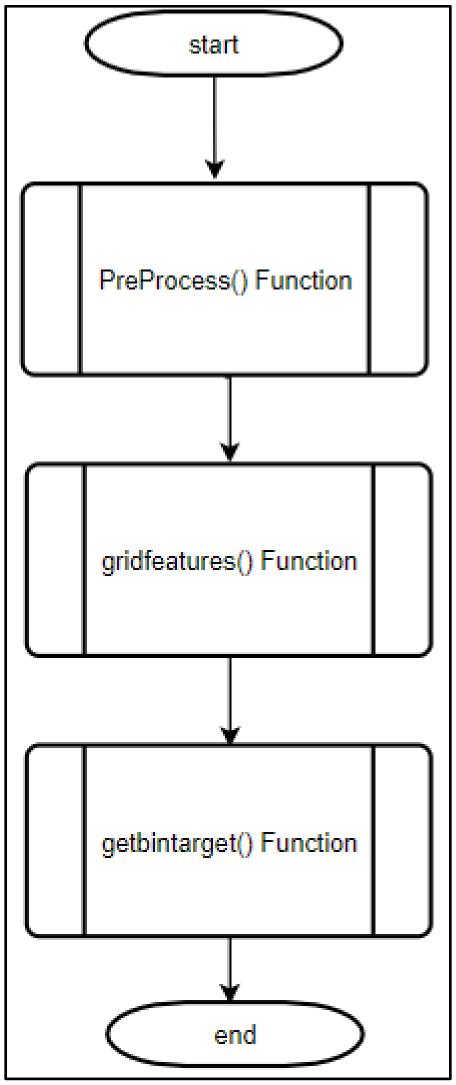
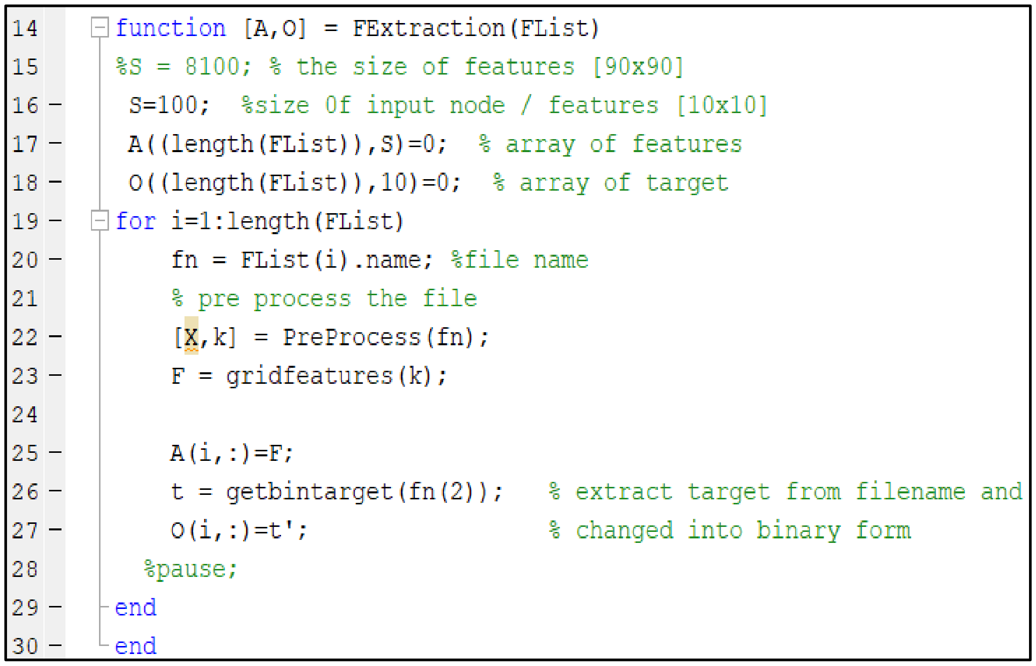


*Figure 7: Workflow for **FExtration()** function.*

In ***FExtraction()*** function, we applied three functions which are ***Preprocess()*** function, ***gridfeatures()*** function and ***getbbintarget()*** function.



```
function [A,O] = FExtraction(FList)
%S = 8100; % the size of features [90x90]
 S=100;  %size Of input node / features [10x10]
 A((length(FList)),S)=0;  % array of features
 O((length(FList)),10)=0;  % array of target
for i=1:length(FList)
    fn = FList(i).name; %file name
    % pre process the file
    [X,k] = PreProcess(fn);
    F = gridfeatures(k);

    A(i,:)=F;
    t = getbintarget(fn(2));   % extract target from filename and
    O(i,:)=t';                 % changed into binary form
  %pause;
end
end
```

*Figure 8:**Feature Extraction()** Function.*

Preprocess() Function

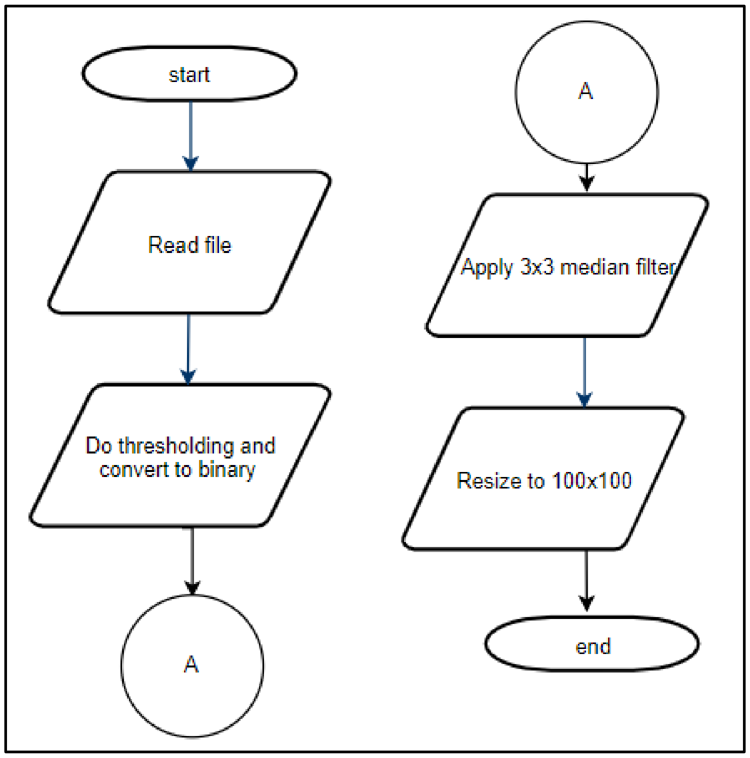
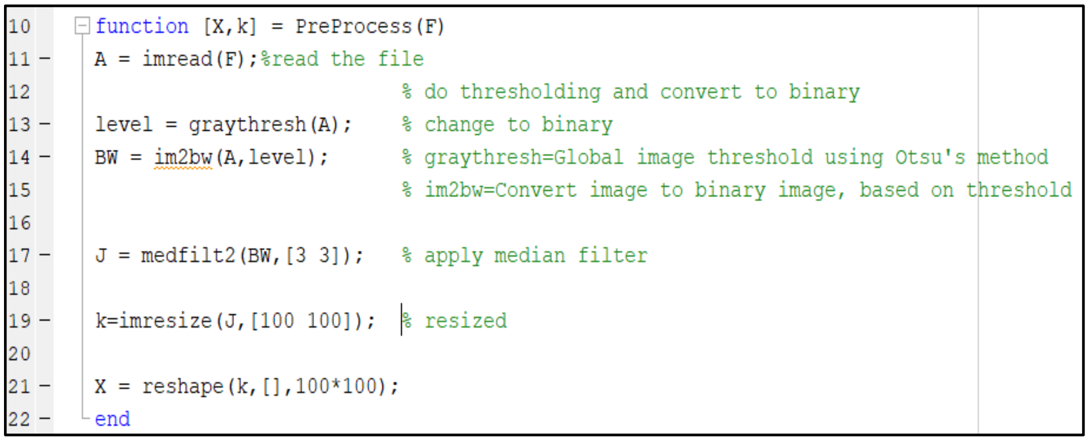


*Figure 9:Workflow for **Preprocess() function**.*



```
function [X,k] = PreProcess(F)
A = imread(F);%read the file
                         % do thresholding and convert to binary
level = graythresh(A);   % change to binary
BW = im2bw(A,level);     % graythresh=Global image threshold using Otsu's method
                         % im2bw=Convert image to binary image, based on threshold

J = medfilt2(BW,[3 3]);  % apply median filter

k=imresize(J,[100 100]); % resized

X = reshape(k,[],100*100);
end
```

*Figure 10:**Preprocess() function**.*

*In **Preprocess()** function*, ***Greythresh()*** had been applied to compute the global threshold so that it can be used to convert the data file to binary image with ***im2bw()***; After that, ***medfilt2()*** median filter with 3*3 neighborhood size had been applied. Then, we used ***imresize()*** to resize to data into 100*100. The reason why we chose this sequence of preprocessing steps will be discussed in the CRITERIA part.

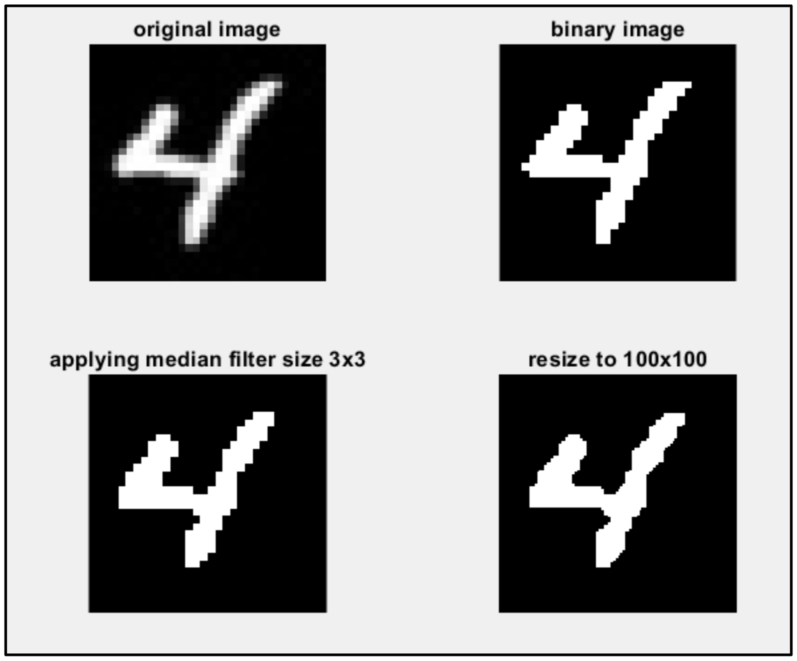
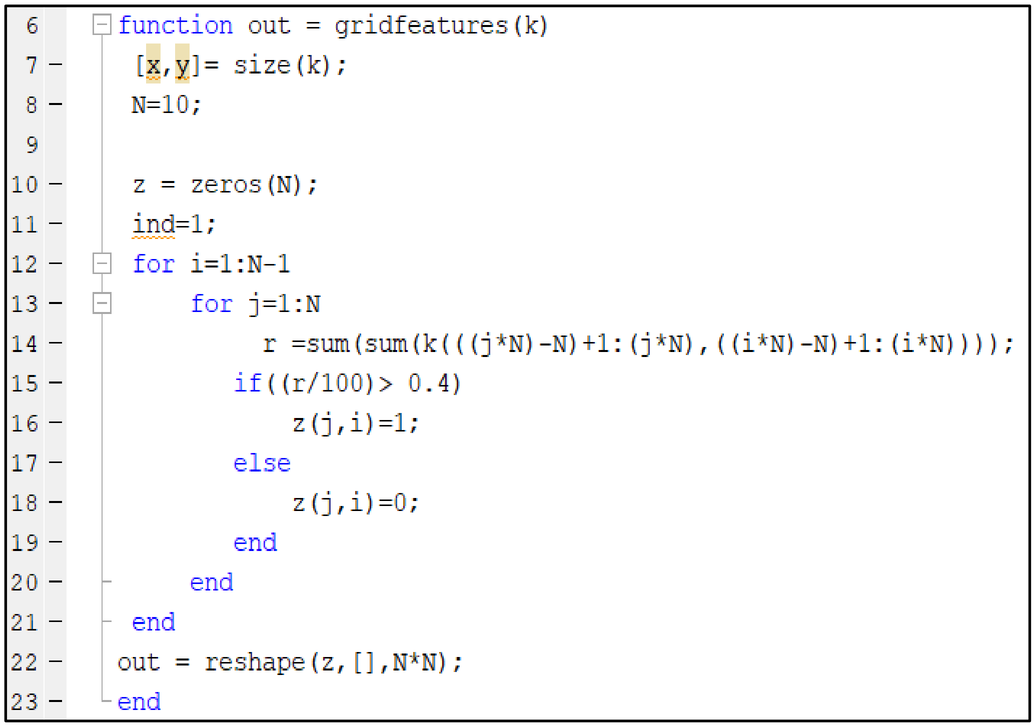


Figure 11: Preprocessing output

gridfeature() Function

After the preprocessing part, we have to extract the features. The processed data files undergo ***gridfeature()*** function and the image is divided into 10*10 grid.



```
function out = gridfeatures(k)
 [x,y]= size(k);
 N=10;

 z = zeros(N);
 ind=1;
 for i=1:N-1
     for j=1:N
          r =sum(sum(k(((j*N)-N)+1:(j*N),((i*N)-N)+1:(i*N))));
        if((r/100)> 0.4)
            z(j,i)=1;
        else
            z(j,i)=0;
        end
     end
 end
out = reshape(z,[],N*N);
end
```

*Figure 12:**gridfeatures()** Function.*

Since our dataset has a thick outline, we choose 0.4 as the factor to ease the detection of the Arabic number. It will produce a better recognition rate as compared to other numbers (will be discussed in CRITERIA part).

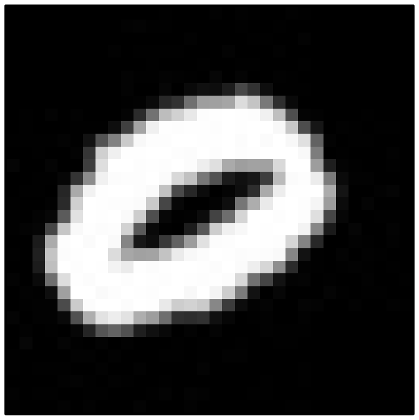


Figure 13:Sample data - a very thick outline 0.

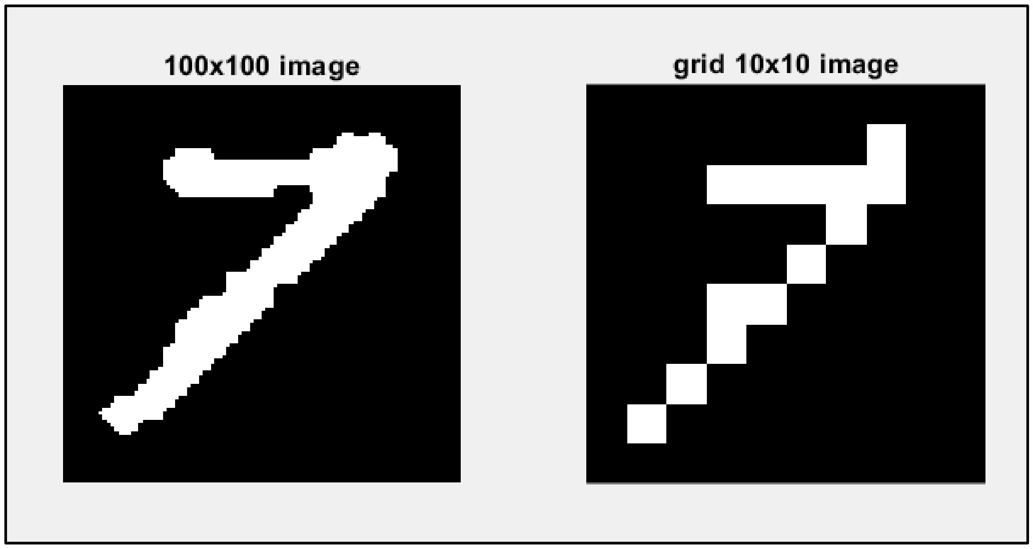
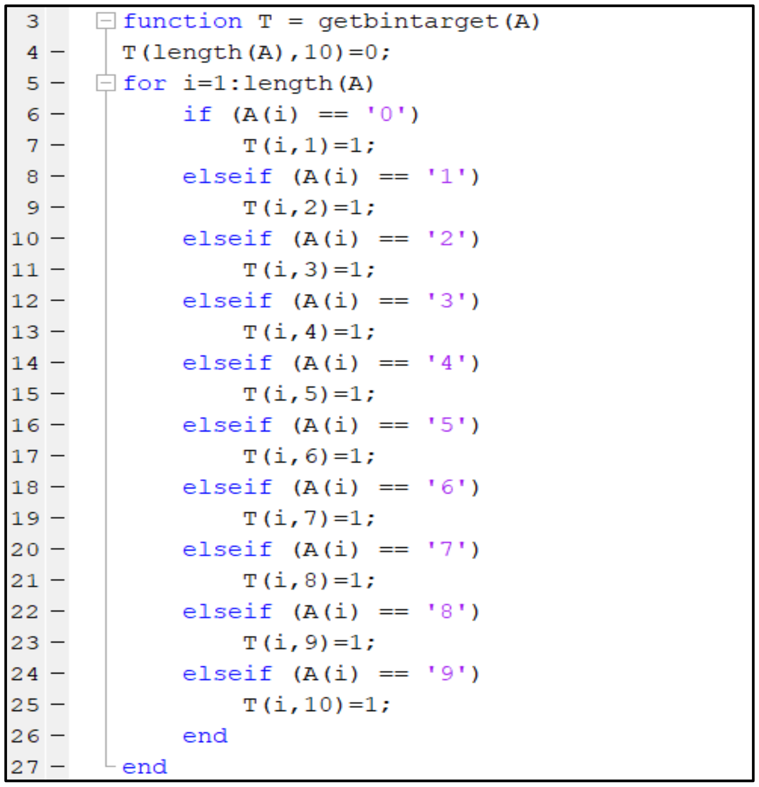


Figure 14: The sample output after undergoing gridfeatures() function.

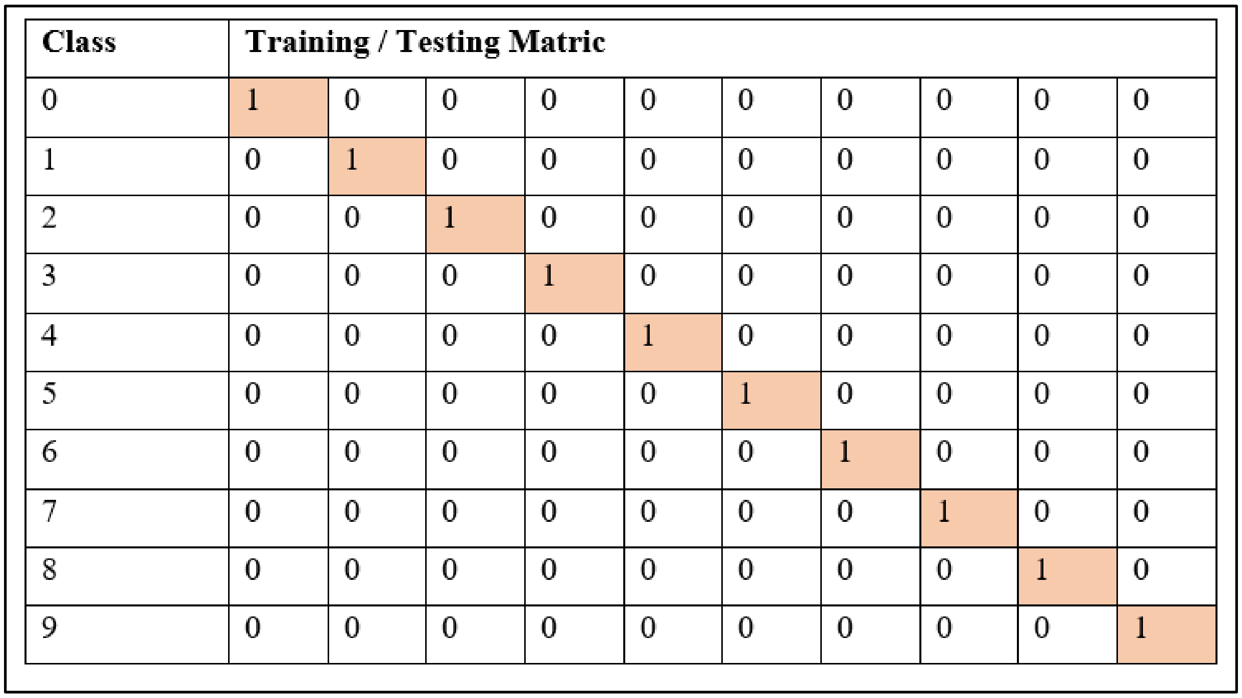
getbintarget() Function



```
function T = getbintarget(A)
T(length(A),10)=0;
for i=1:length(A)
    if (A(i) == '0')
        T(i,1)=1;
    elseif (A(i) == '1')
        T(i,2)=1;
    elseif (A(i) == '2')
        T(i,3)=1;
    elseif (A(i) == '3')
        T(i,4)=1;
    elseif (A(i) == '4')
        T(i,5)=1;
    elseif (A(i) == '5')
        T(i,6)=1;
    elseif (A(i) == '6')
        T(i,7)=1;
    elseif (A(i) == '7')
        T(i,8)=1;
    elseif (A(i) == '8')
        T(i,9)=1;
    elseif (A(i) == '9')
        T(i,10)=1;
    end
end
```

*Figure 15:**getbintarget()** Function.*

This function will rearrange the pattern into training or testing matric. Since we have ten classes, this will be our sample output. The computer will use these matrics to recognize the patterns.

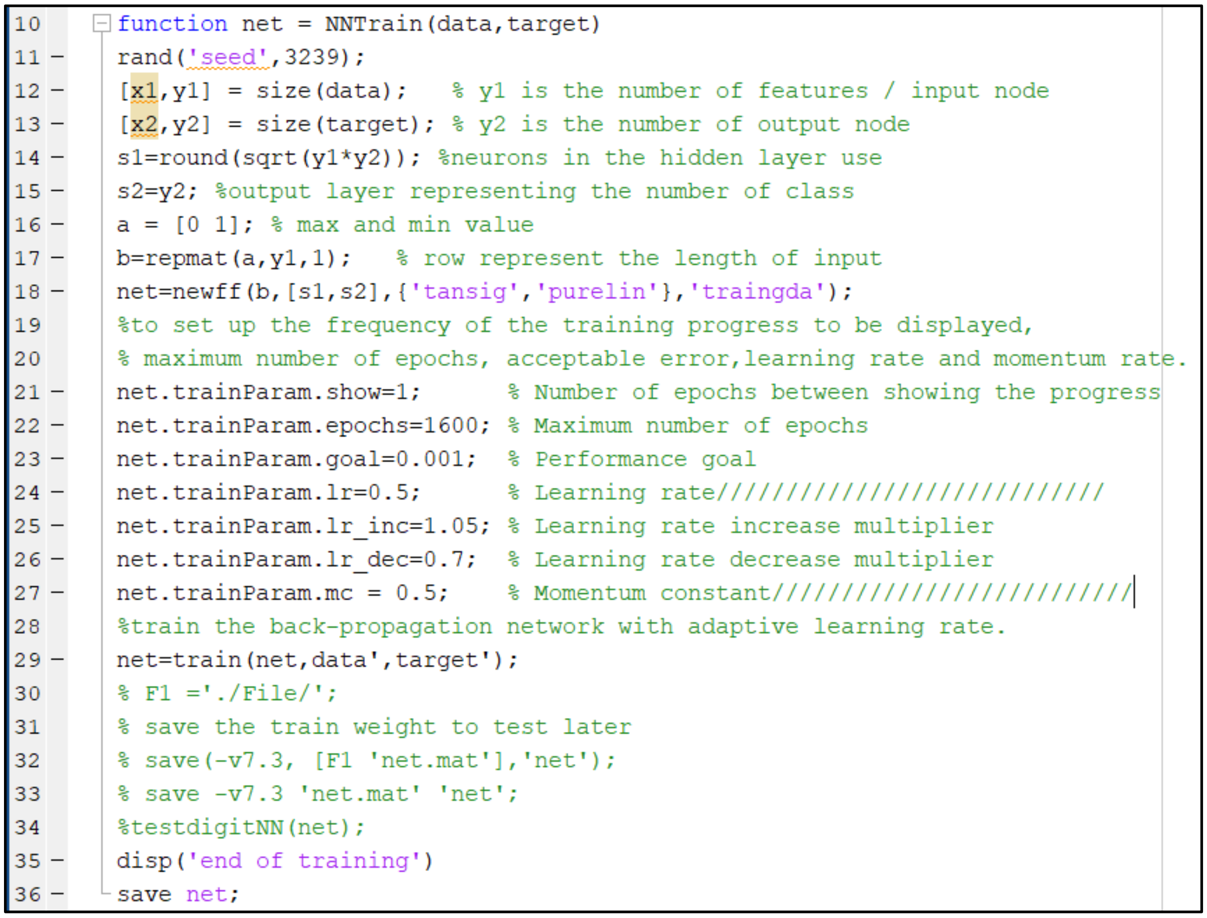


Class	Training / Testing Matric									
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

Figure 16:Sample training or testing matric based on class

3.3 Recognition

After that, ***NNTrain()*** function has been called for recognition training. Then, the train neural network had been saved by using ***save net*** so that no need to train the whole data set every time whenever we want to use it.



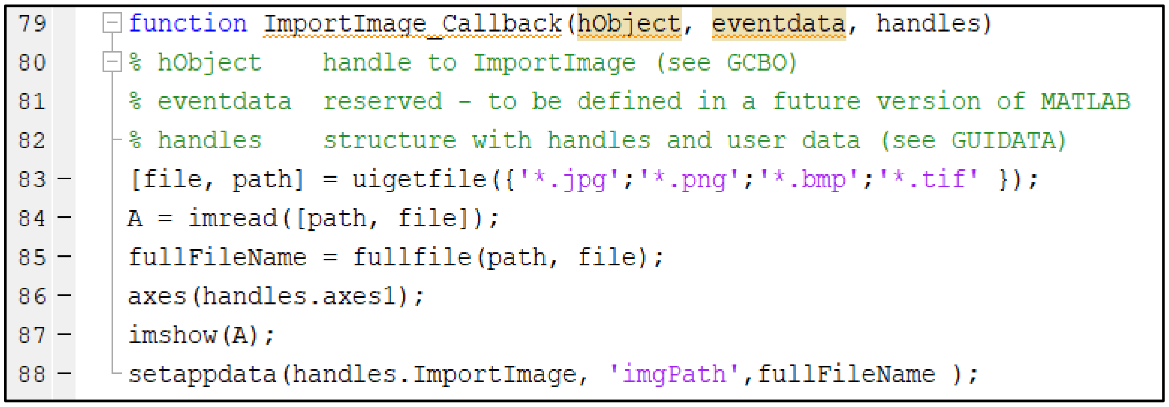
```
function net = NNTrain(data,target)
rand('seed',3239);
[x1,y1] = size(data);    % y1 is the number of features / input node
[x2,y2] = size(target); % y2 is the number of output node
s1=round(sqrt(y1*y2)); %neurons in the hidden layer use
s2=y2; %output layer representing the number of class
a = [0 1]; % max and min value
b=repmat(a,y1,1);    % row represent the length of input
net=newff(b,[s1,s2],{'tansig','purelin'},'traingda');
%to set up the frequency of the training progress to be displayed,
% maximum number of epochs, acceptable error,learning rate and momentum rate.
net.trainParam.show=1;       % Number of epochs between showing the progress
net.trainParam.epochs=1600; % Maximum number of epochs
net.trainParam.goal=0.001;  % Performance goal
net.trainParam.lr=0.5;      % Learning rate////////////////////////////
net.trainParam.lr_inc=1.05; % Learning rate increase multiplier
net.trainParam.lr_dec=0.7;  % Learning rate decrease multiplier
net.trainParam.mc = 0.5;    % Momentum constant/////////////////////////
%train the back-propagation network with adaptive learning rate.
net=train(net,data',target');
% F1 ='./File/';
% save the train weight to test later
% save(-v7.3, [F1 'net.mat'],'net');
% save -v7.3 'net.mat' 'net';
%testdigitNN(net);
disp('end of training')
save net;
```

*Figure 17:**NNTrain()** Function.*

3.4 GUI

ImportImage_Callback() Function

A callback button ***ImportImage_Callback()*** has been set to import the test data. After getting the path of the image, the image will be shown in the axes and save the ***fullFileName*** to then handles.

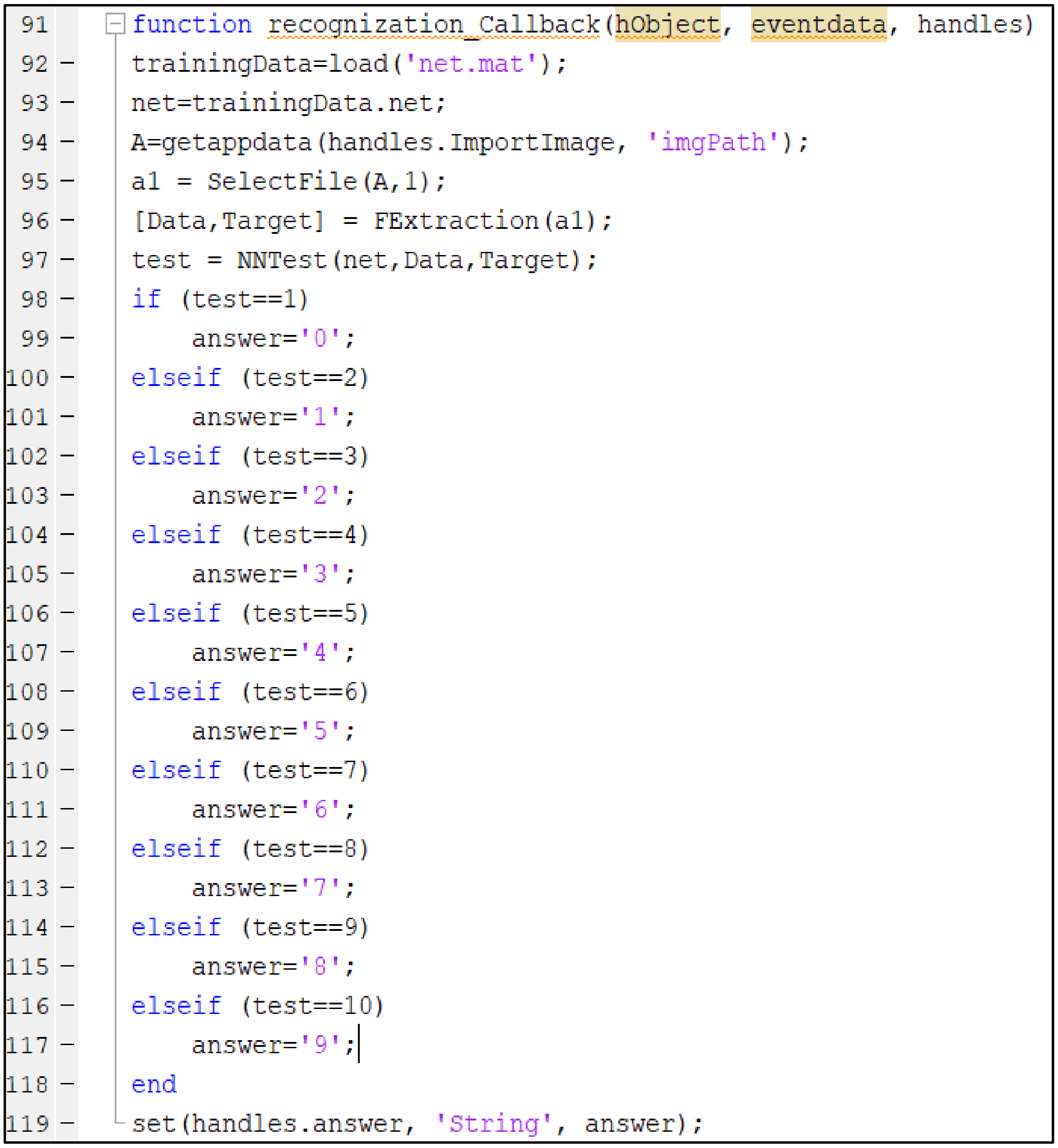


```
function ImportImage_Callback(hObject, eventdata, handles)
% hObject    handle to ImportImage (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[file, path] = uigetfile({'*.jpg';'*.png';'*.bmp';'*.tif' });
A = imread([path, file]);
fullFileName = fullfile(path, file);
axes(handles.axes1);
imshow(A);
setappdata(handles.ImportImage, 'imgPath',fullFileName );
```

*Figure 18:**ImportImage_Callback()** Function.*

recognization_Callback() Function

Besides, another callback button ***recognization_Callback()*** has been set for analyzing and recognize the test data. First, the trained neural network had been load. After that, the image is undergoing a series of processes and extraction to enable the recognition process. Lastly, ***NNTest()*** function have been called to undergo recognition process, if the target data is the same as one of the pattern class, it will show the digit, whereas, the system will show the pattern class that is similar to it, or confuse it.

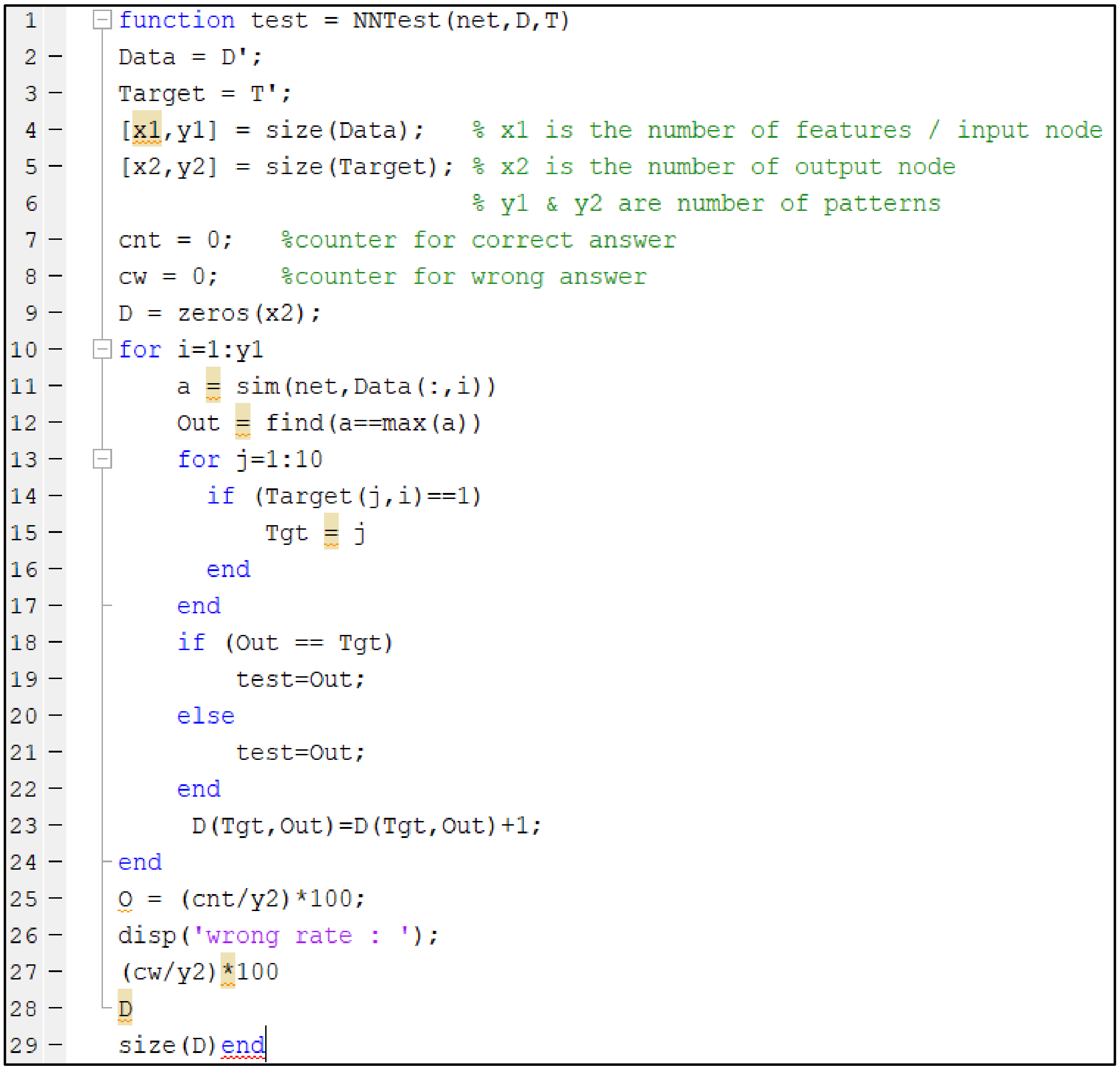


```
function recognization_Callback(hObject, eventdata, handles)
trainingData=load('net.mat');
net=trainingData.net;
A=getappdata(handles.ImportImage, 'imgPath');
a1 = SelectFile(A,1);
[Data,Target] = FExtraction(a1);
test = NNTest(net,Data,Target);
if (test==1)
    answer='0';
elseif (test==2)
    answer='1';
elseif (test==3)
    answer='2';
elseif (test==4)
    answer='3';
elseif (test==5)
    answer='4';
elseif (test==6)
    answer='5';
elseif (test==7)
    answer='6';
elseif (test==8)
    answer='7';
elseif (test==9)
    answer='8';
elseif (test==10)
    answer='9';
end
set(handles.answer, 'String', answer);
```

*Figure 19:**recognition_Callback()** Function.*

Since we have 10 classes, there will be 10 output from 0-9. The system will display the result if the test data match the target. The system will also show the most possible result if the input image is not clear. The ‘answer’ is the string that will be displayed on the GUI.

NNTest() Function



```
function test = NNTest(net,D,T)
Data = D';
Target = T';
[x1,y1] = size(Data);    % x1 is the number of features / input node
[x2,y2] = size(Target); % x2 is the number of output node
                         % y1 & y2 are number of patterns
cnt = 0;    %counter for correct answer
cw = 0;     %counter for wrong answer
D = zeros(x2);
for i=1:y1
    a = sim(net,Data(:,i))
    Out = find(a==max(a))
    for j=1:10
      if (Target(j,i)==1)
          Tgt = j
      end
    end
    if (Out == Tgt)
        test=Out;
    else
        test=Out;
    end
     D(Tgt,Out)=D(Tgt,Out)+1;
end
O = (cnt/y2)*100;
disp('wrong rate : ');
(cw/y2)*100
D
size(D)end
```

*Figure 20:**NNTest()** Function.*

User Workflow

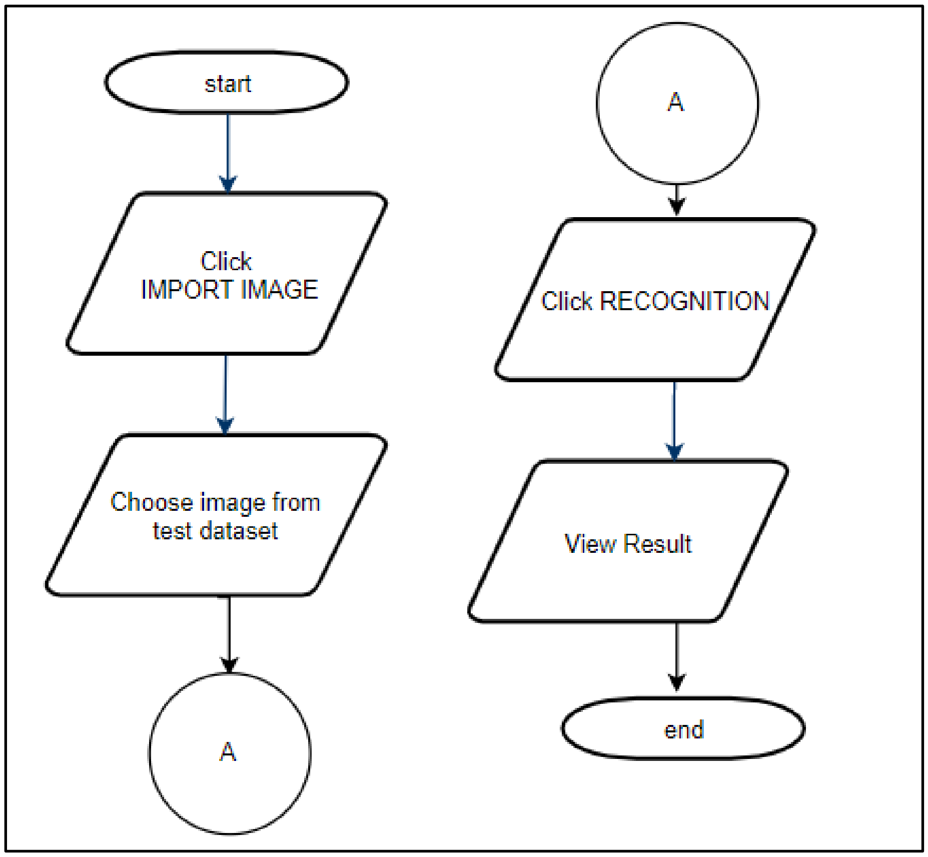


Figure 21:The user workflow when using this system

1. The user needs to click the **IMPORT IMAGE** button in order to import the test image.

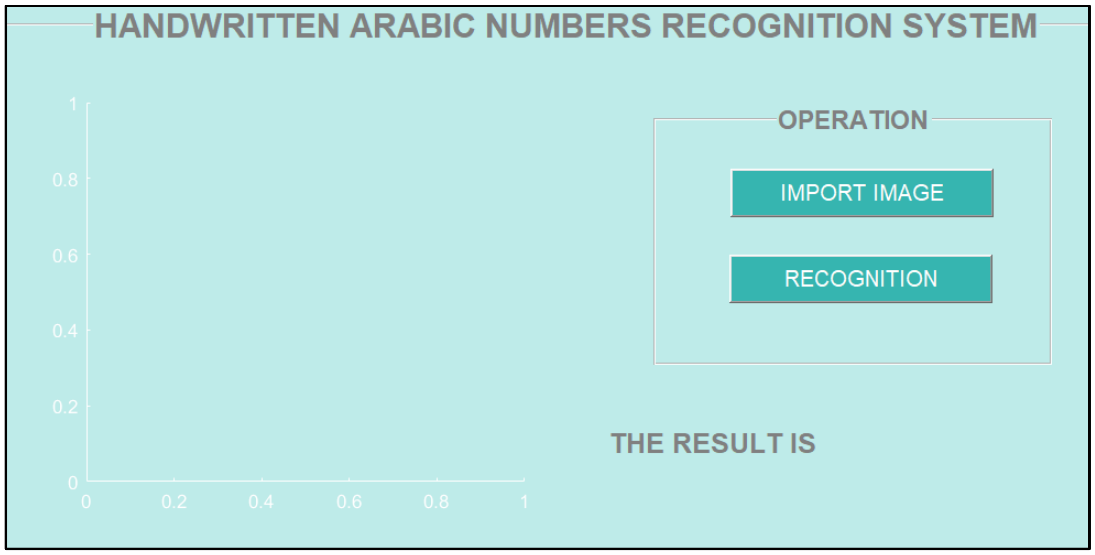


Figure 22: GUI of the system.

2. The user can choose an image from the Test Data dataset to be tested..

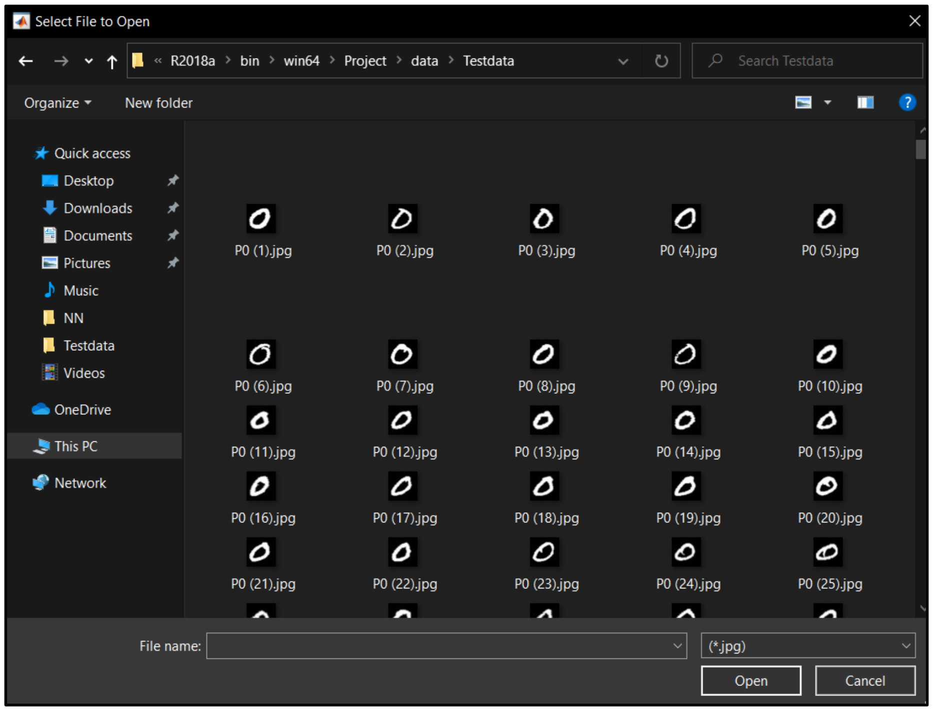


Figure 23: A window will pop up to enable the user to choose the test file.

3. The user has to click **RECOGNITION** button to start the recognition process.

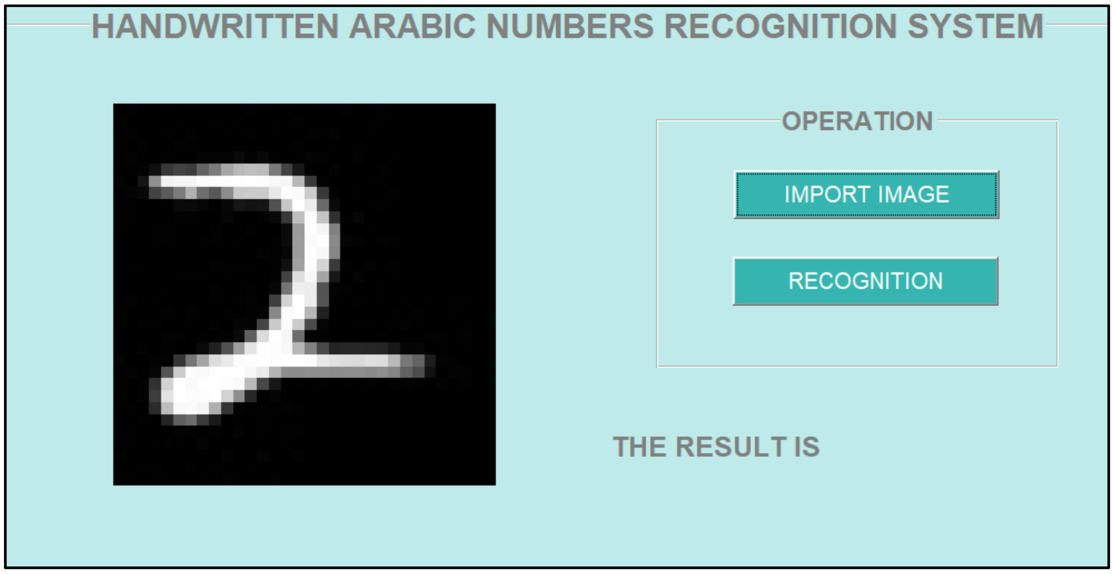


Figure 24: The chosen image will be shown at the axes.

4. The user can view the result.

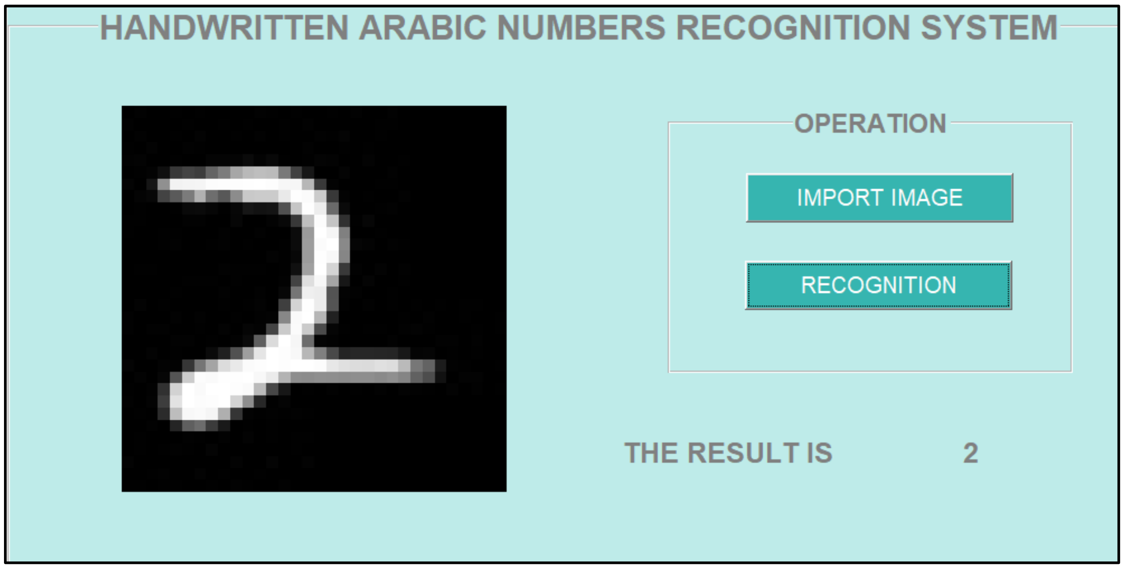
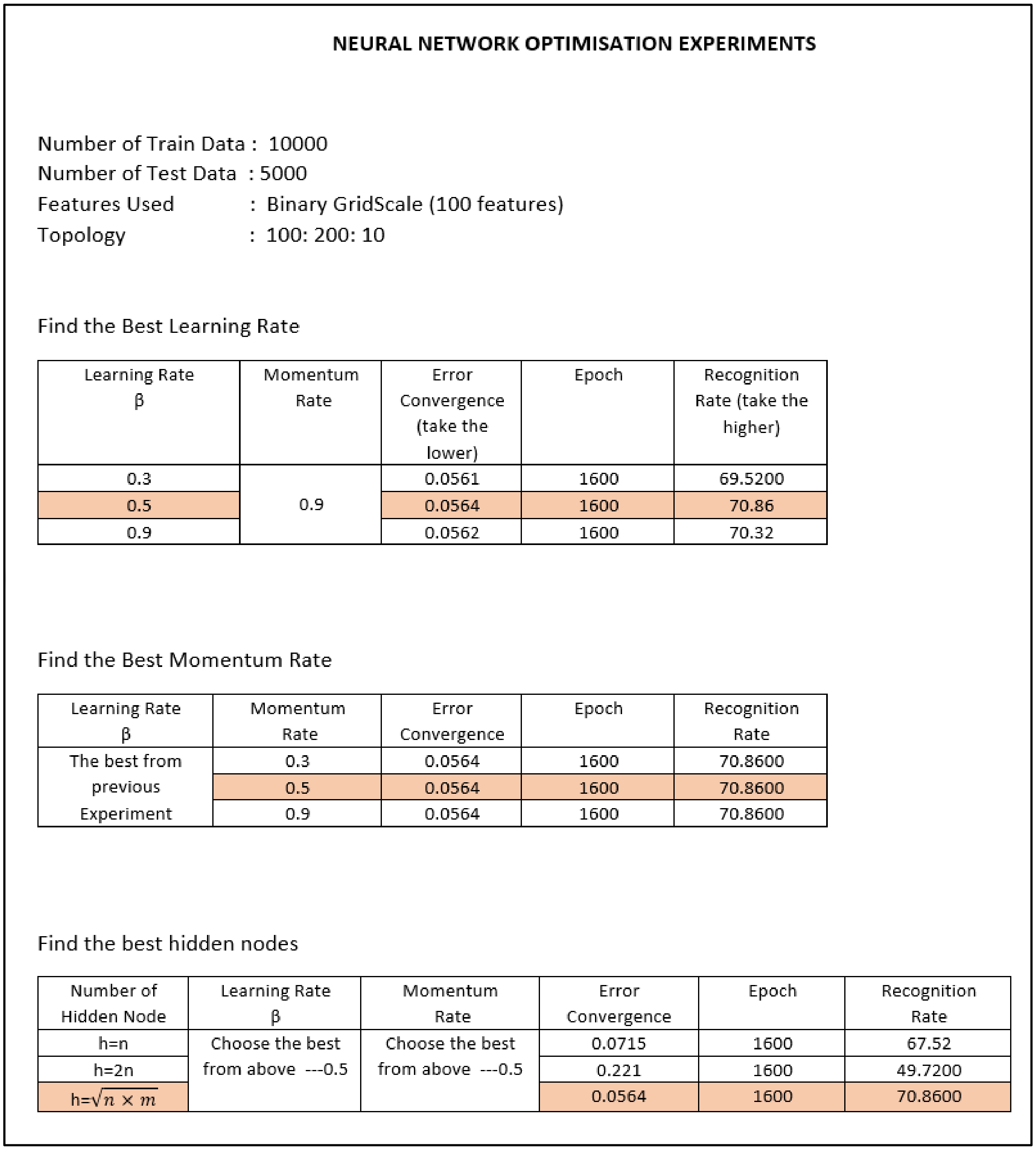


Figure 25: The result will be shown

4. EXPERIMENT RESULT

Experiment Setup

From our experiment sheet above, the number of train data we used is 10000 and we used 5000 for our test data. The featured used is 10 x 10 binary grid-scale which contains 100 features. The topology we used is 100:200:10. For the best learning rate, we chose the learning rate of 0.5. This is because the learning rate of 0.5 produces the highest recognition rate which is 70.86 although the error convergence is highest which is 0.0564. But since the differences only a little bit so we still decided to choose the learning rate of 0.5. For the best momentum rate, we chose the momentum rate of 0.5 since all the recognition rates produced from using different momentum rates are the same. So, we decided to choose the middle value which is 0.5. For the best number of hidden nodes, we chose h = sqrt of n times m since the recognition rate is the highest among the others.



NEURAL NETWORK OPTIMISATION EXPERIMENTS

Number of Train Data : 10000
Number of Test Data : 5000
Features Used : Binary GridScale (100 features)
Topology : 100: 200: 10

Find the Best Learning Rate

Learning Rate β	Momentum Rate	Error Convergence (take the lower)	Epoch	Recognition Rate (take the higher)
0.3	0.9	0.0561	1600	69.5200
0.5		0.0564	1600	70.86
0.9		0.0562	1600	70.32

Find the Best Momentum Rate

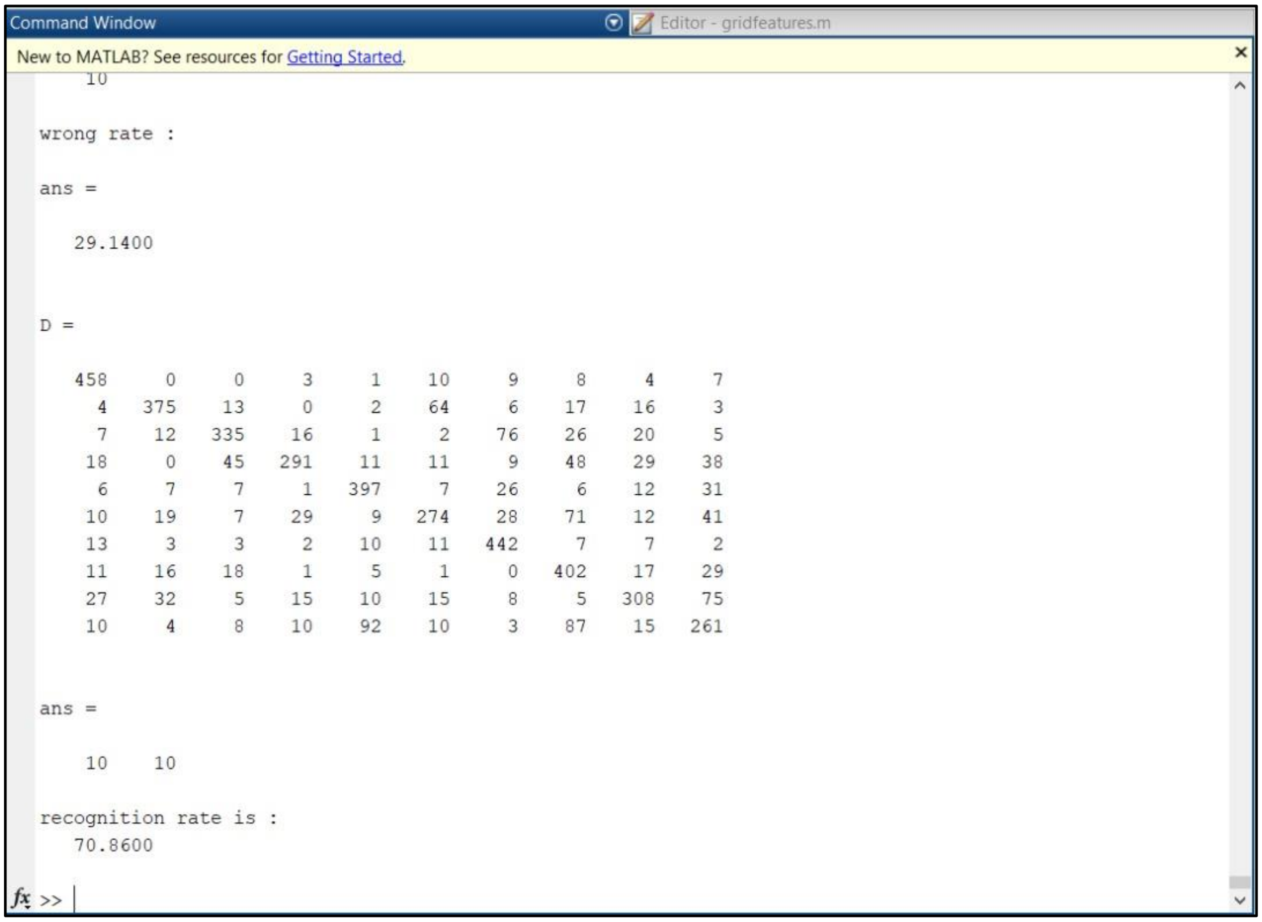
Learning Rate β	Momentum Rate	Error Convergence	Epoch	Recognition Rate
The best from previous Experiment	0.3	0.0564	1600	70.8600
	0.5	0.0564	1600	70.8600
	0.9	0.0564	1600	70.8600

Find the best hidden nodes

Number of Hidden Node	Learning Rate β	Momentum Rate	Error Convergence	Epoch	Recognition Rate
h=n	Choose the best from above ---0.5	Choose the best from above ---0.5	0.0715	1600	67.52
h=2n			0.221	1600	49.7200
$h=\sqrt{n \times m}$			0.0564	1600	70.8600

Figure 26: Results of the experiment testing sheet

Recognition Rate

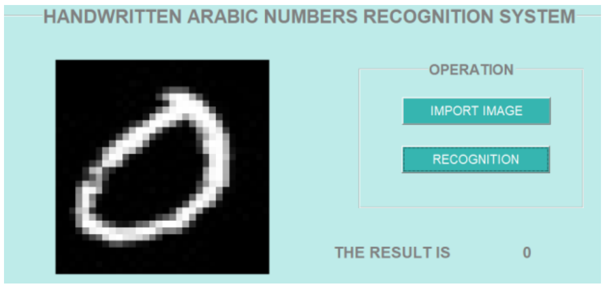


```
    10

wrong rate :

ans =

   29.1400


D =

   458     0     0     3     1    10     9     8     4     7
     4   375    13     0     2    64     6    17    16     3
     7    12   335    16     1     2    76    26    20     5
    18     0    45   291    11    11     9    48    29    38
     6     7     7     1   397     7    26     6    12    31
    10    19     7    29     9   274    28    71    12    41
    13     3     3     2    10    11   442     7     7     2
    11    16    18     1     5     1     0   402    17    29
    27    32     5    15    10    15     8     5   308    75
    10     4     8    10    92    10     3    87    15   261


ans =

    10    10

recognition rate is :
   70.8600

fx >> 
```

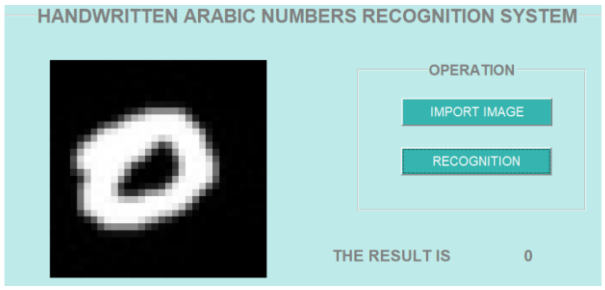
Figure 27: Recognition rate.

After the training session, the testing session has been carried out and the recognition rate is 70.86. The recognition rate is unsatisfactory and the main reasons are due to the data set is handwritten. People may have different styles of writing Arabic numbers such as the ‘open 4’ or ‘close 4’ and the 7 and 7 with a horizontal line. Also, some of the datasets have bad handwriting. Therefore, these handwritten may confuse the system recognition. We are not purposely filtering the train and test data because we want to get a natural result as the system may not always get the same type of writing all the time. We need to make sure that the system is able to recognize different style in the future. However, most of the handwritten still can be recognized correctly.

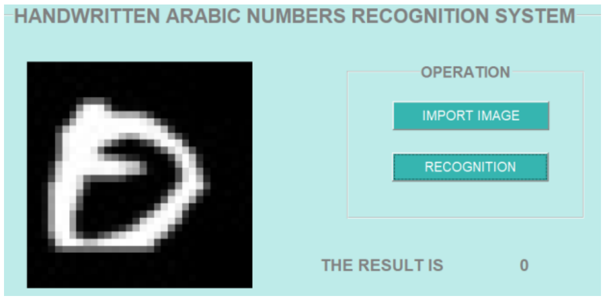
4.1 Digit Pattern : 0



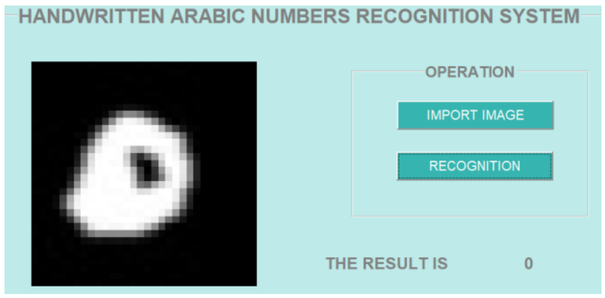
THE RESULT IS 0



THE RESULT IS 0

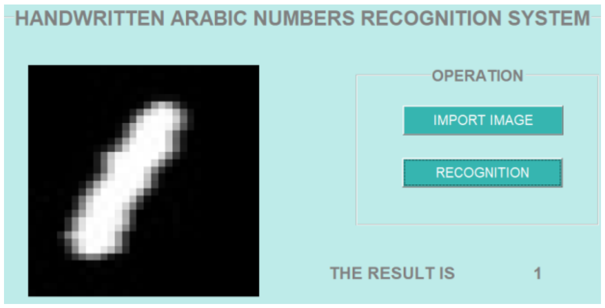


THE RESULT IS 0

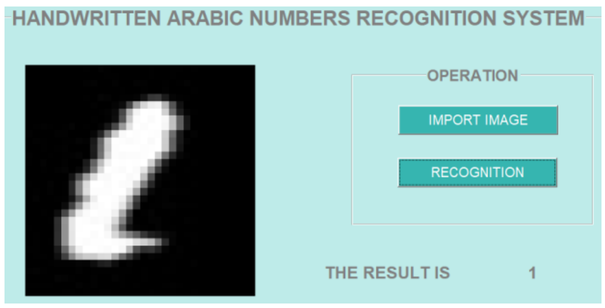


THE RESULT IS 0

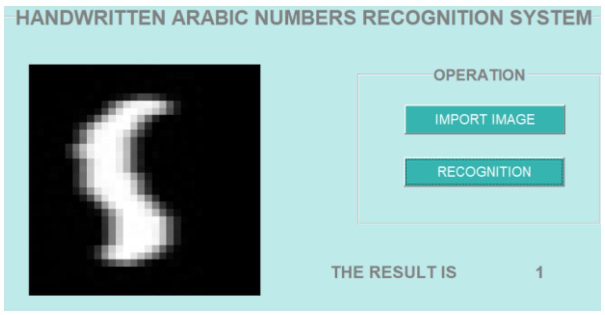
4.2 Digit Pattern : 1



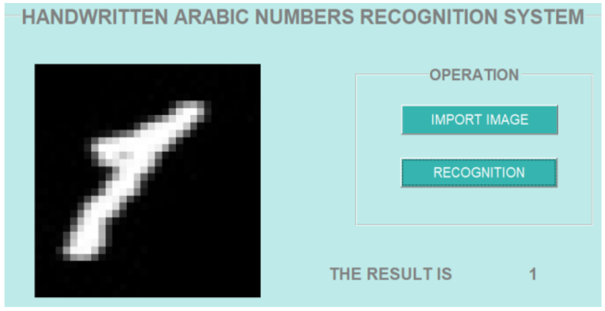
THE RESULT IS 1



THE RESULT IS 1

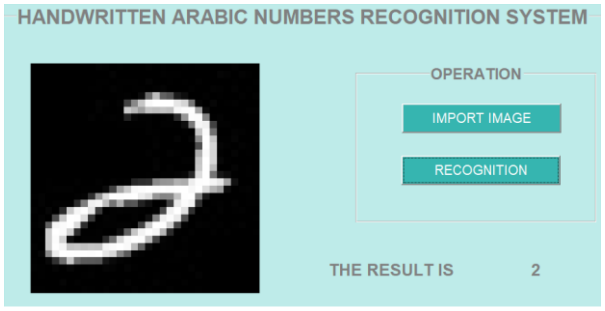


THE RESULT IS 1

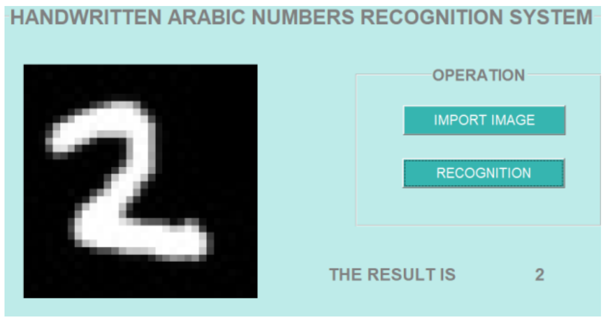


THE RESULT IS 1

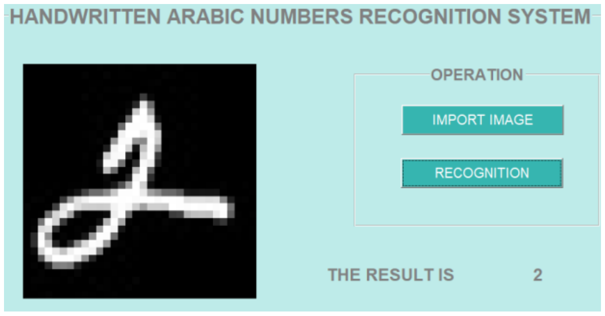
4.3 Digit Pattern : 2


HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

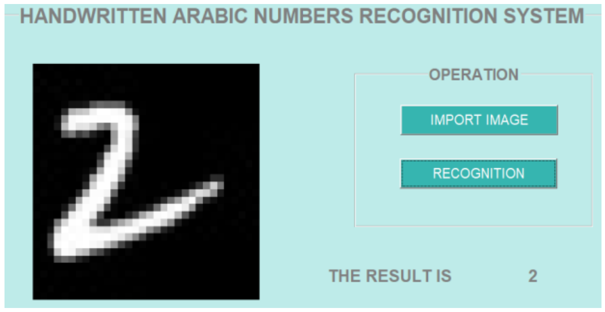
THE RESULT IS 2


HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

THE RESULT IS 2

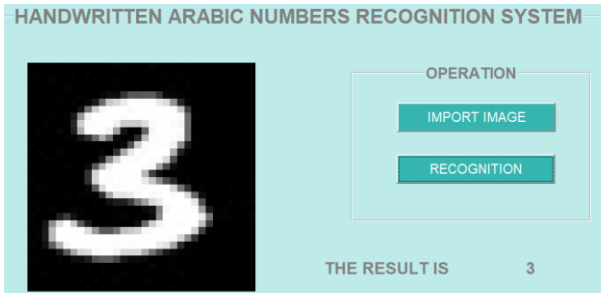

HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

THE RESULT IS 2

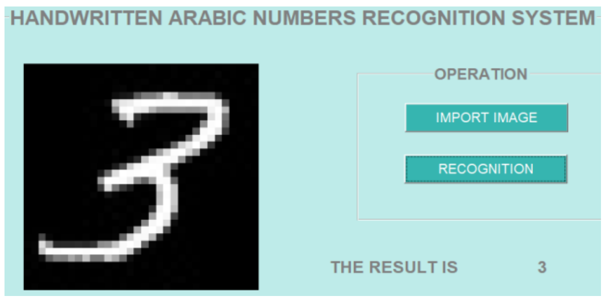

HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

THE RESULT IS 2

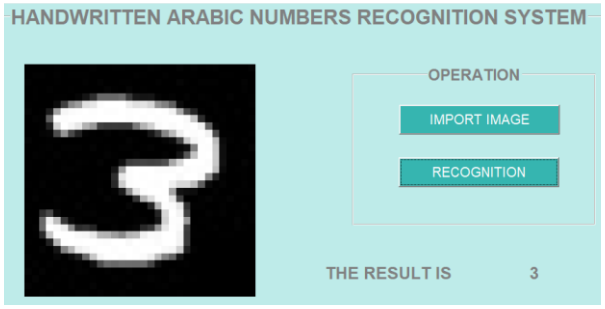
4.4 Digit Pattern : 3


HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

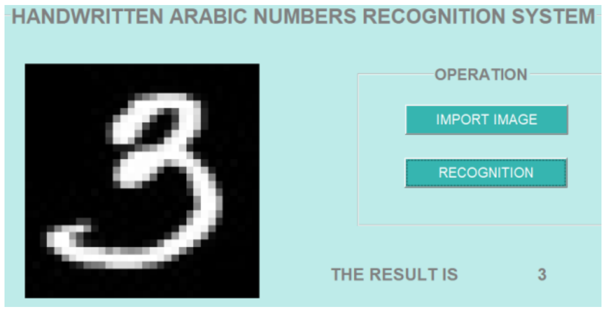
THE RESULT IS 3


HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

THE RESULT IS 3

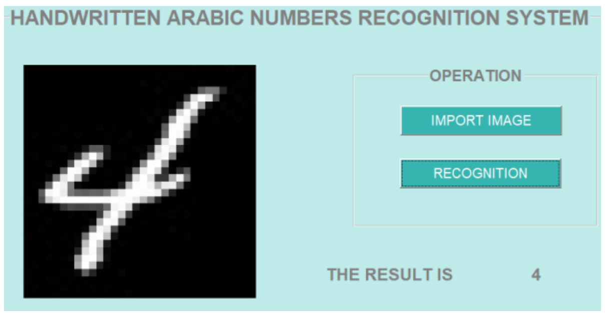

HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

THE RESULT IS 3

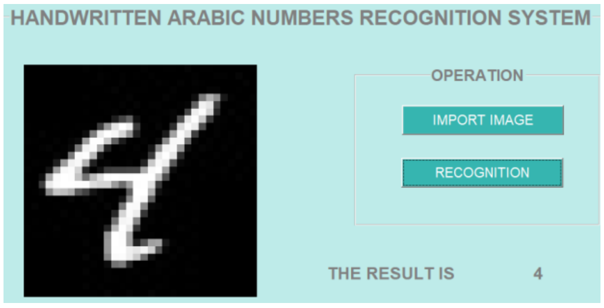

HANDWRITTEN ARABIC NUMBERS RECOGNITION SYSTEM

THE RESULT IS 3

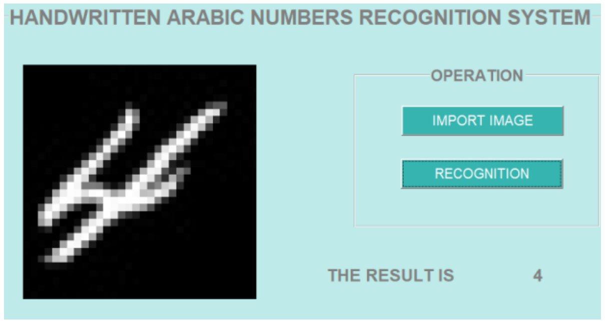
4.5 Digit Pattern : 4



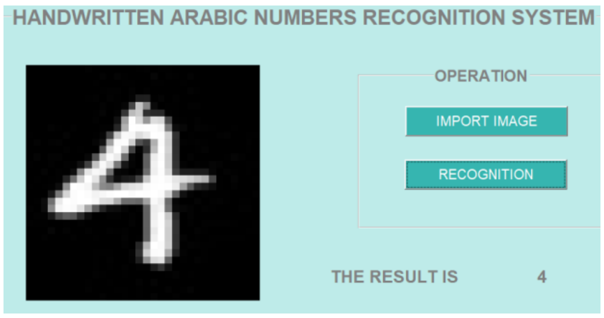
THE RESULT IS 4



THE RESULT IS 4

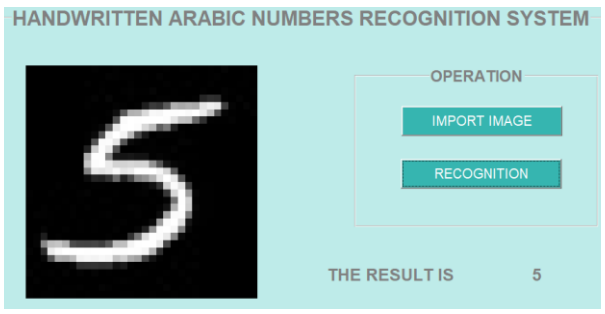


THE RESULT IS 4

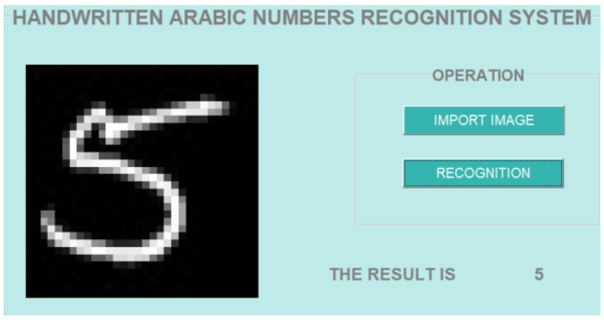


THE RESULT IS 4

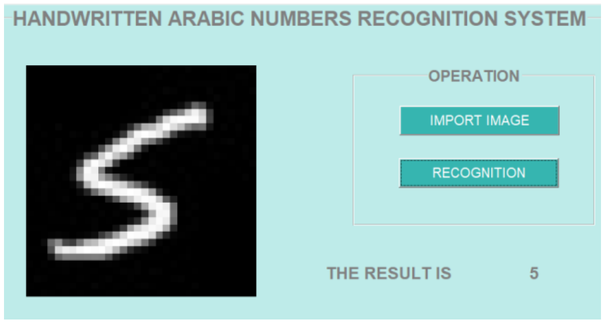
4.6 Digit Pattern : 5



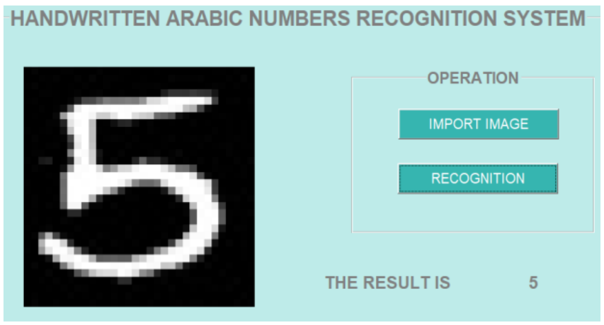
THE RESULT IS 5



THE RESULT IS 5

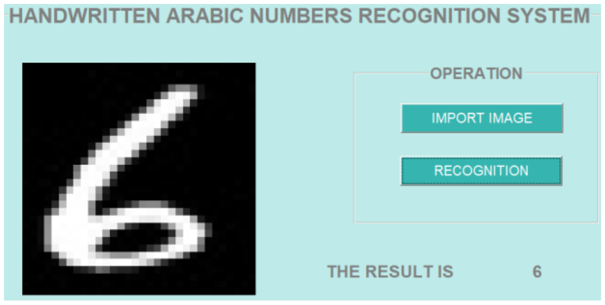


THE RESULT IS 5

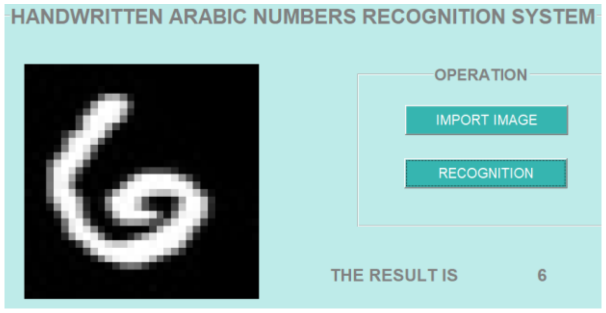


THE RESULT IS 5

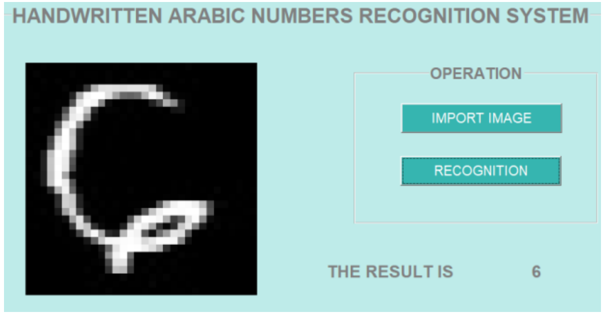
4.7 Digit Pattern : 6



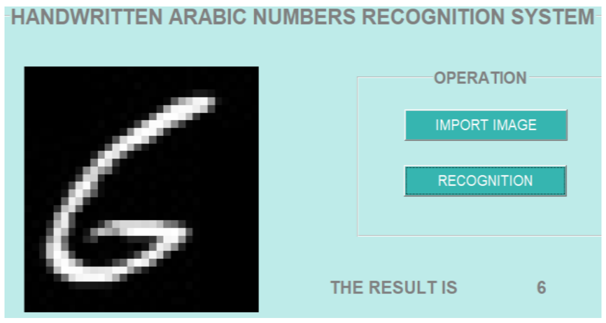
THE RESULT IS 6



THE RESULT IS 6

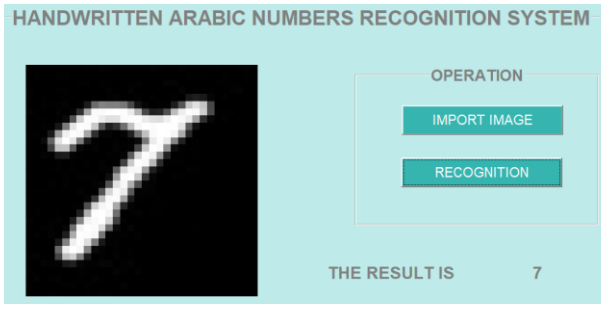


THE RESULT IS 6

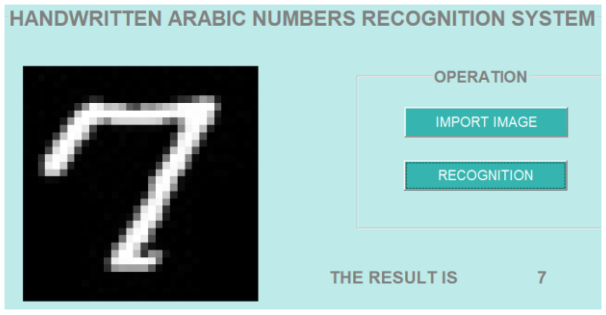


THE RESULT IS 6

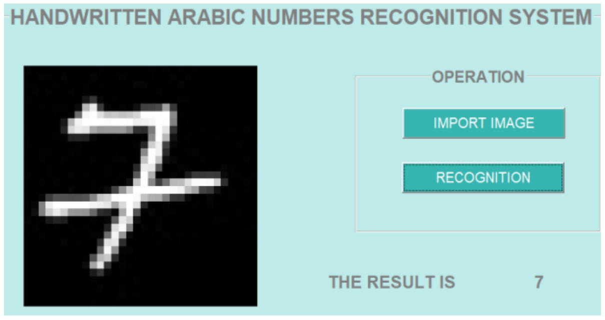
4.8 Digit Pattern : 7



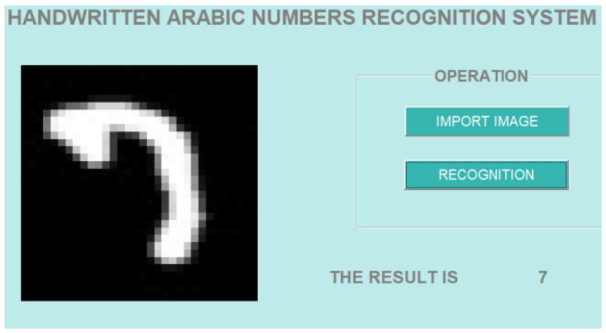
THE RESULT IS 7



THE RESULT IS 7

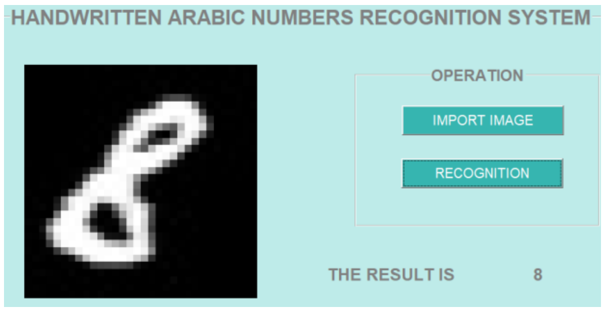


THE RESULT IS 7

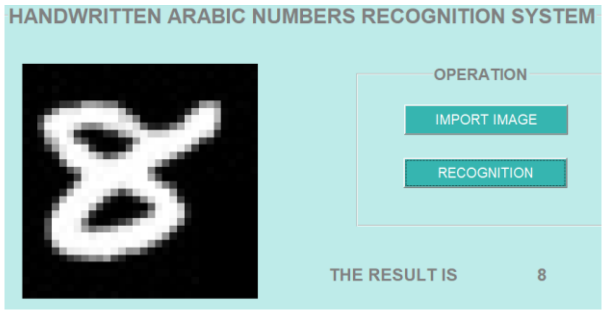


THE RESULT IS 7

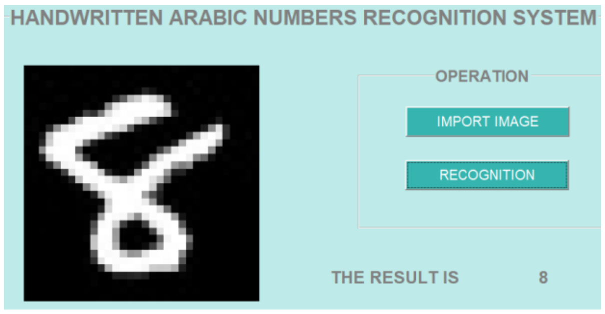
4.9 Digit Pattern : 8



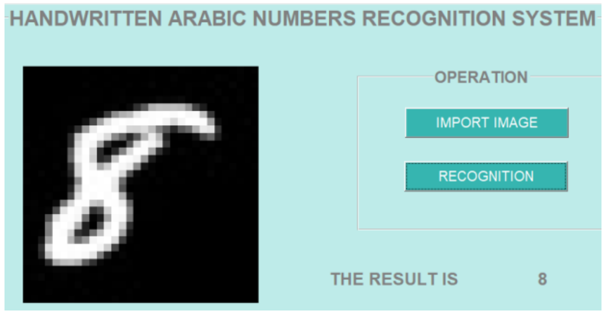
THE RESULT IS 8



THE RESULT IS 8

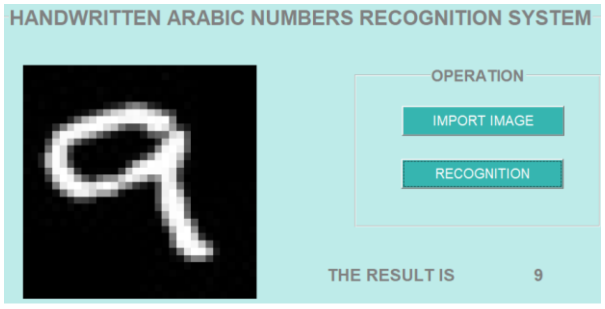


THE RESULT IS 8

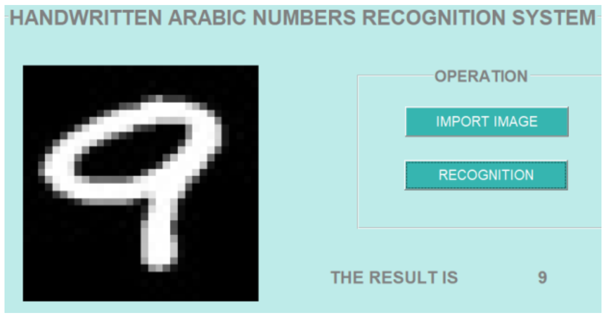


THE RESULT IS 8

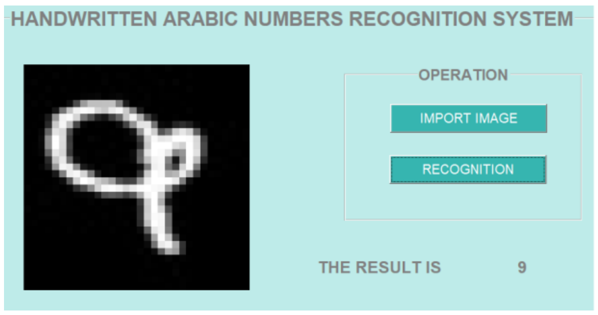
4.10 Digit Pattern : 9



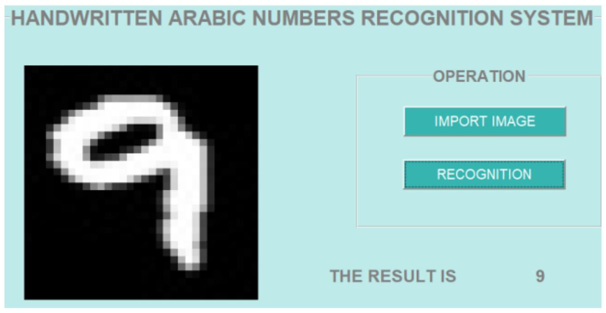
THE RESULT IS 9



THE RESULT IS 9



THE RESULT IS 9



THE RESULT IS 9

5. CRITERIA

Feature Extraction steps

1. Import image
2. Convert to binary image
3. Median filter 3x3
4. Resize 100x100

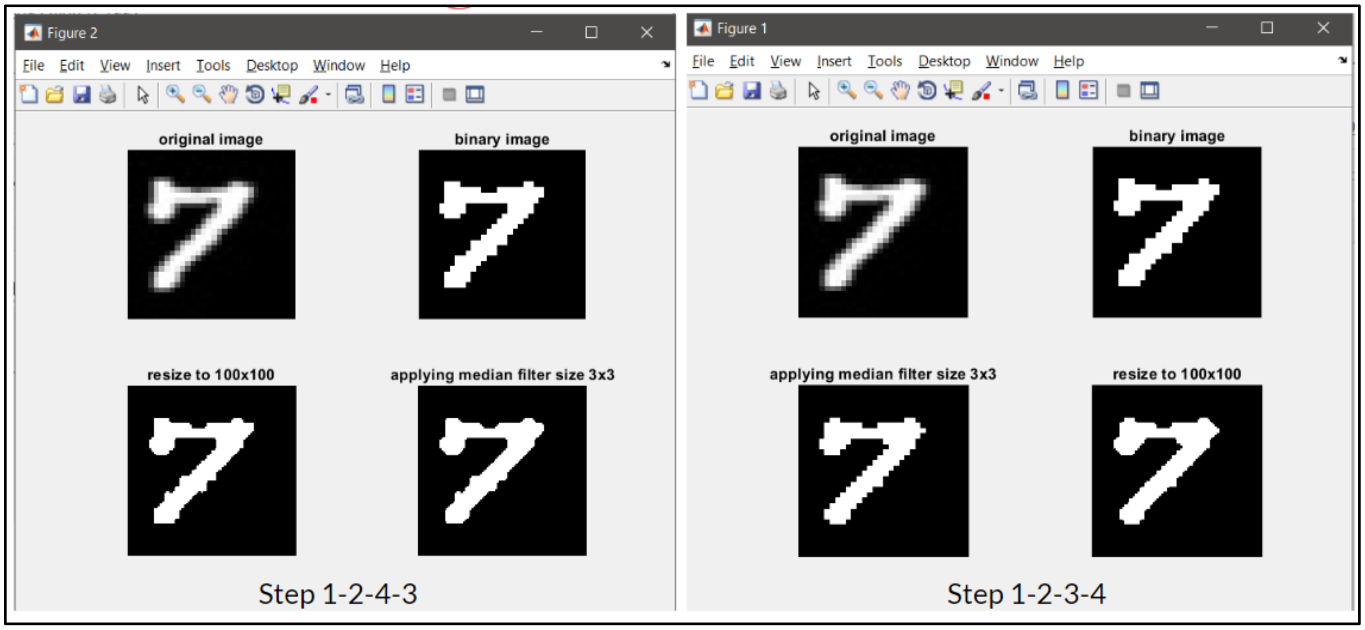


Figure 2
File Edit View Insert Tools Desktop Window Help
original image
binary image
resize to 100x100
applying median filter size 3x3
Step 1-2-4-3
Figure 1
File Edit View Insert Tools Desktop Window Help
original image
binary image
applying median filter size 3x3
resize to 100x100
Step 1-2-3-4

Figure 28: Output using different sequence of preprocessing steps

When using resize before median filter (left figure):

From the figure above, we can see that the final output is not really good as there are some pointy parts at the lower part of the number 7.

When using the median filter before resize (right figure):

From the figure above, we can see that the final output is better compared to the previous output as there are less pointy parts at the lower part of the number 7 which will helps the machine to recognize the number easier.

Conclusion

Best sequence: 1-2-3-4

Median Filter Size

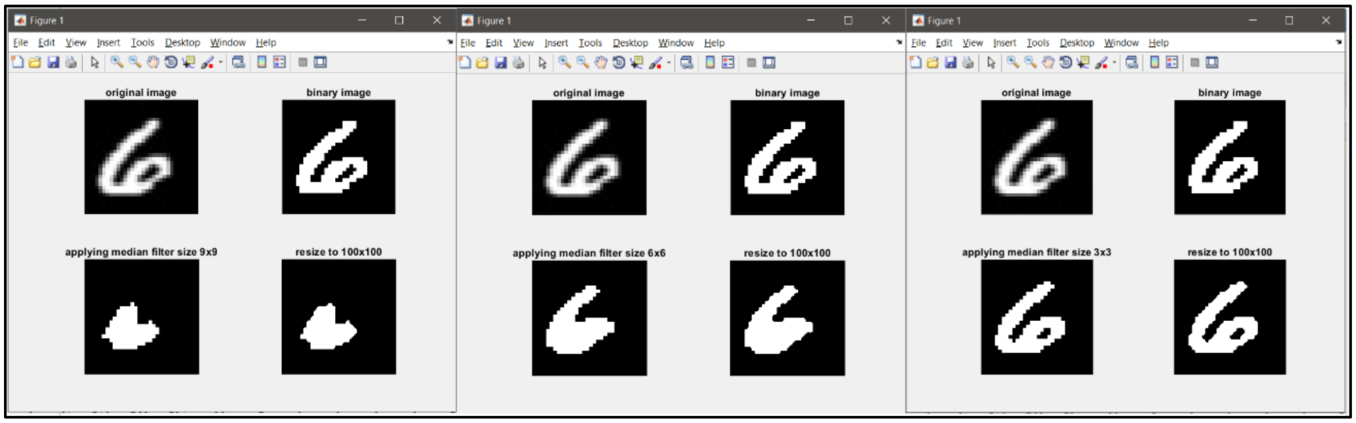


Figure 1
File Edit View Insert Tools Desktop Window Help
original image
binary image
applying median filter size 9x9
resize to 100x100
applying median filter size 6x6
resize to 100x100
applying median filter size 3x3
resize to 100x100

Figure 29: Output using median filter size 9x9, 6x6, 3x3

Median Filter 9x9 (left figure):

From the figure above, we can see that the output of number 6 is unrecognizable when the median filter size of 9 x 9 is applied to it.

Median Filter 6x6 (middle figure):

From the figure above, although the outline shape of the output for the number 6 can be obtained, the empty space part of the number is filled with white colors. Although the output looks better compared to the output using the median filter size of 9x9, the number 6 is still unrecognizable and it will affect our results.

Median Filter 3x3 (right figure):

From the figure above, we can see that the output using median filter size 3x3 is the best among the 3 outputs. The output looks similar to the original image and more clearer than the original image which will help the machine to easily recognize the image. That’s the reason we chose the median filter with a size of 3 x 3.

Conclusion

Best median filter size: 3x3

Resize Factor

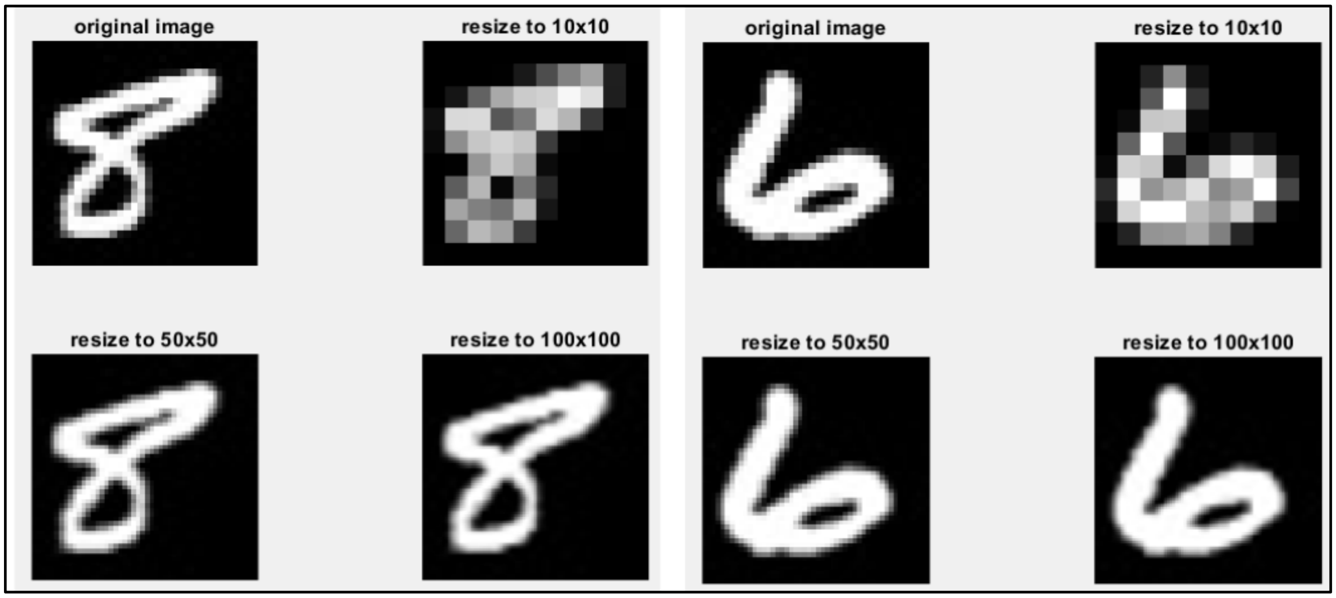


Figure 30: Output using resize factor 10x10, 50x50 and 100x100

Resize 10x10:

From the figure above, we can see that the output of number is unrecognizable when the resize of size 10x10 is applied to it.

Resize 50x50:

From the figure above, although the outline shape for the number can be obtained, the output image is still blur.

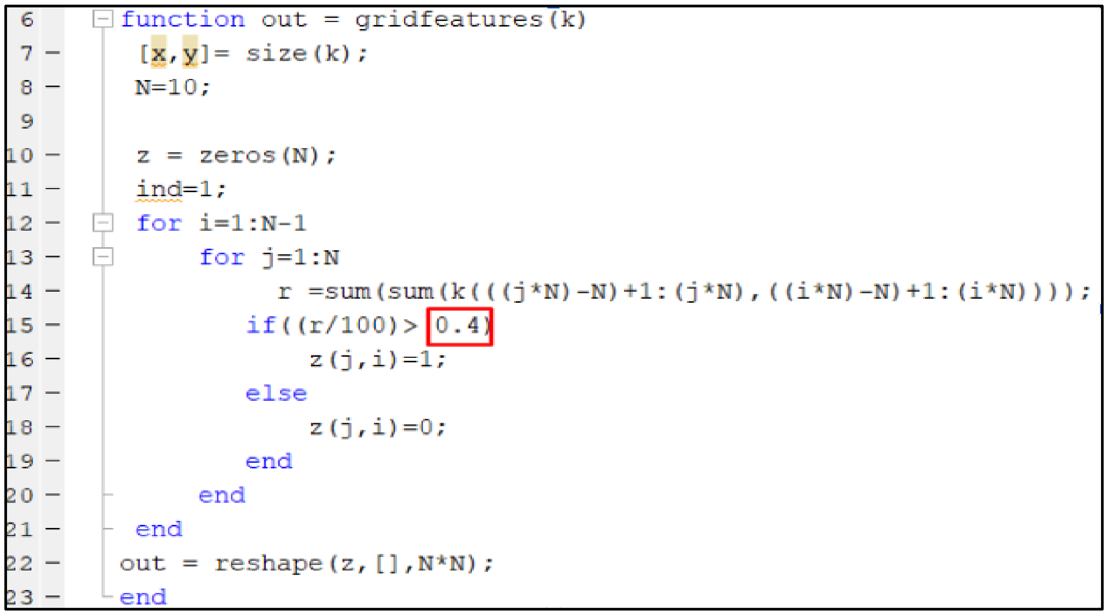
Resize 100x100:

From the figure above, we can see that the output using resize 100x100 is the best among the 3 outputs. The output looks similar to the original image and more clearer than the original image which will help the machine to easily recognize the image. That’s the reason we chose the resize with 100x100.

Conclusion

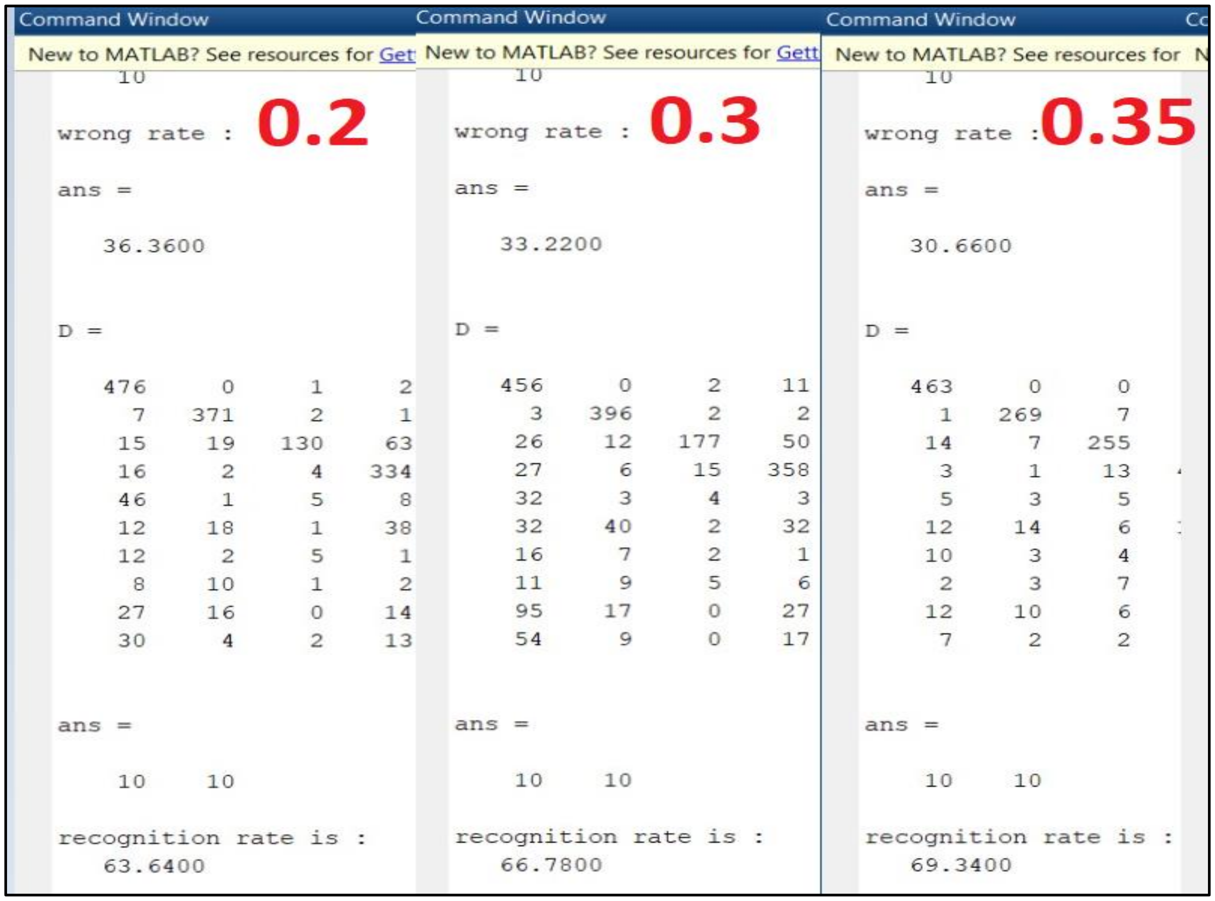
Best resize factor: 100x100

Grid Features Variable



```
function out = gridfeatures(k)
 [x,y]= size(k);
 N=10;

 z = zeros(N);
 ind=1;
 for i=1:N-1
     for j=1:N
          r =sum(sum(k(((j*N)-N)+1:(j*N),((i*N)-N)+1:(i*N))));
        if((r/100)> 0.4)
            z(j,i)=1;
        else
            z(j,i)=0;
        end
     end
 end
out = reshape(z,[],N*N);
end
```

Figure 31: Sample function using 0.4.

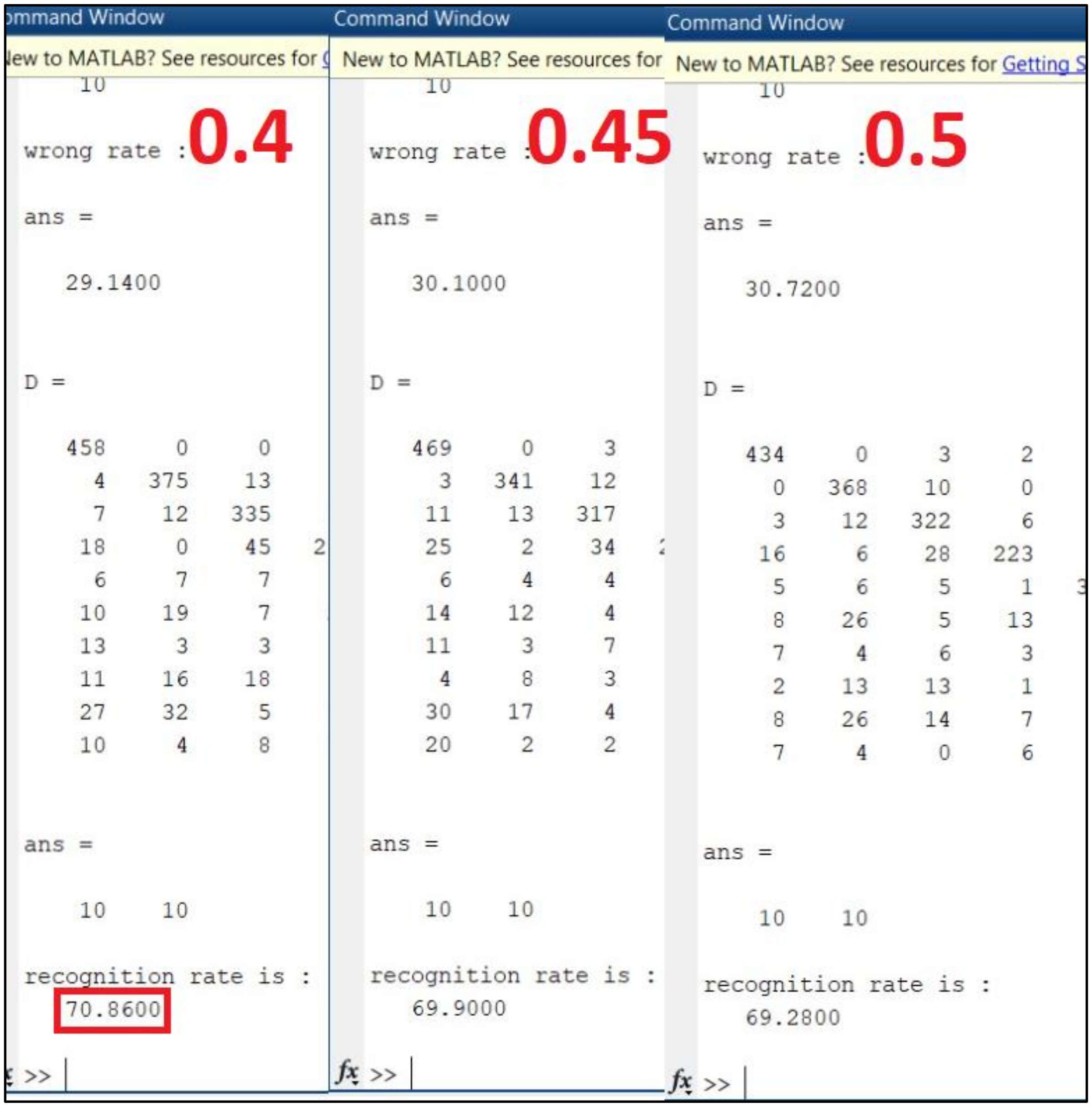


```
Command Window
New to MATLAB? See resources for Get
   10

wrong rate : 0.2

ans =

    36.3600


D =

   476     0     1     2
     7   371     2     1
    15    19   130    63
    16     2     4   334
    46     1     5     8
    12    18     1    38
    12     2     5     1
     8    10     1     2
    27    16     0    14
    30     4     2    13


ans =

    10    10

recognition rate is :
   63.6400
```

```
Command Window
New to MATLAB? See resources for Gett
   10

wrong rate : 0.3

ans =

    33.2200


D =

   456     0     2    11
     3   396     2     2
    26    12   177    50
    27     6    15   358
    32     3     4     3
    32    40     2    32
    16     7     2     1
    11     9     5     6
    95    17     0    27
    54     9     0    17


ans =

    10    10

recognition rate is :
   66.7800
```

```
Command Window
New to MATLAB? See resources for
   10

wrong rate :0.35

ans =

    30.6600


D =

   463     0     0
     1   269     7
    14     7   255
     3     1    13
     5     3     5
    12    14     6
    10     3     4
     2     3     7
    12    10     6
     7     2     2


ans =

    10    10

recognition rate is :
   69.3400
```

Figure 32: Sample recognition rate using 0.2, 0.3 and 0.35



```
   10

wrong rate :

ans =

   29.1400


D =

   458     0     0
     4   375    13
     7    12   335
    18     0    45   2
     6     7     7
    10    19     7
    13     3     3
    11    16    18
    27    32     5
    10     4     8


ans =

    10    10

recognition rate is :
   70.8600
>>
```

```
   10

wrong rate :

ans =

   30.1000


D =

   469     0     3
     3   341    12
    11    13   317
    25     2    34
     6     4     4
    14    12     4
    11     3     7
     4     8     3
    30    17     4
    20     2     2


ans =

    10    10

recognition rate is :
   69.9000
>>
```

```
   10

wrong rate :

ans =

   30.7200


D =

   434     0     3     2
     0   368    10     0
     3    12   322     6
    16     6    28   223
     5     6     5     1   3
     8    26     5    13
     7     4     6     3
     2    13    13     1
     8    26    14     7
     7     4     0     6


ans =

    10    10

recognition rate is :
   69.2800
>>
```

Figure 33: Sample recognition rate using 0.4, 0.45 and 0.5

Grid Features Variable:

From the figure above, we can see that the recognition rate using 0.4 is the highest among the others. So we choose 0.4 to be the grid features variable.

Conclusion

Best grid features variable: 0.4

EXAMPLE TEST DATA WHICH CANNOT RECOGNIZE

P4(19)

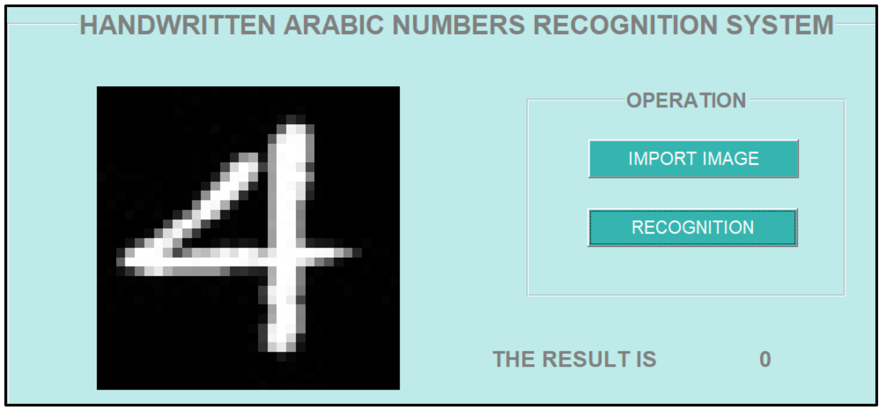


Figure 34: Sample data that failed to be recognized.

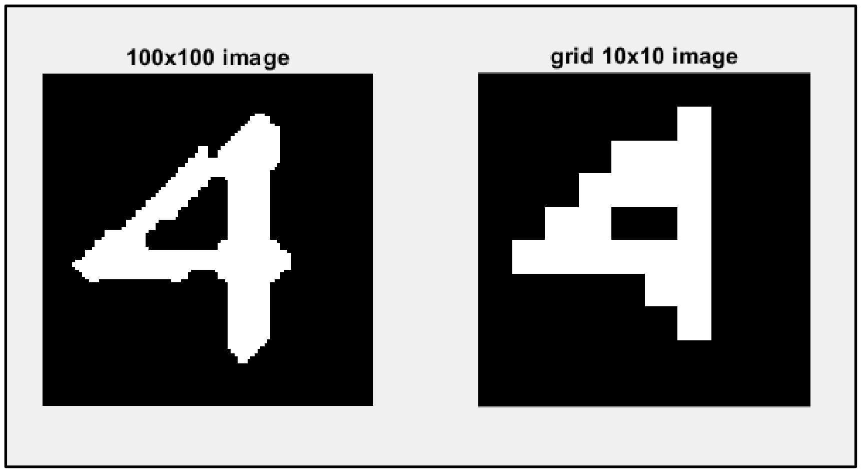


Figure 35:The output after gridfeature() function.

The system recognise this 4 as 0 because majority of the training data is the 'open 4' thus the 'closed 4' will be recognized as 0 since it consists of 1 black part in the middle and the bottom part is smooth just like a 0.

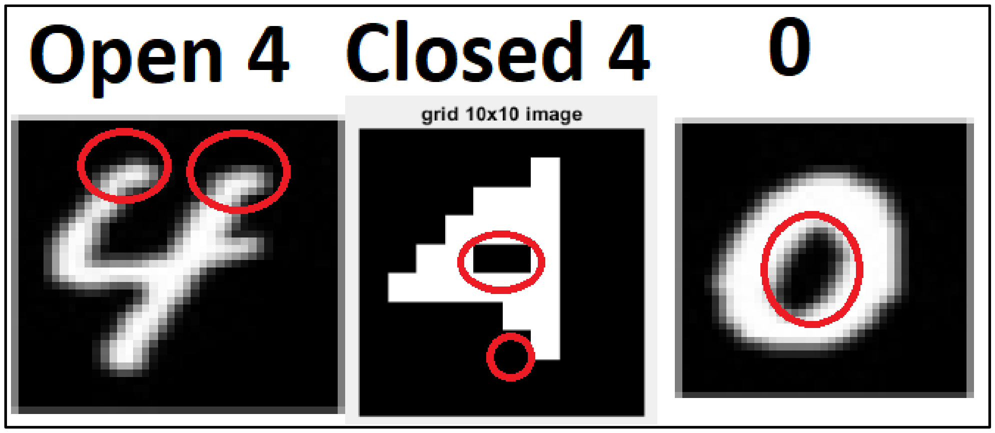


Figure 36: Critical analysis.

P5(108)

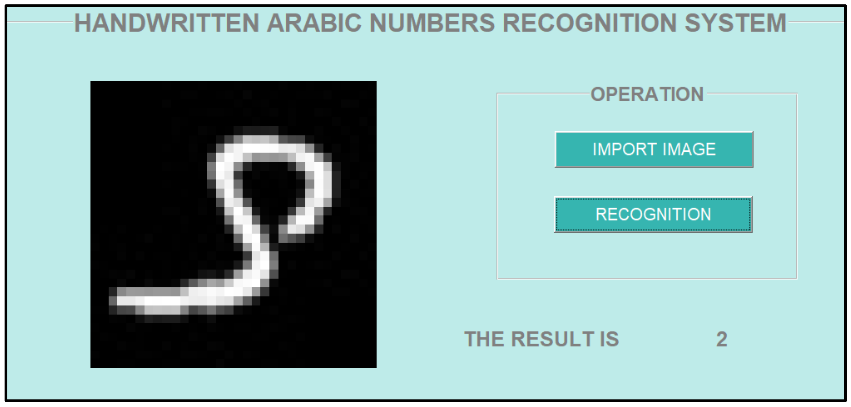


Figure 37: Sample data that failed to be recognized.

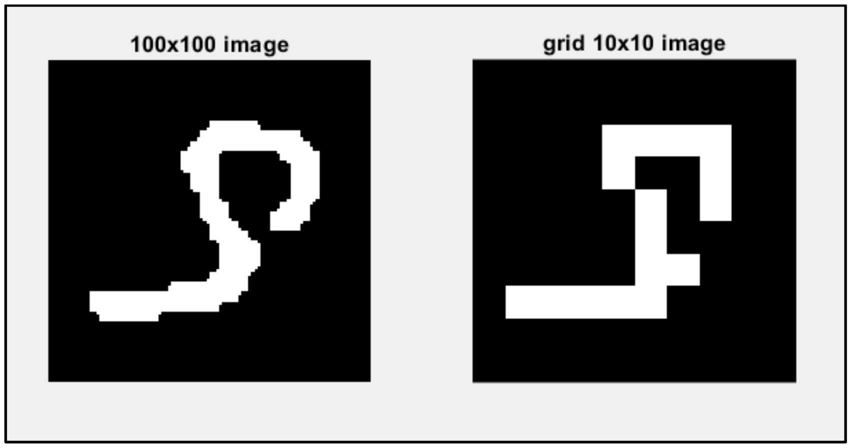


Figure 38: The output after gridfeature() function.

This is definitely a bad handwriting because we are not filtering the test and train data to have a natural result. The bad handwriting 5 has some similar black and white portion like a number 2. Thus, the system recognise it as a 2.

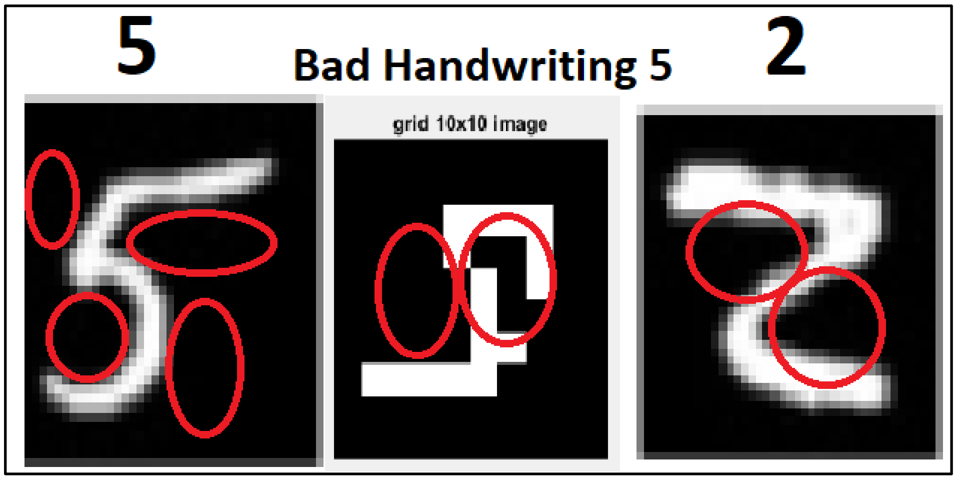


Figure 39: Critical analysis.

6. CONCLUSION

In conclusion, neural network is really a useful and beneficial algorithms that helps in machine learning which can be applied in different fields. After completing this project, we have understood more about how the neural networks work since we are new to the neural network. Before training the machine, we have to find the suitable train data and the number of the trained data has to be suitable and enough too to obtain a better result for the testing process. We also learnt how to work together as a group while doing this project as we have to divide our tasks among us. Besides, we also learnt how to communicate with each other while we are having discussions regarding this project. We also learnt how to help each other. When any of us having any trouble while doing this project, all the team members are willing to help by giving suggestions or useful comments.

We also want to take this opportunity to thank our lovely lecturer, Dr Md. Sah Bin Hj. Salam A who had taught us in image processing subject and willing to teach us about the neural network so that we can have the extra knowledge on it and learn how to neural network approach in this project. We also want to thank him for guiding us when we met some problems and troubles in this project so that we can complete this project successfully and on time. Lastly, we also want to thank our friends and our coursemates who were willing to share their knowledge and give us some ideas and guidance to complete this project. Without them, it is impossible for us to complete this project.

7. REFERENCE

Corey Messer. (2016, April 25). Matlab Neural Network Number Recognition [Video File]. Retrieved from https://www.youtube.com/watch?v=c8BQqxL5wr4

McDonnell, M.D., Tissera, M. D., Tony, V., Andr, S., Jonathan, T. (2015, August). *Fast, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the 'Extreme Learning Machine' Algorithm*. https://doi.org/10.1371/journal.pone.0134254

Nielsan, A. M. (2015). *Neural network and Deep Learning*. Retrieved from http://neuralnetworksanddeeplearning.com/chap1.html

Tony, Y. (n.d.) *Understanding Neural Network.* Retrieved from https://towardsdatascience.com/understanding-neural-networks-19020b758230