PROJECT REPORT

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

BCA III

BY:

Jitendra Sai (177404)
Akhil Sai (174405)

OPTICAL CHARACTER RECOGNITION

Optical Character Recognition

For single Alphabets and Numbers

A Project as a Course requirement for

Bachelor of Computer Application

Akhil Sai(174405)

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

Department of Mathematics and Computer Science
Muddenahalli Campus
March 2020

SRI SATHYA SAI INSTITUTE OF HIGHER LEARNING
(Deemed to be University)

CERTIFICATE

This is to certify that this Project titled **Optical Character Recognition (For single Alphabets or Numbers)** submitted by Akhil Sai.G, Department of Mathematics and Computer Science, Muddenahalli Campus is a bonafide record of the original work done under my/our supervision as a Course requirement for the Degree of Bachelor of Computer Applications.

…….
Sri Udhaya Ravishankar,
Project Supervisor

Countersigned by

Place: Muddenahalli

……..
Sri B. Venkatramana,
Deputy Director of the campus

Date: 20 March, 2020

DECLARATION

The Project titled **Optical Character Recognition (For single Alphabets or Numbers)** was carried out by me under the supervision of Sri B. Venkatramana, Department of Mathematics and Computer Science, Muddenahalli Campus as a Course requirement for the Degree of Integrated Master of Computer Applications and has not formed the basis for the award of any degree, diploma or any other such title by this or any other University.

.....

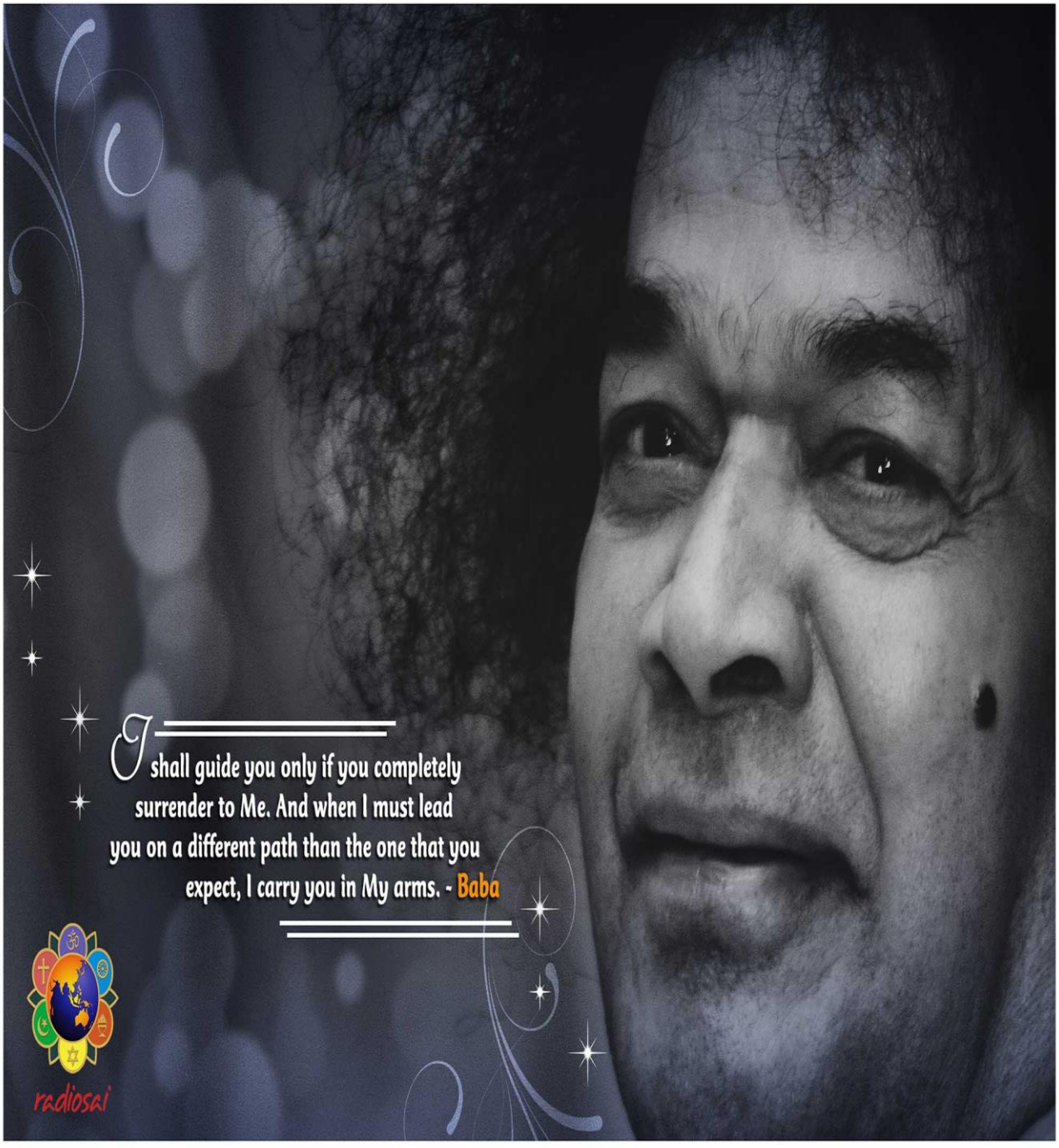
Akhil Sai. G

Place: Muddenahalli

174405.

Date: 20th March, 2020

Integrated Master of Computer Applications
Muddenahalli Campus



Dedicated to Bhagawan Sri Sathya Sai Baba

ACKNOWLEDGEMENTS

First and foremost, I thank and express my heartfelt gratitude to **Bhagawan Sri Sathya Sai Baba,** who have blessed and guided me in my every endeavour.

I am thankful to my parents for their deepest love and supporting us throughout the journey.

I would like to thank the Director B. Venkatramana and Warden *Sri Sai Manohar* sir.

I specially thank Udhaya ravishankar Sir, for giving me this project and constant support and his motivation.

I sincerely thank all the teachers for supporting and guiding me in developing this project.

Without their support this project wouldn't have come to this shape.

I am greatly thankful to *my friends* for their encouragement and providing insightful feedbacks and also our seniors for answering my queries and providing valuable inputs.

I also thank the *Stack Overflow, google, GitHub* community for answering my queries.

I thank all who've helped me directly or indirectly. I beg pardon if I've missed out any names.

Special Thanks

To

My Mentor

Sri Udhaya Ravishankar
Associate Professor
Department of Mathematics and Computer Science
Sri Sathya Sai Institute Higher Learning
Prashanti Nilayam Campus

ABSTRACT

This project is regarding optical character recognition which works for a single Alphabet or Number. The Graphical User Interface provides the interface to draw the image and it will be given to algorithm which process the image and gives the predicted output in three confidence guesses. We have worked on it to know how this recognition works, and we tried to implement and write our own algorithm at most. Some basic libraries that are available in have been used in this project. We also used some Computer vision techniques which will be helpful to understand our project and move on it. This project has been divided into two modules one is the backend which runs the algorithms which gives the output and the other is the Interface which connects the algorithm, getting the image and displaying the output.

Why Python:

- ✓ Python has the simplicity in its usage.
- ✓ Python has the libraries which are efficient for the usage and which is useful for our project such as numpy, matplotlib, scipy and scikit learn.
- ✓ Python has the efficient and inbuilt data types such as lists and dictionaries which are most useful for our project.

- ✓ We are doing it by not using any advanced algorithms such as Neural networks.
- ✓ We have used basic lists, dictionaries, arrays etc. and some computer vision concepts.

NOTE:

- ➢ This project has been divided into two modules which will be dealing with the backend and front end.
- ➢ Front end Part has been taken by Jitendra Sai .G (174404)
- ➢ Backend part has been taken by Akhil Sai.G(174405)
- ➢ This report consists of the dealings with backend (Akhil Sai.G(174405)) and some common modules between both.

Contents:

1.Introduction

2.Image Source

3.Deskewing

4.Character Recognising Algorithm

-Getting the points from an image.

-Clustering of points in an Image (Land marks)

-Finding Shortest Path

-Representing Images

-Working with Dictionaries and Pickle files

5.Testing images

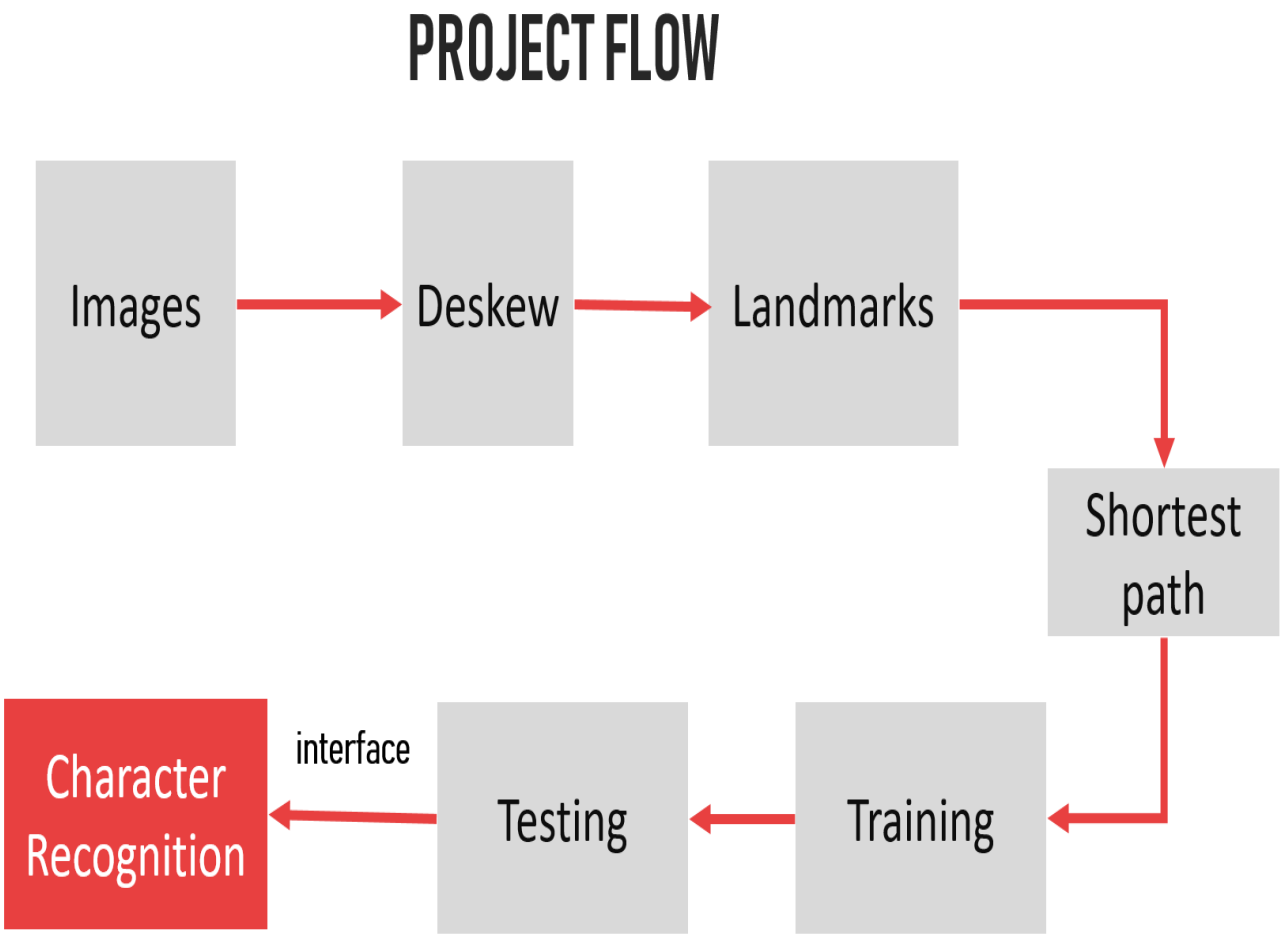
6.Graphical User Interface

7.Pros and Cons

8.Conclusion

Introduction:

- ✓ This Project is about recognizing the characters such as alphabets and number.
- ✓ The Interface has been used to draw the image and the image will be taken as input for the test.
- ✓ The algorithm runs behind the interface that results output at three confidence levels which will be given for output.
- ✓ It consists of basic python coding such as lists, dictionaries, arrays and matrix.
- ✓ For the backend we used libraries such as math, copy, numpy, matplotlib, scipy, scikit-learn, PIL and pickle.
- ✓ For Frontend we used libraries such as tkinter for Graphics.
- ✓ In this report number images have been given for better understanding.


PROJECT FLOW
Images
Deskew
Landmarks
Shortest path
Training
Testing
interface
Character Recognition

1.Image Source

- ✓ Number images are taken from the MNIST Dataset as ".gz" files.
- ✓ After getting these images downloaded, MNIST images should be kept in '/tmp' directory.
- ✓ Alphabets images are taken from the Alphabet Dataset in csv format.
- ✓ These datasets are available in the Website "https://kaggle.com/"
- ✓ And for the Alphabet images we have to convert that csv format images into '.png' format by writing program for that.
- ✓ Alphabet images are further modified and kept as folder and images has been accessed from that folder.

These are the libraries used throughout the project.

```
#import mnist
import os
import math
import copy
import string
import pickle as p
import numpy as np
import matplotlib.pyplot as graph
import matplotlib.image as mpimg
import scipy as sp
from scipy.spatial.distance import cdist
from scipy.spatial import distance
from sklearn.cluster import KMeans
from PIL import Image
##########     MY LIBRARIES ########
import MyClust            #MyClust consists of clustering algorthims
import Euclid             #EUCIDIAN-DISTANCES
import Deskew             #DESKEWING
import Hamilton as Ham    #Shortest path with less cost
#########################################

```

Python libraries:

Os in this project is used for the file operations.
String is used for the string operations.
Pickle files are used for storage purpose.
Numpy is used for the array manipulations.
Matplotlib is used for plotting and image retrievals.
Scipy is used for Deskewing of image and distance calculations.
Sklearn.cluster is used for clustering specifically K-means.
PIL for retrieval and manipulation of images.

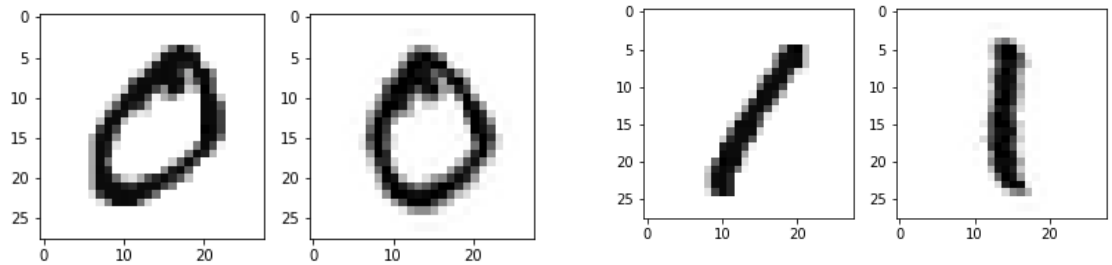
Own libraries:

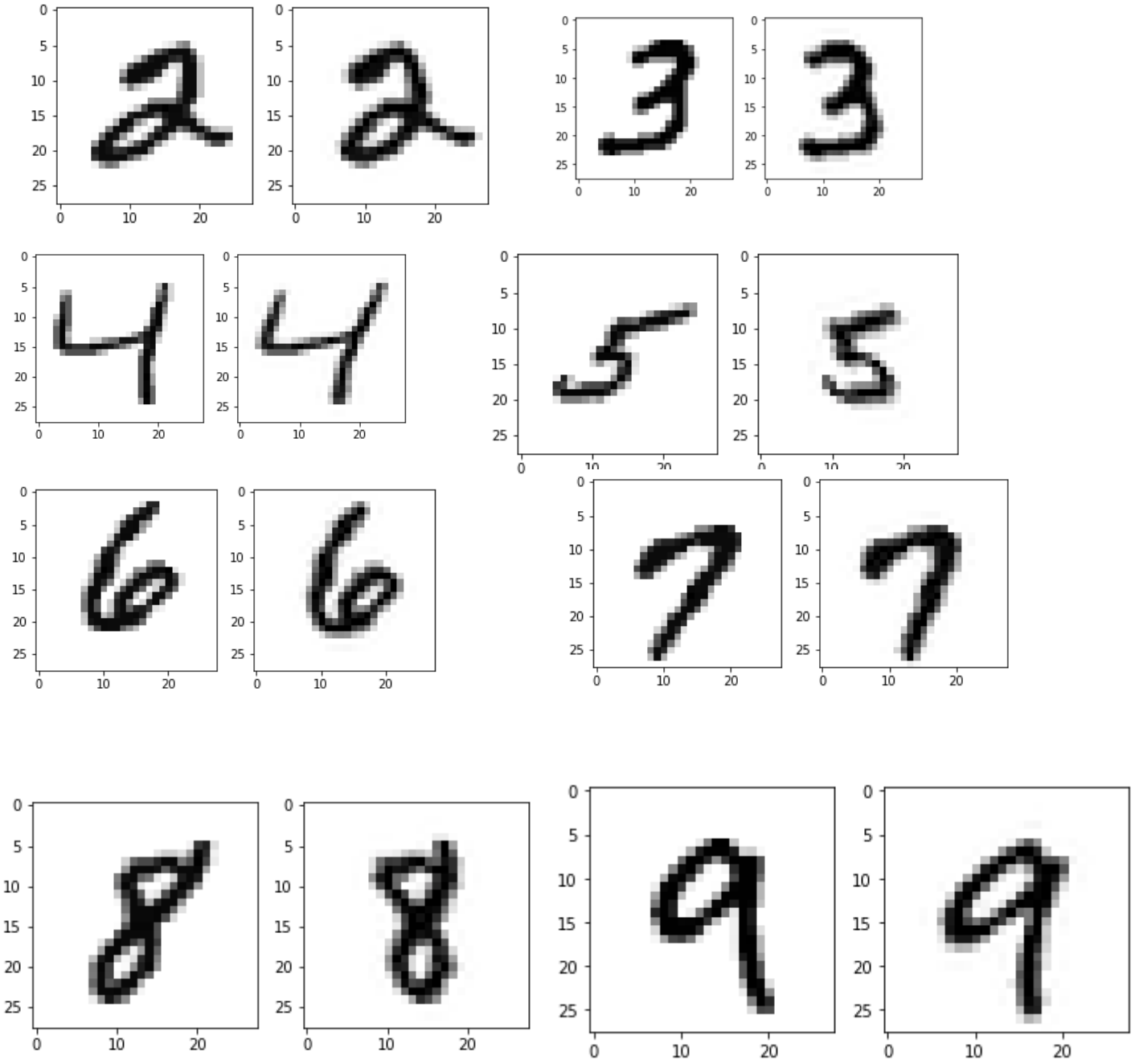
MyClust - This is used for clustering purpose.
Euclid - It is used for getting Euclidian distance and some functions.
Deskew - It is used for Deskewing purpose.
Hamilton - It is used for getting the shortest path.

2.Deskewing:

- ✓ When we write, we often write at angles to the paper, which cause letters and numbers to be skewed. Unfortunately, unlike the human eye, computers cannot easily find similarities between images that are transformations of each other. Thus, the process of deskewing.
- ✓ Very formally, deskewing is the process of straightening an image that has been scanned or written crookedly — that is an image that is slanting too far in one direction, or one that is misaligned.
- ✓ Deskewing part has been taken from the internet source which had a little advanced coding needed for that.
- ✓ *What Will It Do?*
- ✓ Deskewing involves the changes to keep the slanting image into correct position by varying its angles.
- ✓ It is used to make the prediction more appropriate.
- ✓ These are functions used for Deskewing of the image.

Images after Deskewing:





Here we can see the difference between original and deskewed image.

We can see the rapid changes taken place in the appearance of the number.

3.Character Recognising Algorithm

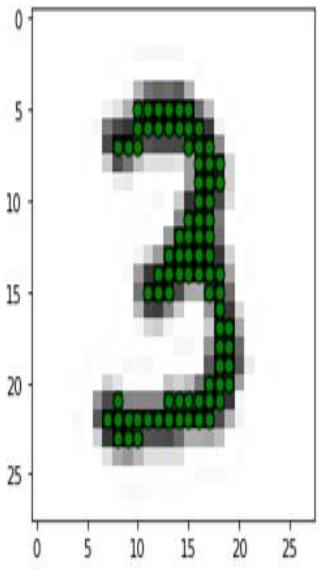
Getting the points from an image:

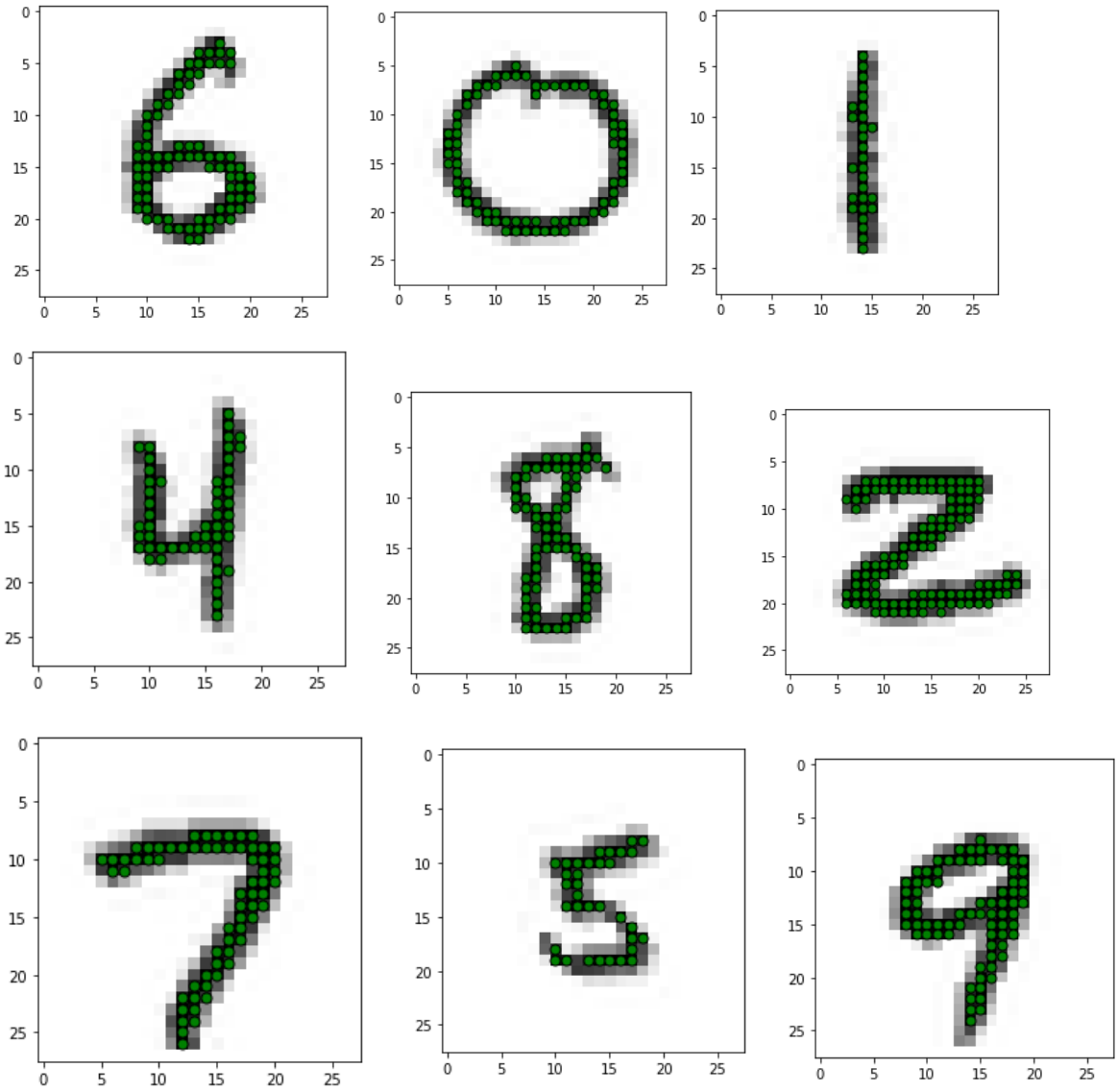
- ✓ First we need to get points from the image to get land marks.
- ✓ The code of K-means clustering on images threshold values will be used for getting the points from the image.
- ✓ K-means is taken from the library scikit-learn.

These points are used for getting the landmarks on the image

--- Points from the image ---

```
[[ 5 10]
 [ 5 11]
 [ 5 12]
 [ 5 13]
 [ 5 14]
 [ 5 15]
 [ 6 10]
 [ 6 11]
 [ 6 12]
 [ 6 13]
 [ 6 14]
  . . .
   .
   .
   .
```





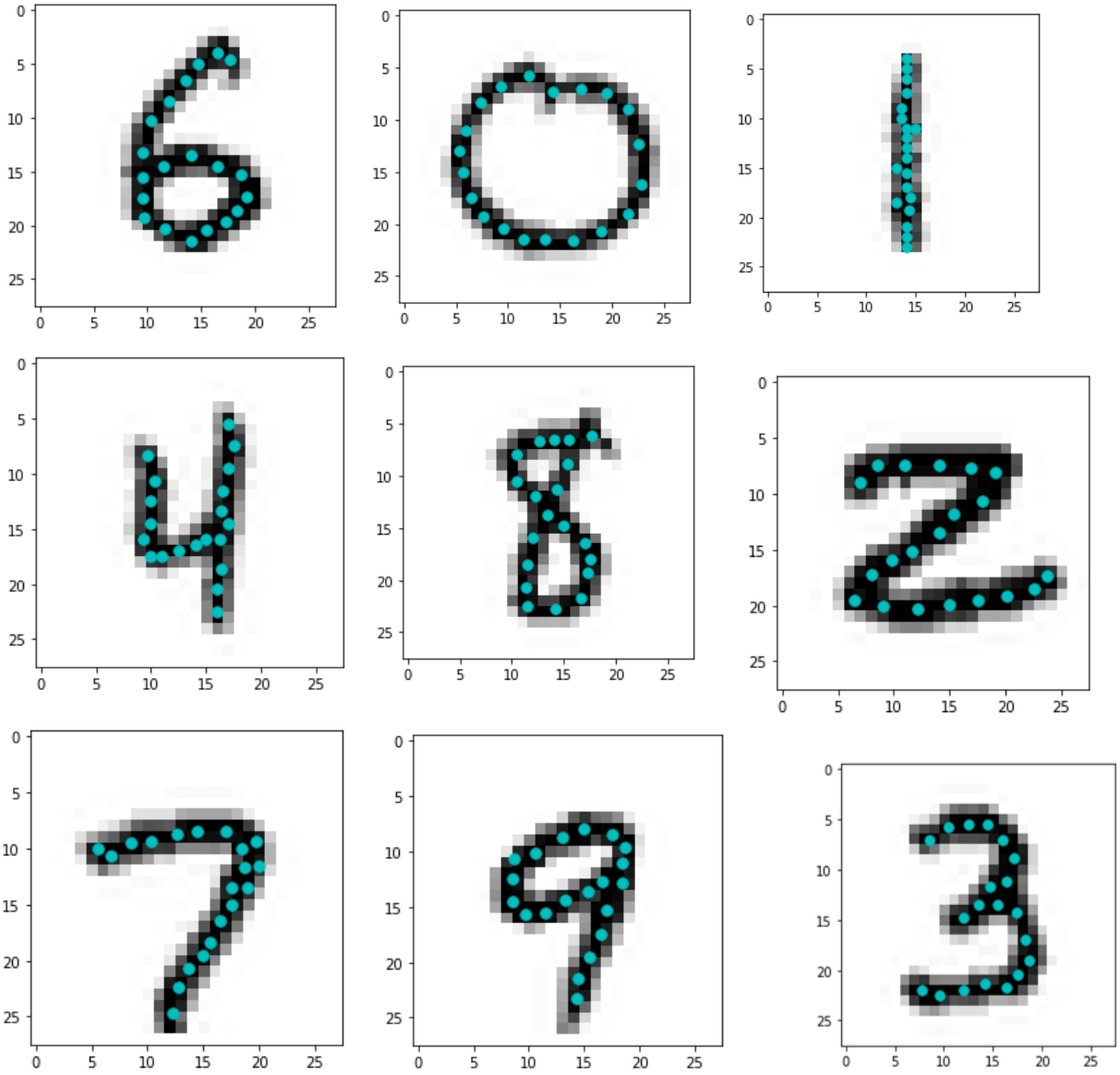
We can see points on different images of different numbers this can be applicable for alphabets as well.

Clustering of points in an Image (Land marks)

- ✓ Firstly, after getting the image we get some points out of that image by using the threshold value as we took if threshold is greater than 127, we will choose that points.
- ✓ These points are shown as green dots on the image using matplotlib.
- ✓ Now we will use the K-means clustering technique to choose some points on the image to represent it.
- ✓ After applying the technique, this points are plotted on the image as sky blue dots.
- ✓ Thus Landmarks are chosen for the image.

```
from sklearn.cluster import KMeans
num_clusters = 20

km = KMeans(n_clusters = num_clusters,n_init = 10,max_iter=300, tol = 1e-04,random_state = 0)
km= km.fit(data)  # It fits the data given data and gives required points
lm= km.cluster_centers_
print("Landmarks")
```



----------Cluster centres----------

```
[[14.2          17.4         ]
 [21.4          14.2         ]
 [ 5.5          14.5         ]
 [22.            7.75        ]
 [20.5          17.5         ]
 [ 8.83333333 17.16666667]
 [13.5          13.5         ]
 [17.           18.25        ]
 [ 5.8          10.4         ]
 [11.66666667 14.66666667]
 [14.75         12.          ]
 [22.5           9.5         ]
 [ 7.            8.5         ]
 [11.2          16.4         ]
 [19.           18.75        ]
 [13.5          15.5         ]
 [ 7.           16.          ]
 [21.66666667 16.33333333]
 [22.           12.          ]
[ 5.5          12.5         ]]
```

By getting these clusters we need to arrange these in an order to get the shape of the image.
The logic of getting the appropriate order leads to the next step in the algorithm

Finding Shortest Path:

- ✓ THE KEY IDEA is "If we join those points appropriately we can represent the according number.
- ✓ And the question is how to join those points appropriately?"
- ✓ One of the solution is, "If we join them, one point will be the starting and the other will be the end point. And again how to decide which is starting and which is the end point".
- ✓ For that we need to create list of all distance from one point to every other point for the landmarks.
- ✓ If we plot the landmarks and draw the image by joining them, the intuition is obvious that shortest path gives the respective image.
- ✓ The paths which is shortest with respect to their starting points will be taken and plotting each of them gives varied results.
- ✓ Among them resulting minimum distance which applied to the set of landmarks results similarity with the original image. Therefore, that shortest distance might predicts approximate probability of the number

For Example:

Suppose we have the 4 points, and their distances from one point to the other.

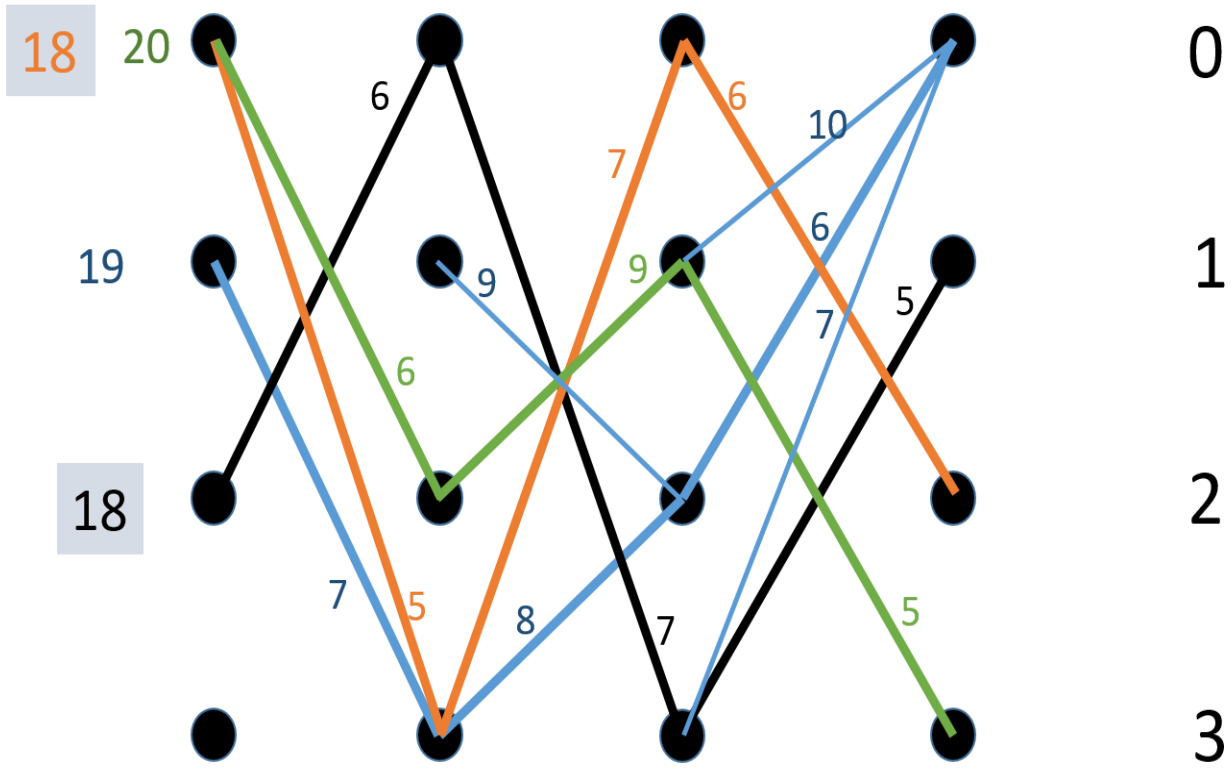
Points	0	1	2	3
0	0	10	6	7
1	10	0	9	5
2	6	9	0	8
3	7	5	8	0

Rules:

- ✓ We have to travel through all the nodes only once.
- ✓ We have to choose smallest distance from the starting point to the another point, except to itself.
- ✓ We have to find shortest path among all the paths obtained.
- ✓ Thus we can get the shortest path for the requirement.

Thus we can join the land marks efficiently.

The below figure shows the traversal through all the nodes only once.



List = [0 2 3 1] : 6+8+5=19

List = [[0 2 3 1] [1 3 0 2] [2 0 3 1] [3 1 2 0]]

Sum of the distances: 5+7+6=*18* --> [1 3 0 2]

6+7+5=*18* --> [2 0 3 1]

5+9+6=20 --> [3 1 2 0]]

We will take shortest distances among the various paths obtained.

We need to write functions to give the calculations for the shortest path.

```
List of Distances

[[ 0.          7.87908624  9.17060521 12.40816264  6.3007936   5.37173674
   3.96232255  2.92617498 10.93434955  3.72677996  5.42793699 11.45862121
  11.44770719  3.16227766  4.98623104  2.02484567  7.33484833  7.54247233
   9.48683298  9.98498873]
 [ 7.87908624  0.         15.90282994  6.47784686  3.42052628 12.91209597
   7.93095202  5.98017558 16.05615147  9.74451413  7.00446286  4.82700735
  15.4870914  10.43455797  5.14417146  8.00624756 14.51206395  2.1499354
   2.28035085 15.99062225]
 [ 9.17060521 15.90282994  0.         17.82729649 15.29705854  4.26874949
   8.06225775 12.0959704   4.11096096  6.16891851  9.58188395 17.72004515
   6.18465844  6.00832755 14.15317985  8.06225775  2.12132034 16.27028648
  16.68831927  2.        ]
 [12.40816264  6.47784686 17.82729649  0.          9.86470983 16.18748659
  10.26218788 11.62970335 16.41531297 12.43455088  8.40386816  1.82002747
  15.0187383  13.83699751 11.40175425 11.50271707 17.11906832  8.58980339
   4.25       17.17010483]
 [ 6.3007936   3.42052628 15.29705854  9.86470983  0.         11.6714276
   8.06225775  3.57945527 16.32482772  9.27661337  7.95691523  8.24621125
  16.22498074  9.36482781  1.95256242  7.28010989 13.58307771  1.64991582
   5.70087713 15.8113883 ]
```

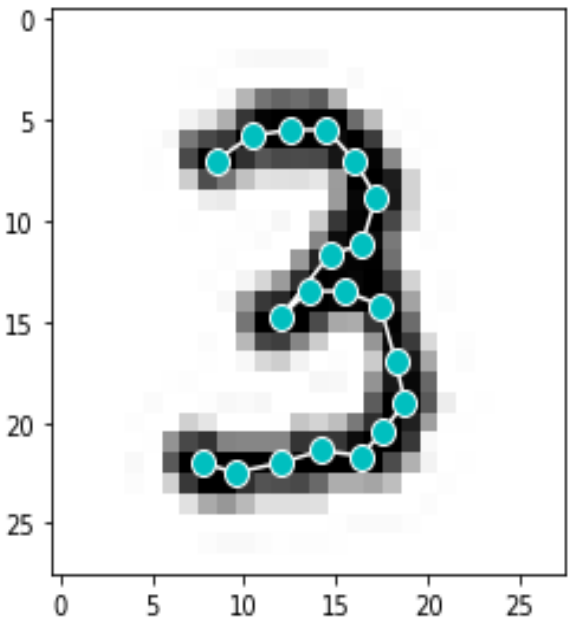
This is how landmarks will be stored in an array

Shortest Path: [[3, 11, 18, 1, 17, 4, 14, 7, 0, 15, 6, 10, 9, 13, 5, 16, 2, 19, 8, 12, 42.38969388976693]]

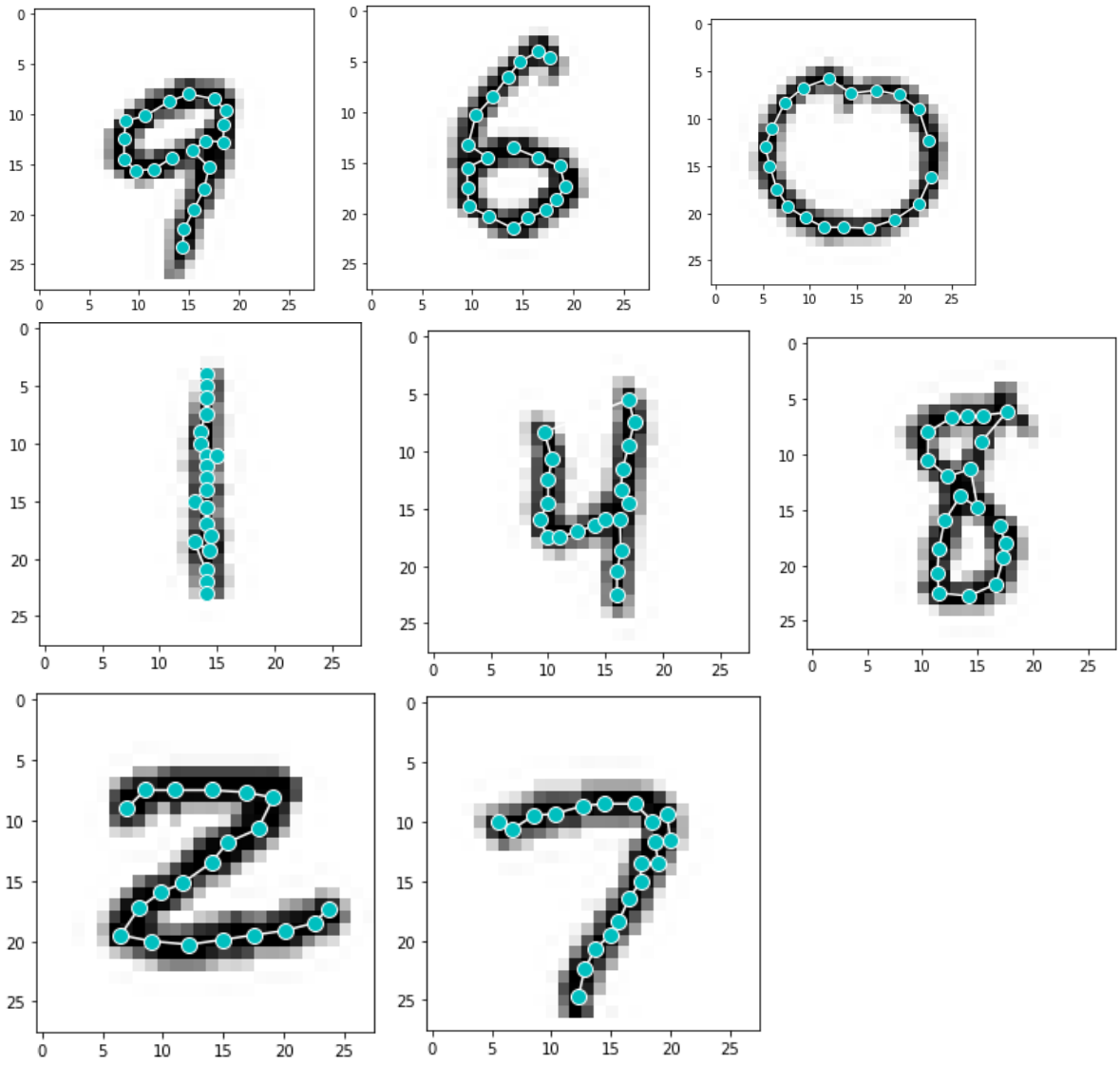
This code gives the ordered clusters.

Shortest path in the order:

```
[[22.          7.75      ]
 [22.5         9.5       ]
 [22.         12.        ]
 [21.4        14.2       ]
 [21.66666667 16.33333333]
 [20.5        17.5       ]
 [19.         18.75      ]
 [17.         18.25      ]
 [14.2        17.4       ]
 [13.5        15.5       ]
 [13.5        13.5       ]
 [14.75       12.        ]
 [11.66666667 14.66666667]
 [11.2        16.4       ]
 [ 8.83333333 17.16666667]
 [ 7.         16.        ]
 [ 5.5        14.5       ]
 [ 5.5        12.5       ]
 [ 5.8        10.4       ]
 [ 7.          8.5       ]]
```



This figures represents the path that clusters taken by joining the dots.



Now the crux part of getting the shortest path is completed. Now we need to generalize it for all the training images.

Working with Dictionaries and Pickle files:

- ✓ From here generalisation starts.
- ✓ Dictionaries are used for getting the output in some sorted manner.
- ✓ These are used for training purpose.
- ✓ Pickle files are used for storing those values.
- ✓ These are the same functions used in the basic one and here modified a little.

```
# Storing all the distances in the one pickle file

import pickle as p
'''with open('c20.p','wb') as handle:
    p.dump(d,handle)'''
with open('c20.p','rb') as handle:
    d= p.load(handle)
```

usage of Pickle files

d= {i:comp(i) for i in range(0,10)}

- ✓ This is the line which is used for storing all the training examples as their shortest path as per their number in an array respectively in the list.
- ✓ And then we will store output ‘d’ in a pickle file for consuming the time if it is to use again.

This is format how it will store the data.

```
# Printing all the images shortest distances of the training distances
print(d)
```

```
{0: [array([ 8.875     , 14.125     ,  6.125     , 14.875     ,  6.        ,
       12.28571429,  8.16666667, 11.16666667, 10.        , 10.25      ,
       11.6       ,  8.8       , 13.8       ,  8.6       , 15.33333333,
        7.66666667, 18.        ,  8.5       , 20.83333333,  9.83333333,
       22.125     , 12.125     , 22.2       , 14.4       , 21.4       ,
       16.2       , 20.        , 18.        , 18.16666667, 19.83333333,
       15.71428571, 21.        , 13.75      , 21.        , 11.83333333,
       19.83333333,  9.2       , 18.6       ,  7.66666667, 17.33333333]), array([ 7.125     , 13.875     ,  4.83333333,
14.83333333,  6.5       ,
       17.        ,  8.4       , 18.2       , 10.71428571, 19.        ,
       13.5       , 20.5       , 16.8       , 21.6       , 18.2       ,
       20.4       , 20.5       , 18.5       , 21.91666667, 15.41666667,
       21.71428571, 12.14285714, 20.        ,  9.5       , 18.        ,
        8.        , 16.        ,  7.25      , 13.        ,  7.71428571,
       11.16666667,  9.83333333, 10.        ,  8.5       ,  8.5       ,
       11.125     ,  9.83333333, 13.16666667, 10.75      , 15.        ]), array([10.        , 10.        ,  9.5       ,
12.5       , 11.71428571,
       12.        , 13.        ,  9.83333333, 15.        , 11.28571429,
       16.71428571,  9.42857143, 17.8       , 11.6       , 20.25      ,
```

Now clustering will be done for all the train images from the data obtained i.e. ‘d’.

```
# Storing that number representing clusters into the pickle file

'''with open('c20d25.p','wb') as handle:
    p.dump(dict_num,handle)'''
with open('c20d25.p','rb') as handle:
    dict_num= p.load(handle)
#print(dict_num[0][1])
print("\n\n-----Clusters are loaded-----\n\n")
```

```
-----Clusters are loaded-----
```

Thus clustering of images will be done and ‘dict_num’.

```
{0: array([[16.13333333, 22.        , 13.16666667, 21.5       , 10.6       ,
        20.33333333,  7.93055556, 19.05555556,  8.3       , 16.98333333,
         6.19642857, 15.86309524,  6.0452381 , 13.81428571,  5.7962963 ,
        11.85185185,  7.33846154, 11.28498168,  9.43055556,  9.90277778,
        11.58253968,  8.97407407, 13.12289562,  7.51010101, 16.17857143,
         7.08333333, 19.2547619 ,  7.475     , 19.98677249,  9.12698413,
        21.97407407, 11.34444444, 21.8219697 , 14.18939394, 22.01984127,
        16.8452381 , 20.27777778, 18.79365079, 19.225     , 20.9452381 ],
       [ 4.        , 17.        ,  5.        , 14.5       ,  5.5       ,
        12.5       ,  7.        , 11.25      ,  9.8       , 10.6       ,
        12.        , 10.        , 14.5       , 10.        , 16.5       ,
        10.5       , 18.5       , 10.5       , 20.5       , 11.        ,
        20.66666667, 12.33333333, 21.75      , 14.        , 20.5       ,
        16.        , 18.5       , 17.5       , 17.        , 18.5       ,
        15.5       , 18.5       , 13.5       , 18.5       , 11.5       ,
        18.5       ,  9.66666667, 18.33333333,  7.5       , 18.        ],
       [12.31746032,  8.04563492, 11.20400433,  8.64393939,  8.63383838,
         8.97853535,  6.77579365, 10.27083333,  6.70267857, 12.35089286,
         6.69191919, 14.68686869,  7.525     , 16.6       ,  8.97142857,
```

Thus clusters can be loaded into pickle file.

What Clusters do?

- ✓ These clusters are used for the recognition. Each number have the points of similar dimension say (40,1).
- ✓ By calculating the shortest distance of the given image with the all points (40,1) in the dictionary.
- ✓ The nearest distance will give the first guess by telling which cluster it belong consequently the other guesses as well.
- ✓ The distances give the confidence level.

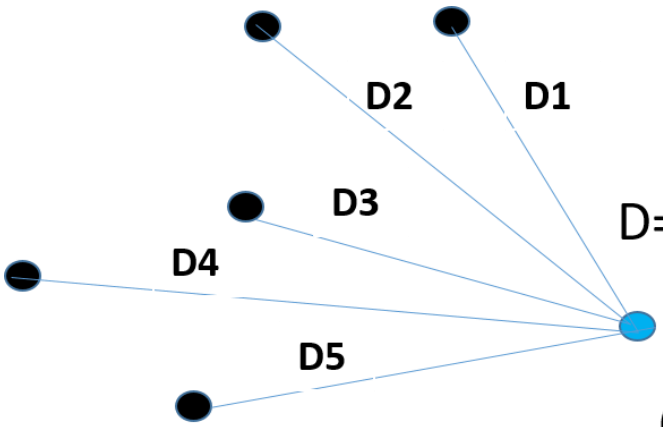
Hamilton:

- ✓ The functions in the algorithm make sure that rules for the shortest path should be followed and should also deals with the calculations of the shortest path given the distances for all the points.
- ✓ Hamilton is the similar type of technique which states “Each vertex should be met exactly once”.
- ✓ This is crucial part of the OCR Algorithm in which the path will be decided for the given image.

Testing for a single image:

- ✓ After getting the training files into the pickle files, connecting to the interface to get images for testing the image.
- ✓ These images will be kept in a particular folder for the access and this is also done in the backend to get those images.
- ✓ In this Alphabets and numbers are also included in the folder separately.
- ✓ This is the function func1() which receives the image and produces the output for the interface.

Confidence level:



D= Sum of all the distances

Confidence level

$$i^{th} \text{ Guess} = 100\left(1 - \frac{Di}{\sum D}\right)$$

By using this algorithms, it gave the efficiency of

Numbers: (For the test of 10000 images)

93.45% - first guess

95.8% - second guess

$> 97\%$ - third guess

Alphabets: (For the test of 2600 images)

85.6% - first guess

88.8% - second guess

90.3% - third guess.

Thus the algorithm can be implemented using python to get the image test done.

This same thing can be applied for Alphabets as though pictures are not shown.

This is just not to confuse the reader from understanding the concept.

Graphical User Interface (GUI):

- ✓ The GUI Interface is used to create an interface for user to draw images of Alphabets or Number and Predict them through the Character Prediction Algorithm.
- ✓ The packages used in developing the Interface are tkinter, PIL and Python.

- ✓ Tkinter is Python's default GUI library. It is based on the Tk toolkit.
- ✓ *Tkinter* provides Python applications with an easy-to-program user interface. Tkinter supports a collection of Tk widgets. A widget is a small graphical component displayed on the screen such as button, text label, drop-down menu, scroll bar, picture, etc.).
- ✓ The PIL (Pillow Image Library) is represented by an Image module which provides a number of functions, including functions to load images from files, and to create new images.

Pros and cons:

- ✓ This algorithm only works for a single Alphabet or Number image.
- ✓ This algorithm won't work for the Words or Sentences.
- ✓ This algorithm won't guarantee that it gives the exact result and it only guesses the results.
- ✓ The algorithm written is not much efficient than the existing ones on the internet.
- ✓ This algorithm gave the good results with three guesses or confidence it gives more than 97% for numbers and 90% for Alphabets within three guesses
- ✓ We have not used any neural networks in the process so it gives less accuracy compared to that

Future Scope:

- ✓ Now algorithm is able recognize single letter English Alphabets and Numbers, it can also be built for other languages.
- ✓ It can be developed to recognize words.
- ✓ More accuracy can be acquired using some more prediction techniques.
- ✓ It can also be developed for differentiating the letters.

Conclusion:

- ✓ These are only the snippets of the code which are useful for the explanation of the algorithm.
- ✓ This has been done with the learnt concepts and some new concepts which helps in developing the project.
- ✓ If we use neural networks the accuracy and efficiency can be drastically improved.
- ✓ There is so much scope of developing this algorithm.
- ✓ I thank the mentor, Udhaya Ravishankar sir for his constant support.
- ✓ For any clarifications you can always approach and can mail me at ‘gorthipalliakhilsai@gmail.com’.
- ✓ I will be very grateful for giving this opportunity, at last I can pray to Bhagawan Sri Sathya Sai Baba to not leave my hand and his grace.

References:

https://www.geeksforgeeks.org/python-introduction-matplotlib/
https://www.geeksforgeeks.org/python-numpy/
https://fsix.github.io/mnist/Deskewing.html
https://scikit-learn.org/stable/user_guide.html
https://docs.scipy.org/doc/scipy/reference/
https://www.geeksforgeeks.org/python-dictionary/
https://www.geeksforgeeks.org/python-list/