Hand-written Digit Recognition System

CS 512

Instructor: James Abello

Members: Kuo-wei Chung, Shubham Sinha, Rong Zhang

Outline

Project Goal

Project Description

Project Implementation

Future Work

Reference/Sources

Acknowledgements

Project Goal

- To develop a handwritten digit recognition system which can recognize a user input image.
- To use K-NN algorithm to classify the user input image.
- Implement an user interface to show the image and the classified result.

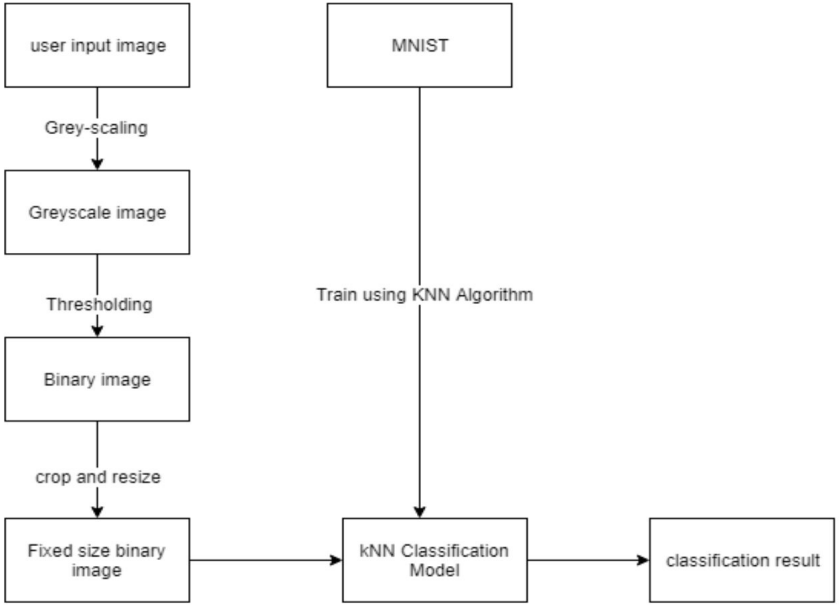
Project Description

- This project proposed a Handwritten Digit Recognition system.
- The user can enter a digit as an image of the handwritten image. This image will be fed to a K Nearest Neighbour algorithm which will compare it to the images of handwritten digits from MNIST dataset.
- Using this system, we can allow post offices to have an automated distribution system by recognizing the handwritten zip code on the envelopes or the packages.

Data Collection

- Training data: MNIST dataset. This dataset consists of many images of hand-written digits and the labels of these digits.
- Test data(user input image): test data will be the user input image, which can be a photo of the handwritten digit.

Flow Diagram



Experimental Setup

- This project is implemented using MATLAB R2016b and written in MATLAB code. The GUI is implemented using MATLAB also.
- The MNIST dataset was downloaded from http://yann.lecun.com/exdb/mnist/.
- The test images were the photos of the hand-written digits took by our mobile phones.

Image Preprocess

- The preprocess consists of grayscaling, binarizing, cropping and resizing.
- Grayscaling: Transfer the input image to grayscale image.
- Binarizing: Transger the grayscale image to binary image
- Cropping: Crop the border of the image and get the portrait of the digit.
- Resizing: Resize the image to 28*28 for classification.

Validation Step

- The MNIST dataset consists of 60000 training images and 10000 test images, each of which is abinary image of size 28*28
- The 10000 test images were divided into a ratio of 3:2
- Larger division used for validation and smaller used for testing accuracy
- To set the value of k, we use the 60000 training images to train and the 6000 images in the validation set
- Iterating over k=1 to 9, k=3 was found to give the maximum accuracy of 95.17% on validation set
- Using Manhattan distance as the distance metric the accuracy was found to be 94.33% on the validation set, which is lower
- Euclidean distance was used as the distance metric
- Testing with the testing data the accuracy comes out to be 97.98%

Finding the Parameters for KNN Algorithm

We record the accuracy on MNIST test data on varying k value using Euclidean distance.

As the accuracy is maximum for k=3 for the test data, we have fixed k=3 for the kNN algorithm.

We test the algorithm using Manhattan distance when k=3 and get the accuracy which is 94.33%.

Euclidean distance performs better in this project, so we use **kNN with Euclidean distance and k=3** to classify the user input.

k	Accuracy
1	95.15
2	94.15
3	95.17
4	94.7
5	94.93
6	94.62
7	94.83
8	94.65
9	94.85

kNN Algorithm

- input: train set[matrix], train label[matrix], test[matrix]
- output: label[integer]
- Step 1: Calculate distances according to Euclidean distance
- Step 2: Set k=3 as it gives the max accuracy on MNIST validation data
- Step 3: Decide the final classification result

Success Cases

Training data: MNIST training set.

Input: Image of the handwritten digit 5 on paper

Output: Classification result (a digit between 0 an 9)

```
>> mnistOnUser
     5
```

From left to right the images are: raw RGB image, grayscale image, binary image,cropped image, resized and color-inverted image

Success Cases

Training data: MNIST training set.

Input: Image of the handwritten digit 2 on white board

Output: Classification result (a digit between 0 an 9)

```
>> mnistOnUser
     2
```

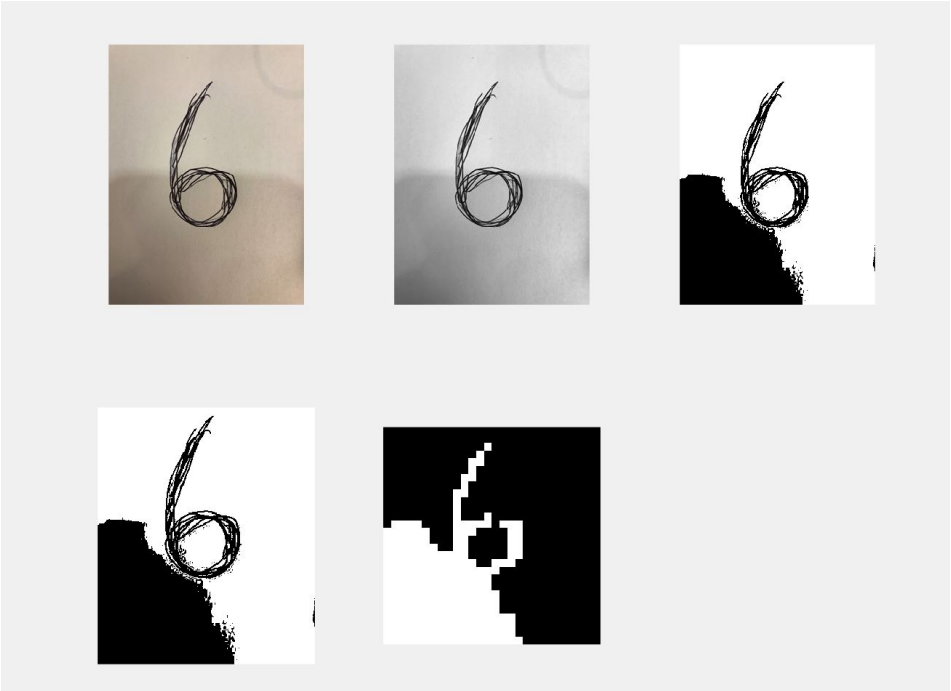
From left to right the images are: raw RGB image, grayscale image, binary image,cropped image, resized and color-inverted image

Failure Cases

Because of the lighting conditions of the image, our approach may face failure.

The classification result is 4 of the given image which is a digit 6.

```
>> mnistOnUser
     4
```

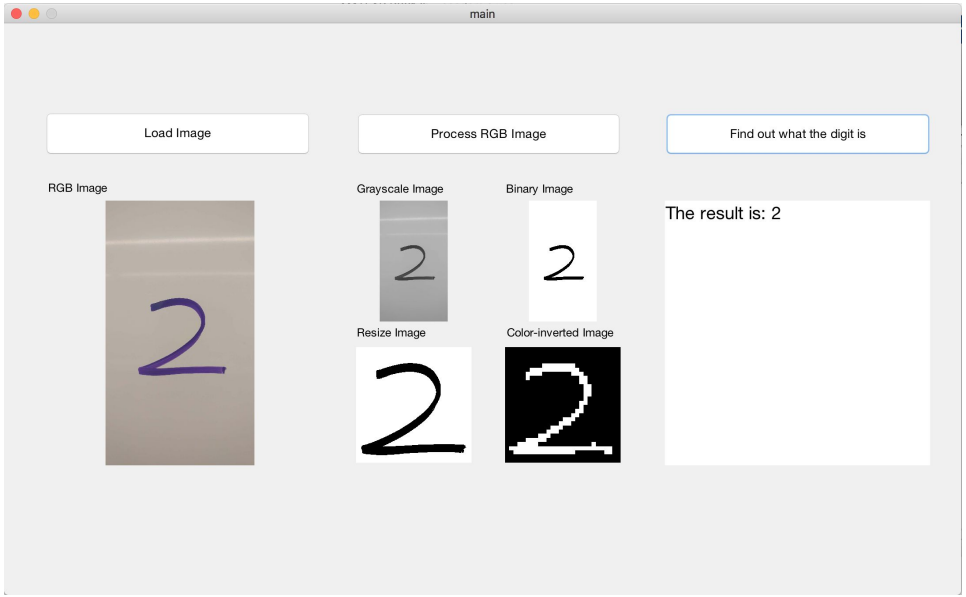


From left to right the images are: raw RGB image, grayscale image, binary image,cropped image, resized and color-inverted image

Limitations

- Performs well only on images where the lighting is good, i.e. uniformly lighted
- Doesn’t work on handwritten alphabets or symbols
- Works only for a single digit
- Time complexity is high

User Interface



Load Image: Choose an image from you computer for recognization.

Process RGB Image: Pre-process the image and get the 28*28 image for classification

Find out what the digit is: Apply kNN algorithm to the image and predict the value of the image.

Future Work

- Trying to apply and test the algorithm for handwritten alphabets and symbols
- Trying to read and recognize multidigit numbers and multiletter words
- Apply better image processing algorithms to allow it to read images with shadows properly

References

- [LeCun et al., 1998a] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.
- https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

Resources

- MATLAB

Acknowledgements

We thank Dr. James Abello Monedero, Harsha Srimath Tirumala for supporting us and giving useful insights throughout the development of this Project.