



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE4020

Machine learning

**Hand-written character recognition using
supervised learning algorithms**

Project Report

By

Group Members:

Saksham (15BCE0279)

Shivang Raj (15BCE0691)

Gurrehmat Singh Oberoi (15BCE0927)

Submitted to:

Prof. Jaisakthi S M

Assistant Professor

School of Computer Science and Engineering

VIT UNIVERSITY

DECLARATION

We hereby declare that the project entitled “**Hand-written character recognition using supervised learning algorithms**” submitted by us to the School of Computer Science and Engineering, VIT University, Vellore-14 in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** is a record of bonafide work carried out by us under the supervision of **Jaisakthi S M, Assistant Professor**. I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree of this institute or of any other institute or university.

Signature

Saksham (15BCE0279)

Signature

Shivang Raj (15BCE0691)

Signature

Gurrehmat Singh Oberoi (15BCE0927)

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of direction and assistance from many people and we are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and we would not forget to thank them.

We respect and thank our Professor Mrs. Jaisakthi S.M., for providing us an opportunity to do the project work and giving us all support and guidance which helped us complete the project duly. We are extremely thankful to her for providing such nice support and guidance, although having busy schedule.

We also want to use this opportunity to thank our University for providing us with the platform to carry out our research and project work.

CONTENT

➤ Abstract.....	5
➤ Introduction.....	5
➤ Problem statement.....	6
➤ Methodology.....	6
➤ Dataset.....	9
➤ Tools used.....	9
➤ Code.....	9
➤ Output screenshots.....	18
➤ Result.....	20
➤ Conclusion.....	21
➤ Future scope.....	21
➤ References.....	21

Abstract

Handwritten character recognition system plays a very important role in today's world. Handwritten character recognition is very popular and computationally expensive work. At present time it is very difficult to find correct meaning of handwritten documents. There are many areas where we need to recognize the words, alphabets and digit. Handwriting recognition system can be used to solve many complex problems and can make human's work easy. One of the ways to do this is to train a model with many data of features of the image of handwritten characters to make the model recognise any random input to it. And here so we are going to use machine learning algorithms i.e. supervised algorithms like k-NN (K Nearest Neighbours) and SVM to train and test the model which would be able to identify the given handwritten character.

Introduction

We live in the 21st century, an era marked by rapid and unbounded growth in all spheres of life. As we attempt to convert all old hand-written documents into the digital format, it is important to convert images of characters into proper Unicode characters for further processing of that data. Optical Character Recognition is the process of mechanical/electronic conversion of any image of text, be it handwritten, typed or printed, into machine-encoded text. Handwriting recognition is an ability and technique of the system which receive the input from touch screen, electronic pen, scanner, images and paper documents. Handwriting recognition is very useful in real world. There are many practical problems where handwriting recognition system is very useful like documentation analysis, mailing address interpretation, bank check processing, signature verification, postal addresses. Human experts often solve difficult problems quickly and effortlessly by categorizing complex situations as special cases of familiar paradigms and applying solution strategies that are known to be effective for given paradigms.

In our project we focus on examining a system that induces general categorization rules within a supervised learning paradigm using two popular supervised learning algorithms are k – Nearest Neighbours and Support Vector Machines. A large number of unique examples of image are presented to the system along with the class i.e. basically the character which is there in the example image as training set to get trained. As here in our project for handwritten digit recognition we have provided a training set of size 5000 involves 10 classes i.e. 0-9 (500 for each).

Problem Statement

Recognising the digit on image having handwritten imprint of digit. Our aim is to be able to implement a handwriting recognition system in python using k-NN and SVM, machine learning algorithm. The training images' features are extracted using a 16-parameter image extraction algorithm in Matlab and also for testing set too. Then using k-NN and SVM we are going to train the raw dataset which would help us to predict the input character.

Methodology

Five thousand images of digits were assembled and while making dataset of extracted features it has been distributed randomly in uniform manner. Here the “on” pixels represent the image of desired character. These arrays of image averaged about 20 pixels high by 20 pixels wide.



Fig.: Training dataset images

Each character image was then scanned, pixel by pixel, to extract 16 numerical attributes. These attributes represent primitive statistical features of the pixel distribution. To achieve compactness, each attribute was then scaled linearly to a range of integer values from 0 to 15. This final set of values was adequate to provide a perfect separation of the 10 classes. That is, no feature vector mapped to more than one class. The attributes (before scaling to 0-15 range) are:

- i. The horizontal position, counting pixels from the left edge of the image, of the centre of the smallest rectangular box that can be drawn with all "on" pixels inside the box.
- ii. The vertical position, counting pixels from the bottom, of the above box.
- iii. The width, in pixels, of the box.
- iv. The height, in pixels, of the box.
- v. The total number of "on" pixels in the character image.
- vi. The mean horizontal position of all "on" pixels relative to the centre of the box and divided by the width of the box.
- vii. The mean vertical position of all "on" pixels relative to the centre of the box and divided by the height of the box.
- viii. The mean squared value of the horizontal pixel distances as measured in vi above.
- ix. The mean squared value of the vertical pixel distances as measured in vii above.
- x. The mean product of the horizontal and vertical distances for each "on" pixel as measured in vi and vii above. This attribute has a positive value for diagonal lines that run from bottom left to top right and a negative value for diagonal lines from top left to bottom right.
- xi. The mean value of the squared horizontal distance times the vertical distance for each "on" pixel. This measures the correlation of the horizontal variance with the vertical position.
- xii. The mean value of the squared vertical distance times the horizontal distance for each "on" pixel. This measures the correlation of the vertical variance with the horizontal position.
- xiii. The mean number of edges (an "on" pixel immediately to the right of either an "off" pixel or the image boundary) encountered when making systematic scans from left to right at all vertical positions within the box.
- xiv. The sum of the vertical positions of edges encountered as measured in 13 above.
- xv. The mean number of edges (an "on" pixel immediately above either an "off" pixel or the image boundary) encountered when making systematic scans of the image from bottom to top over all horizontal positions within the box.
- xvi. The sum of horizontal positions of edges encountered as measured in xv above.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	6	13	4	7	4	6	7	6	3	10	7	9	5	9	5	8					
2	4	2	5	4	4	8	7	6	6	7	6	6	2	8	7	10					
3	3	9	5	7	4	8	7	3	8	5	6	8	2	8	6	7					
4	3	9	5	7	4	8	7	3	8	5	6	8	2	8	6	7					
5	4	2	5	4	4	8	7	6	6	7	6	6	2	8	7	10					
6	2	8	3	5	1	8	13	0	6	6	10	8	0	8	0	8					
7	2	3	3	4	1	0	1	5	6	0	0	6	0	8	0	8					
8	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10					
9	4	11	5	8	3	8	8	6	9	5	6	6	0	8	9	7					
10	3	7	5	5	3	7	8	2	9	11	7	7	1	8	6	7					
11	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10					
12	3	7	5	5	3	7	8	2	9	11	7	7	1	8	6	7					
13	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10					
14	2	3	3	4	1	0	1	5	6	0	0	6	0	8	0	8					
15	2	3	3	4	1	0	1	5	6	0	0	6	0	8	0	8					
16	5	12	3	7	2	10	5	5	4	13	3	9	2	8	4	10					
17	6	13	4	7	4	6	7	6	3	10	7	9	5	9	5	8					
18	2	1	3	1	1	8	6	6	6	6	5	9	1	7	5	10					
19	2	3	3	4	1	0	1	5	6	0	0	6	0	8	0	8					
20	4	2	5	4	4	8	7	6	6	7	6	6	2	8	7	10					
21	4	2	5	4	4	8	7	6	6	7	6	6	2	8	7	10					

Fig.: Generated training dataset

Then, two classifiers have been used to make the model. First one is k-NN classifier that works based on a single nonpragmatic decision. Each input image to get classified or to get predicted is examined based on the distance of its features from the features of other images in training dataset. The distance here gets measured using Euclidean distance.

$$\text{dist}((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

K nearest neighbour algorithm uses K closest samples to the query image. Each of these samples belongs to a known class C_i . The query image I_q is categorized to the class C_M which has the majority of occurrences among the K samples. The performance of the kNN classifiers highly related to value of the k, the number of the samples and their topological distribution over the feature space.

The second classifier used in our project is SVM (Support Vector Machines) that are based on the concept of decision planes that define decision boundaries. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well.

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

Dataset

The dataset we will employ is the open source handwritten character database provided by OpenCV (Open Source Computer Vision Library) which has about 5000 handwritten digit samples (500 for each digit), and about 20000 handwritten alphabets. For digits we have got just image as dataset. So, from the given image we have created whole feature dataset using the above-mentioned methodology on our own.

Tools used

- Python
- Matlab
- NumPy
- OpenCV Library
- Matplotlib

Code

```
% gendataset.m

img = imread('0.png');

resize_img=imresize(img,[16,16]);

ri=resize_img;

%imshow(ri)

sum=0;

sum3=0;

sum31=0;

% average of the distance of first "on" pixel from left

for i=1:16

    cnt1=0;

    for j=1:16

        if ri(i,j)==0

            sum=sum+1;

            cnt1=cnt1+1;

        end

    end

end
```

```

        else if ri(i,j)==1

            sum3=sum3+cnt1-7;

            mean1(i)=cnt1;

            sum31=sum31 + sqrt(abs(cnt1^2-8^2));

            break;

        end

    end

end

end

end

sum=sum/32;

sum=floor(sum)+2;

A(1)=sum;

%disp(sum)

sum1=0;

sum4=0;

sum41=0;

% average of the distance of first "on" pixel from top

for i=1:16

    cnt2=0;

    for j=1:16

        if ri(j,i)==0

            sum1=sum1+1;

            cnt2=cnt2+1;

        else if ri(j,i)==1

            sum4=sum4+cnt2-8;

            mean2(i)=cnt2;

            sum41=sum41 + sqrt(abs(cnt2^2-8^2));

            break;

        end

    end

```

```

        end

    end

end

sum1=sum1+48;

sum1=sum1/16;

sum1=floor(sum1)+2;

A(2)=sum1;

%disp(sum1)

% box width

A(3)=4;

%

% box height

A(4)=7;

%The total number of "on" pixels in the character image.

sum2=0;

for i=1:16

    for j=1:16

        if ri(j,i)==1

            sum2=sum2+1;

        end

    end

end

end

%sum2=sum2;

A(5)=abs(sum2)*2;

%disp(sum2)

%The mean horizontal position of all "on" pixels relative to the center of
the box and

%divided by the width of the box

```

```

sum3=floor(abs(sum3)/(A(3)))+5;

A(6)=sum3;

%The mean vertical position of all "on" pixels relative to the center of the
box and

%divided by the width of the box

sum4=5+floor(abs(sum4)/(A(4)));

A(7)=sum4;

%The mean squared value of the horizontal pixel distances

sum31=floor(sum31/A(3))*2;

A(8)=abs(sum31);

%The mean squared value of the vertical pixel distances

sum41=1+floor(sum41/A(4));

A(9)=abs(sum41);


%The mean product of the horizontal and vertical distances for each "on"
pixel

sum5=0;

for i=1:11

    sum5= sum5 + (mean1(i) * mean2(i));

end

sum5=sum5+48;

sum5= floor(sum5/16)+6;

A(10)=sum5;

%The mean value of the squared horizontal distance times the vertical
distance for each

%"on" pixel.

sum6=0;

for i=1:11

    sum6= sum6 + sqrt(mean1(i)^2 * mean2(i));

end

```

```

sum6=7+floor(sum6/16)*3;

A(11)=sum6;

%The mean value of the squared vertical distance times the horizontal
distance for each

%"on" pixel.

sum7=0;

for i=1:11

    sum7= sum7 + sqrt(mean2(i)^2 * mean1(i));

end

sum7=(floor(sum7/16)+1)*9;

A(12)=sum7;

%The mean number of edges encountered when making systematic scans from
%left to right at all vertical positions within the box

sum8=0;

for i=1:16

    for j=1:8

        if ri(i,j)==0

            sum8=sum8+1;

        end

    end

end

sum8=8-floor((sum8/16));

A(13)=sum8;

%The sum of the vertical positions of edges encountered as measured

sum9=0;

for i=1:16

    for j=9:16

        if ri(i,j)==0

            sum9=sum9+1;

        end

    end

end

```

```

        end

    end

end

sum9=5+floor(sum9/16);

A(14)=sum9;

%The mean number of edges encountered when making systematic scans of the
image from

%bottom to top over all horizontal positions within the box

sum10=0;

for i=1:8

    for j=1:16

        if ri(i,j)==0

            sum10=sum10+1;

        end

    end

end

sum10=floor(sum10/16);

A(15)=sum10;

%The sum of horizontal positions of edges encountered as measured

sum11=0;

for i=9:16

    for j=1:16

        if ri(i,j)==0

            sum11=sum11+1;

        end

    end

end

sum11=4+floor(sum11/16);

A(16)=sum11;

```

```

%disp(A);

for i=1:16

    fprintf('%d ',A(i));

    if(i<16)

        fprintf(';');

    end

end

end


#knnscr.py

from sklearn.neighbors import KNeighborsClassifier

import pandas as pd

import numpy as np

from Tkinter import Tk

from tkFileDialog import askopenfilename

from sklearn import metrics


data = pd.read_csv('digits2.csv', header=None)

targ = pd.read_csv('digitstarget3.csv', header=None)

test = pd.read_csv('test.csv', header=None)

clf=KNeighborsClassifier(n_neighbors=3)

train_data= data[:3000]

train_targ= targ[:3000]

train_targ= train_targ.values

test_data= data[3000:]

test_targ= targ[3000:]

test_targ= test_targ.values


print 'Training kNN algorithm'

```

```

clf.fit(train_data,train_targ)

print 'Training complete'


print 'Testing algorithm'

result = clf.predict(test_data)

print type(test_targ)

#print result


correct=0

for i in range(result.size):

    if (result[i]==(test_targ[i])):

        correct+=1

accuracy = correct*100.0/result.size

print 'Testing complete, Accuracy = ', float(accuracy),'%'

print(metrics.classification_report(test_targ, result))

print(metrics.confusion_matrix(test_targ, result))

#print test

#i=input('Enter row number from above to test: ')

retest=1

while(retest):

    #print '\n\nDEMO'

    Tk().withdraw()

    filename = askopenfilename()

    i = int(filename[-5])

    result = clf.predict(test.iloc[[i]])

    print ('The selected image is predicted to be ')

    print result[0]

    retest= input('Test another image?(1/0):')

```



```

#svmscr.py

from sklearn import svm

import pandas as pd

import numpy as np

from Tkinter import Tk

from tkFileDialog import askopenfilename

from sklearn import metrics


data = pd.read_csv('digits2.csv', header=None)
targ = pd.read_csv('digitstarget3.csv', header=None)
test = pd.read_csv('test.csv', header=None)

clf=svm.SVC(kernel='linear')

train_data= data[:3000]
train_targ= targ[:3000]
train_targ= train_targ.values
test_data= data[3000:]
test_targ= targ[3000:]
test_targ= test_targ.values


print 'Training SVM algorithm'

clf.fit(train_data,train_targ)

print 'Training complete'


print 'Testing algorithm'

result = clf.predict(test_data)

print type(test_targ)

correct=0

for i in range(result.size):

    if (result[i]==(test_targ[i])):

```

```

        correct+=1

accuracy = correct*100.0/result.size

print 'Testing complete, Accuracy = ', float(accuracy), '%'

print(metrics.classification_report(test_targ, result))

print(metrics.confusion_matrix(test_targ, result))

#print test

#i=input('Enter row number from above to test: ')

retest=1

while(retest):

    #print '\n\nDEMO'

    Tk().withdraw()

    filename = askopenfilename()

    i = int(filename[-5])

    result = clf.predict(test.iloc[[i]])

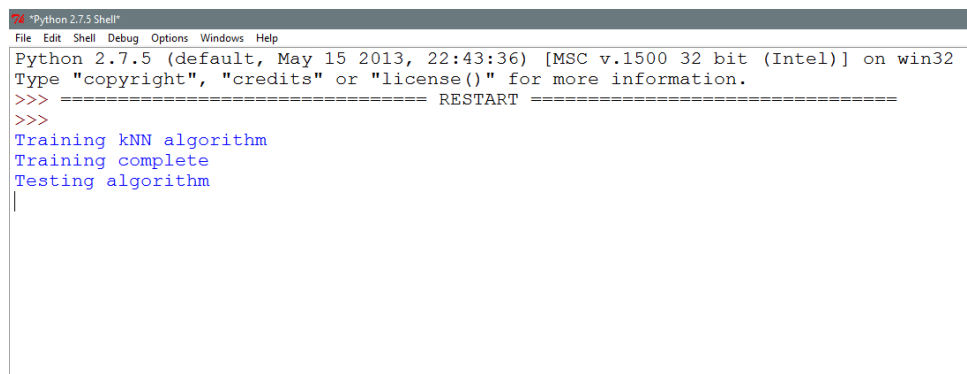
    print ('The selected image is predicted to be ')

    print result[0]

    retest= input('Test another image?(1/0):')

```

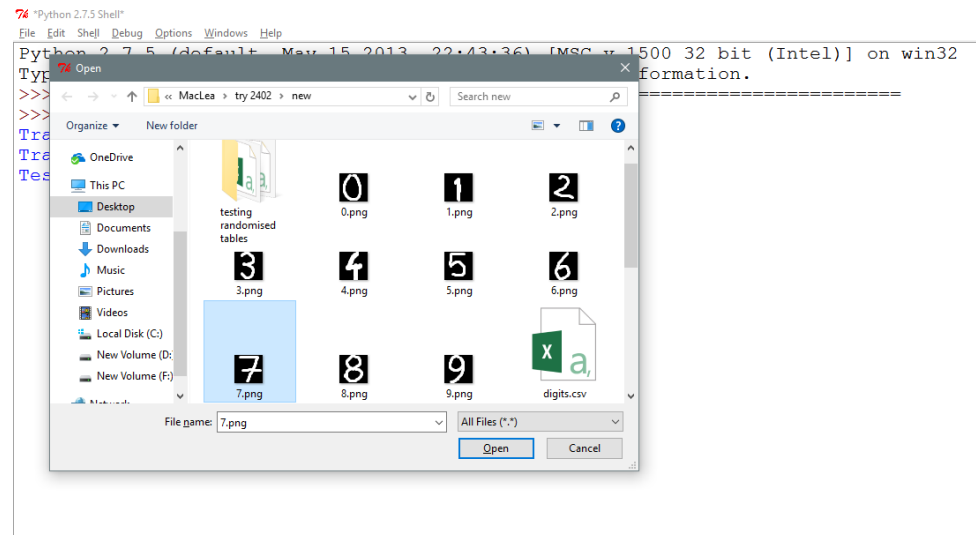
Output Screenshots



```

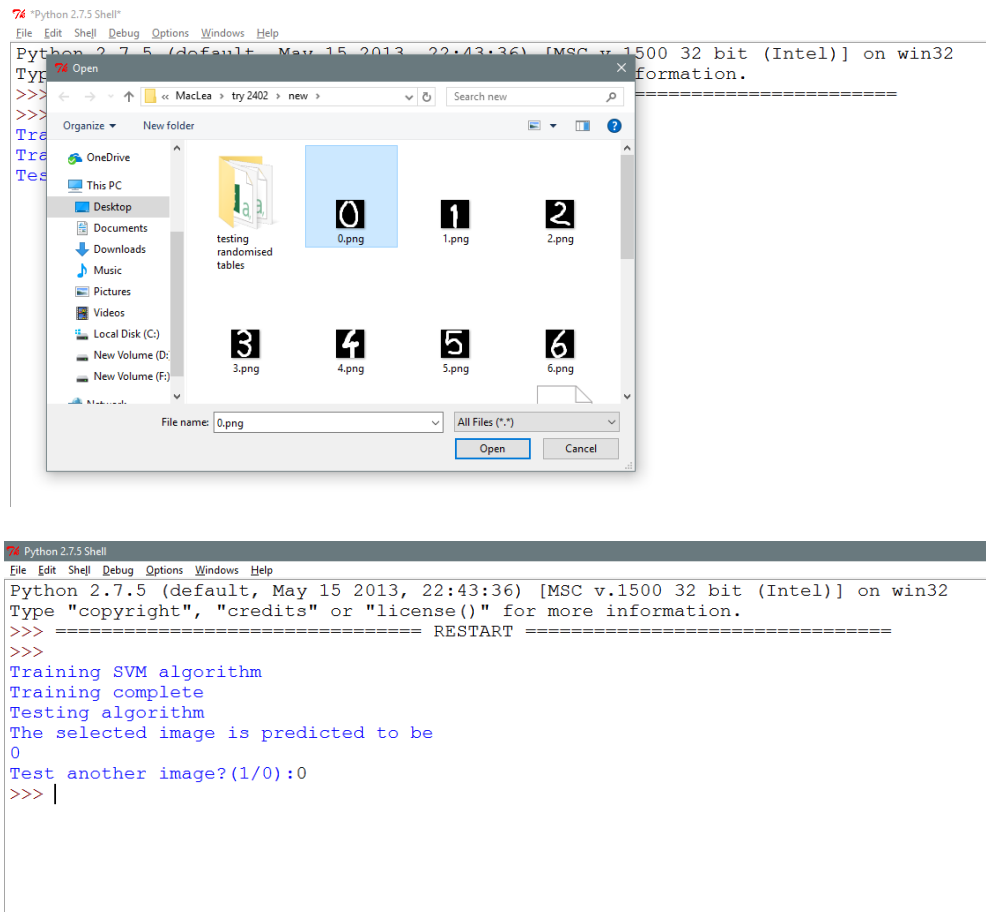
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Training kNN algorithm
Training complete
Testing algorithm

```



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Training kNN algorithm
Training complete
Testing algorithm
The selected image is predicted to be
7
Test another image?(1/0):0
>>>
```

```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Training SVM algorithm
Training complete
Testing algorithm
```



Result

kNN Accuracy = 96.0981047938%

class	precision	recall	f1-score	support
0	0.99	1	0.99	88
1	0.96	0.97	0.96	91
2	0.95	0.97	0.96	86
3	0.91	0.96	0.93	91
4	0.99	0.95	0.97	92
5	0.97	0.97	0.97	91
6	0.99	1	0.99	91
7	0.98	1	0.99	89
8	0.94	0.91	0.92	86
9	0.94	0.9	0.92	92
avg/total	0.96	0.96	0.96	897

Table: Knn Result

SVM Accuracy = 94.2028985507%

class	precision	recall	f1-score	support
0	0.97	0.99	0.98	88
1	0.94	0.9	0.92	91
2	1	0.99	0.99	86
3	0.96	0.86	0.91	91
4	0.99	0.95	0.97	92
5	0.89	0.97	0.93	91
6	0.98	0.99	0.98	91
7	0.97	0.96	0.96	89
8	0.88	0.9	0.89	86
9	0.87	0.93	0.9	92
avg/total	0.94	0.94	0.94	897

Table: SVM Result

Conclusion

We have extracted the feature of images to form the dataset using a newly compounded dynamic. Based on the results, that we have tested, the model that we have proposed is working successfully using both the classifiers i.e. k-NN and SVM. The current step is on a small scale as it is purely for demonstrative purpose. The training dataset could be further modified with more number of data to make it better to use in real world.

Future Scope

Handwritten character recognition has several applications in the world around us. Having successfully implemented the basics of this system, we can go for model it for more complex conditions to make the system recognise characters. Like, as we have just taken images of individual character to get it recognised, in future we can go big on it and ask for a whole word or sentence to get recognised using the basis of the system we have created.

References

- Peter W Frey, David J Slate, Letter recognition using Holland-style adaptive classifiers (1991), Kluwer Academic Publishers, Boston.

- Guo G., Wang H., Bell D., Bi Y., Greer K. (2003) KNN Model-Based Approach in Classification. In: Meersman R., Tari Z., Schmidt D.C. (eds) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Lecture Notes in Computer Science, vol 2888. Springer, Berlin, Heidelberg
- Hirali S. Amrutiya, Payal P. Moghariya, Vatsal H. Shah (2014) Handwritten Character Recognition. In: International Journal of Advance Engineering and Research Development (IJAERD) Special Issue, Volume 1, Issue 4, April 2014.
- Durgesh K. Srivastava, Lekha Bhambhu (2010) Data Classification Using Support Vector Machine. In: Journal of Theoretical and Applied Information Technology 2010.
- Machine Learning - OpenCV-Python Tutorials http://www.opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/
- Support Vector Machines, http://scikitlearn.org/stable/modules/neural_networks_supervised.html
- K-Nearest Neighbor, <https://gurus.pyimagesearch.com/lesson-samplek-nearest-neighbor-classification>