Train a system to recognize 5 English characters Recognize how often the trained characters are present and locate them. (Trained and tested on different font styles)

Anshul Agarwal *(IWM2017008)*, Vardhan Malik *(IWM2017007)*, Aman Gupta *(IWM2017006)*

***Abstract*—This paper trains a system to recognise some 5 English characters (uppercase only). Here we are considering the words where characters are in upper case. It will recognize how often the trained characters are present. Also, we locate the characters in the words. Training and testing on different text fonts is feasible here.**

I. KEYWORDS

Matrix, Manhattan distance, Image, Scan, Gray scale, Minimum distance, 5 characters, Uppercase, Words.

II. INTRODUCTION

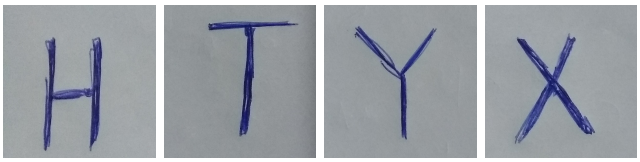
A Digital image in a computer is presented by pixels matrix. Each pixel of such image is presented by one matrix element – integer from the set. The numeric values in pixel presentation are uniformly changed from zero (black pixels) to 255 (white pixels).

All Image Processing Techniques focused on gray level transformation as it operates directly on pixels. The gray level image involves 256 levels of gray and in a histogram, horizontal axis spans from 0 to 255, and the vertical axis depends on the number of pixels in the image.The gray scale images are the best images to work with in case of dealing with the graphs.

The Pixel values are mostly in the RGB format.The Alphabets ,if looked precisely, can be represented as the Matrix. The handwritten image on being processed can be mapped with that alphabet matrix ,thus providing the recognition of the Character.

The most efficient method of text recognition is considered to be KNN.(K Nearest Neighbours). In this paper we will implement our own simplistic modified version of KNN using Manhattan Distance. Manhattan distance : The sum of the horizontal and vertical distances between points on a grid.Our definition of Manhattan is slightly different. Here the distance is calculated as sum of adjacent 4 pixels with respect to reference pixel.

The 5 Alphabets that we have considered are ”H”,”Y”,”T”,”X”,”K”.



Each character in a word is separated by some columns of 0 intensity value. And thus in this way we can separate the characters so that they can get recognised by the system.

III. ALGORITHM DESIGN AND EXPLANATION

Our approach to given problem includes the following steps:

1. Take the handwritten text to be processed as the input.

2.Initially read the five test images of the chosen alphabets and convert them to gray scale images.

3.Now for each of the 5 pixel matrix of each alphabet, resize them to a suitable fixed size.

4.For all 5 matrices, if the pixel value is less than 128 then assign it as 0 otherwise assign 1.The matrix so obtained can be treated as a adjacency matrix.

5.Break the handwritten text into alphabets by finding a column with complete zero intensity.

6. Calculate the Manhattan distance of each of the alphabet with the 5 trained matrices using a 3x3 mask and the matrix with minimum Manhattan distance is considered as the matching character.

The pseudo-code for this algorithm is:-

IV. ALGORITHM ANALYSIS

The above algorithm is designed to reduce the time complexity for this question as much as possible.The image conversions are carried out using cv2 python library. Since the algorithm is trained for 2 images of each alphabet and

Algorithm 1 Image Training and Testing

```
call ReadImage() function                                   ▷ t ← t+1
call ResizeImage() function                                 ▷ t ← t+1
call ConvertGrayImage() function                            ▷ t ← t+1
while i < rows do                                           ▷ t ← t+1
    while j < columns do                                    ▷ t ← t+1
        if gray[i, j] <= 128 then                           ▷ t ← t+1
            gray[i, j] = 0                                  ▷ t ← t+1
            gray[i, j] = 1                                  ▷ t ← t+1
        end if
    end while
end while
```

Algorithm 2 Manhattan Distance Calculation

```
create distance storage                                     ▷ t ← t+5
while i < rows do                                           ▷ t ← t+1
    while j < columns do                                    ▷ t ← t+1
        if gray[i, j] == 1 and test[i, j]! = gray[i, j] then
▷ t ← t+1
            distance[x] = distance[x] + 1                   ▷ t ← t+1
        end if
    end while
end while
find for each matrix Minimum(distance[x]) inbuilt
library function                                            ▷ t ← t+5
```

the Manhattan distance is calculated not only with the desired pixel but also with the adjacent edges with the help of 3x3 mask, thus increasing the efficiency of the algorithm.

V. Experimental Study

Let the input images be :

After appltying the algorithm the manhattan distance between pixel matrix of input image with pixel matrix of handwritten images of H,K,Y,T comes out to be

Training image	Manhattan distance
H	5624
K	5489
Y	4846
T	4788

Training image	Manhattan distance
H	8716
K	8609
Y	8372
T	8355

Minimum Manhattan distance for fig(1) 4788, which is between fig(1) and training T image,so fig(1) is classified as T.

Minimum Manhattan distance for fig(2) 8355, which is between fig(2) and training T image,so fig(2) is classified as T.

A. Time Complexity

The time complexity analysis of an Algorithm is quite necessary to learn about it's efficiency and optimization.

The time complexity for this algorithm is-

$$T = O(max(h * w))$$

Where h is the height and w is width of 2D array of pixels, maximum product of height and width of all the training and testing images will be time complexity.
While reading and resizing the image each pixel of image is considered so image with maximum number of pixel will be max time taken by algorithm.

B. Space Complexity

The Space Complexity for this algorithm is-

$$O(max(h * w)).$$

Where h is the height and w is width of images pixel matrix and maximum product of h and w of all images be space complexity.
A digital image is a 2D array of pixel. So image with the maximum number of pixel will be space complexity.

VI. Conclusion

In this paper, we implemented our own version of KNN using Manhattan distance and image masking by creating a 3x3 mask.The words were broken to characters using the 0 intensity concept.The image was considered as an adjacency matrix. The issues faced were on the location of unmatched character and it's frequency. The proposed algorithm is best optimised to our knowledge and has a quadratic time complexity.

REFERENCES

[1] J. M. White and G. D. Rohrer, "Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction," in IBM Journal of Research and Development, vol. 27, no. 4, pp. 400-411, July 1983, doi: 10.1147/rd.274.0400.

[2] P. C. Loizou and A. S. Spanias, "High-performance alphabet recognition," in IEEE Transactions on Speech and Audio Processing, vol. 4, no. 6, pp. 430-445, Nov. 1996, doi: 10.1109/89.544528.

[3] Online "https://github.com/abidrahmank/OpenCV2-Python-Tutorials"

[4] Introduction to Algorithm 3rd Edition By Thomas H.Cormen

[5] Online "https://towardsdatascience.com/module-6-image-recognition-for-insurance-claim-handling-part-i-a338d16c9de0"

[6] Online "https://www.overleaf.com"