School of Science Engineering

HANDWRITTEN DIGITS RECOGNITION: IMAGE CORRELATION VS MACHINE LEARNING

EGR 4402 - Capstone Design

April 16, 2018

Sara Elouafiq

Supervised by: Dr. Naeem Nisar Sheikh

RECOGNIZING HANDWRITTEN DIGITS: IMAGE CORRELATION VS MACHINE LEARNING

Capstone Report

Student Statement:

I affirm that I have applied ethics throughout during the whole process of designing and implementing the project. Also, I have held the safety of the public to be paramount and I have addressed this in the presented design wherever may be applicable.

[illegible]

Sara Elouafiq

Approved by Supervisor

[illegible]

Dr. N. N. Sheikh

Acknowledgements

As I have reached the end of this project, I got the chance to reflect on the last four years I have spent at Al Akhawayn University, and how thankful I am for this amazing journey. I would like to express my deepest gratitude to my supervisor, Dr. Naeem Nisar Sheikh, for his hard work, dedication, guidance, help, and availability for the whole semester, as well as for the semesters before. Thanks to him my journey at AUI was both instructive and enjoyable. I would, also, like to thank my family who supported me both financially and emotionally. They always stood by my side and believed in me. They have never let me give up on anything and pushed me towards achieving my goals. Furthermore, I would like to express my recognition to the rest of the professors who helped shape the person I am today and helped me reach a higher level of knowledge through their assistance, guidance, and motivation. Also, I would like to thank all the staff members in AUI for their hard work and dedication which makes our lives on campus much easier. And last but not least, I would like to express my thanks and acknowledgement to my friends who always pushed me to give my best and helped me keep the balance between my studies, health, and social life. They have always been by my side in all kinds of situations. Without them these last four years would have never been the same.

A big thank you to everyone who made this journey one of a kind.

Table of Content

List of figures

Abstract

Optical Character Recognition (OCR) is a subfield of Image Processing which is concerned with extracting text from images or scanned documents. In this project, we have chosen to focus on recognizing handwritten digits available in the MNIST database. The challenge in this project is to use basic Image Correlation, also known as Matrix Matching, techniques in order to maximize the accuracy of the handwritten digits recognizer without going through sophisticated techniques like machine learning.

Keywords: *Image Processing, Optical Character Recognition, Handwritten Digits, Image Correlation, Matrix Matching, Machine Learning.*

1. Introduction

It is easy for the human brain to process images and analyse them. When the eye sees a certain image, the brain can easily segment it and recognize its different elements. The brain automatically goes through that process, which involves not only the analysis of this images, but also the comparison of their different characteristics with what it already knows in order to be able to recognize these elements. There is a field in computer science that tries to do the same thing for machines, which is Image Processing.

Image processing is the field that concerns analysing images so as to extract some useful information from them. This method takes images and converts them into a digital form readable by computers, it applies certain algorithms on them, and results in a better quality images or with some of their characteristics that could be used in order to extract some important information from them.

Image processing is applied in several areas, especially nowadays, and several softwares have been developed that use this concept. Now we have self driven cars which can detect other cars and human beings to avoid accidents. Also, some social media applications, like Facebook, can do facial recognition thanks to this technique. Furthermore, some softwares use it in order to recognise the characters in some images, which is the concept of optical character recognition, that we will be discussing and discovering in this project.

One of the narrow fields of image processing is recognizing characters from an image, which is referred to as Optical Character Recognition (OCR). This method is about reading an image containing one or more characters, or reading a scanned text of typed or handwritten characters and be able to recognize them. A lot of research has been done in this field in order to find optimal techniques with a high accuracy and correctness. The most used algorithms that proved a very high performance are machine learning algorithms like Neural Networks and Support Vector Machine.

One of the main applications of OCR is recognizing handwritten characters. In this project, we will focus on building a mechanism that will recognize handwritten digits. We will be reading images containing handwritten digits extracted from the MNIST database and try to recognize which digit is represented by that image. For that we will use basic Image Correlation techniques, also referred to as Matrix Matching. This approach is based on matrices manipulations, as it reads the images as matrices in which each element is a pixel. It overlaps the image with all the images in the reference set and find the correlation between them in order to be able to determine the digit it represents.

The goal of this project is to apply and manipulate the basic image correlation techniques to build program and keep polishing and enhancing in order to investigate to which extent it can get improved. This would allow us to see how far we can go, in terms of accuracy and performance, but using just the very simple and basic techniques of matrix matching and without going into complicated methods like machine learning.

2. Background

In this section, we will present the main concepts that this project is concerned with.

2.1. Image Processing

Image processing is a very wide field within computer science which deals mainly with analysing images and trying to get some information out of them. The image to be processed is imported then analysed using some computations, which, by the end, results either in an image with a better quality or some of the characteristics of this image depending on the purpose of this analysis [1]. This is a very wide field within computer science, which also has several other subfields of which Optical Character Recognition that we will be mainly dealing with throughout this project.

2.2. Optical Character Recognition (OCR) - History

It is easy for the naked eye to recognize a character when spotted in any document; however, computers cannot identify the characters from an image or scanned document. In order to make this possible, a lot of research has been done, which resulted in the development of several algorithms that made this possible. One of the fields that specialize in character recognition under the light of Image Processing is Optical Character Recognition (OCR).

In Optical Character Recognition, a scanned document or an image is read and segmented in order to be able to decipher the characters it contains [2]. The images are taken and are preprocessed so as to get rid of the noise and have unified colors and shades, then the characters are segmented and recognized one by one, to finally end up with a file containing encoded text containing these characters, which can be easily read by computers [2].

Optical Character Recognition dates back to the early 1900s, as it was developed in the United States in some reading aids for the blind [3]. In 1914, Emanuel Goldberg was able to implement a machine able to convert characters into “standard telegraph code” [4]. In the 1950s, David Shepard, who was at that time an engineer at the Department of Defense, developed a machine that he named Gismo, which is able to read characters and translate them into machine language [5]. In 1974, Ray Kurzweil decided to develop a machine that would read text for blind and visually impaired people under his company, Kurzweil

Computer Products. There are several softwares and programs, nowadays, which use OCR in several different applications. In 1996, the United States Postal Services were able to develop a mechanism, HWAI, which recognizes handwritten mail addresses [6].

2.3. Methods Used in OCR

A lot of research has been done in the field of OCR, and still being done, which resulted in the development of several algorithms which enable computers to recognize characters from images or scanned texts. Many of these techniques have attained very high efficiency and a low error rate. However, these algorithms are still being investigated and improved for a better performance.

2.3.1. Machine Learning

Machine learning is a field that concerns making programs learn and know how to behave in different situations using data. One of its applications is Optical Character Recognition.

2.3.1.1. Artificial Neural Network

An Artificial Neural Network (ANN) is a system that mimics the human's biological neural network in the brain. It is an algorithm used for machine learning, which means it uses data to learn how to respond to different inputs. The ANN can be seen as a box, which takes one or more inputs and gives one output. Inside the box, there exist several interconnected nodes. The input is fed into the program, which goes through the several layers and nodes of the ANN and gives an output using a transfer function [7].

Artificial Neural Networks are used for OCR and have proved a very high accuracy rate. In this case, the ANN would "recognize a character based on its topological features such as shape, symmetry, closed or open areas, and number of pixels" [8]. The high accuracy of this kind of algorithms is mainly thanks to its ability of learning from the training set, which would contain characters with similar features.

Some Neural Networks have proven a very high performance. An implementation of the ANN done by Simard, Steinkraus, and Platt has reduced the error rate of recognizing handwritten digits from the MNIST dataset to a percentage as low as 0.7% [9]

2.3.1.2. Support Vector Machine

Support Vector Machine (SVM) is an algorithm that belongs to machine learning as well. SVMs are known as high performance pattern classifiers. While Neural Networks aim at minimizing the training error, SVMs have as goal to minimize the “upper bound of the generalization error” [10]. The learning algorithm in this technique is based on classification and regression analysis.

This kind of classifier has been used in the recognition of very complex characters like the Khmer language and has proved a very high performance.

2.3.2. Image Correlation

Image Correlation is a technique used to recognize characters from images. This approach, also referred to as Matrix Matching, uses mathematical computations in order to analyse the images [11].

By using this technique, the images are read as matrices, where each element represents a pixel, which makes it easier to manipulate them using mathematical approaches. The image to be identified is loaded as a matrix and compared to the images in the reference set. The test image is overlapped with each image in the reference set to be able to see how it matches with each one of them so as to tell which one represents it the most. The decision can be made by seeing the pixels that match and the ones left out from either one of the two images.

This technique has many challenges and limitations, as it only overlaps the images and tries to see how much they look alike. By using this method, problems arise when having characters of different sizes, or when one of them is rotated by a certain angle.

2.3.3. Feature Extraction

Feature extraction is a technique based on pattern recognition. The main idea of feature extraction is analysing the images and derive some characteristics from these images that identify each specific element [12]. An example of these characteristics would be the curvatures, the holes, the edges, etc. In the case of digits recognition, these features could be the holes inside the digits (for example for the eight, the six, and maybe the two as well) as well as the angles between some straight lines (for example in the one, the four, and the seven). Whenever an unknown image is to be recognized, its features are compared to these so that it can be classified.

3. Tools

This project's main objective is to be able to read the images containing the handwritten digits and be able to identify those digits using basic image correlation techniques. These images are normally represented and read as matrices, in which every element portrays a pixel. The image correlation technique takes these matrices and compares them using some algorithms so as to identify the match that represents the digit we are trying to figure out. This project will be mainly using matrices and heavy numerical computations, that is why it is very important to consider the tools that would provide us with a suitable environment for performing these computations.

3.1. Octave

Octave is a free and open source software that uses a high-level programming language. It has the same functionalities as Matlab and is compatible with it. It offers a very simple and suitable interface to exert some mathematical computations. It provides some tools to solve mathematical problems like some common linear algebra problems [13]. It is also very efficient when it comes to the use of resources, i.e., time and memory, when it comes to these operations. Also, it is very easy to use it when dealing with matrices, as it provides with many functions and operations that make it less costly to manipulate them. In this project, we will deal with images as matrices, in which each element represents a pixel, that is why it is very necessary for us to choose a tool that will make our computations easier and more efficient in terms of time and memory resources. Both Matlab and Octave are very easy to learn and work with and provide a suitable environment for this kind of projects. We have opted for Octave as it is free and open source.

3.2. MNIST Database

The MNIST database, which stands for the Modified National Institute of Standards and Technology database, is a very large dataset containing several thousands of handwritten digits. This dataset was created by mixing different sets inside the original National Institute of Standards and Technology (NIST) sets, so as to have a training set containing several types and shapes of handwritten digits, as the NIST set was divided into those written by high school students and others written by the Census Bureau workers [14]. The MNIST dataset has been the target of so many research done in recognizing handwritten digits. This allowed the development and improvements of many different algorithms with a very high performance, such as machine learning classifiers.

In order to be able to implement our recognizer and test its performance, it is necessary to have a suitable dataset which contains a large number of handwritten digits. This dataset should be able to allow us to discover the challenges and limitation of the image correlation technique and push us to look for ways and rules to enhance it and assess its accuracy. We have opted for this dataset to be used for testing our program since it has proved a great reliability and importance in the field.

3.3. Feasibility Study

From a technical perspective, since this project makes heavy use of numerical computations, using Octave is a wise choice as it will make the program more efficient.This software will also provide us with some libraries to read and manipulate the images that will make the implementation process easier.

As for the dataset to use in the testing of the project, we have chose the MNIST Database. This database contains thousands of handwritten digits that have been used in the development of programs with a similar aim. This dataset is open for public use with no charges. It is also very convenient for our project and will help us reduce the time by using directly as a test set without having to make one ourselves.

Since all the tools to be used in this project are free of charge and very easy to use, we can conclude that this project is very feasible in terms of financial resources as well as effort and time.

4. Methodology

4.1. Getting Familiar with the Tools

The first step we had to go through while working on this project was getting familiar with the tools used, i.e., Octave and the MNIST dataset. After setting up the environment for Octave to work perfectly and downloading the dataset, I have started experimenting with both in order to get familiar with them and know how to use them easily in the future.

Since all the programming is mainly done in Octave, we had to download it along with its Graphical User Interface into the computer, and learn a little bit about its functions and how to use it. Octave is a free software which makes it very easy to work with matrices and vectors and is very efficient in performing calculations on them. I have started learning how to use it and looking for its main functions that I will be using in the implementation of the project. For that, I have used some random images of digits to see how they can be read and modified as well as how to apply some computations on them.

Moreover, I had to investigate the format of the MNIST dataset and get familiar with its representation.The MNIST dataset, which was used to create our test set, contains thousands of handwritten digits, represented as matrices. It has been used in the development of several programs and projects with the same aim as ours. After downloading the file which contains the handwritten digits, I have loaded it on Octave in order to visualize the images and figure out how to use and manipulate them.

4.2. Creating the reference and test set

One of the main steps in the project is creating the reference and the test set that will both be used in the implementation phase.

The test set is to be used in order to assess the performance of the program and evaluate its success or error rate. It is to be taken from the MNIST dataset, since it contains the handwritten digits that we intend to recognize and identify.

As for the reference set, it is used to compare the test images and be able to identify the digit they represent. It is to be created using different fonts.

4.3. Different Versions

After having a look at the dataset and deciding on tools to be used for the implementation, we have started the development of our mechanism by developing a very basic version. After that, we have started identifying the challenges and problems we have faced and kept enhancing it. We have ended up with several versions different one from another, in each new version we increase the accuracy of the program by improving the method and introducing some new rules.

5. Data

It is very necessary to know the kind of data we are using before we start the design and the implementation of the program. That is why we had to have a look at its format to understand how it is represented before creating the reference and the test set.

5.1. Dataset Format

The dataset that I have downloaded from the MNIST database contains 60,000 images of handwritten digits, from zero to nine, all grouped in one file. Each of the images is of size 28 by 28 pixels and represents a digit. I have noticed that there is no pattern or order to the way the images were organized in the file. The images are represented as matrices, of which the elements represent the pixels. Also, each image has a label that indicates the digit represented. This label was very helpful later on in order to be able to create the test set. Furthermore, the data did not contain noise or any major problems to deal with, that is why it was used without preprocessing it.

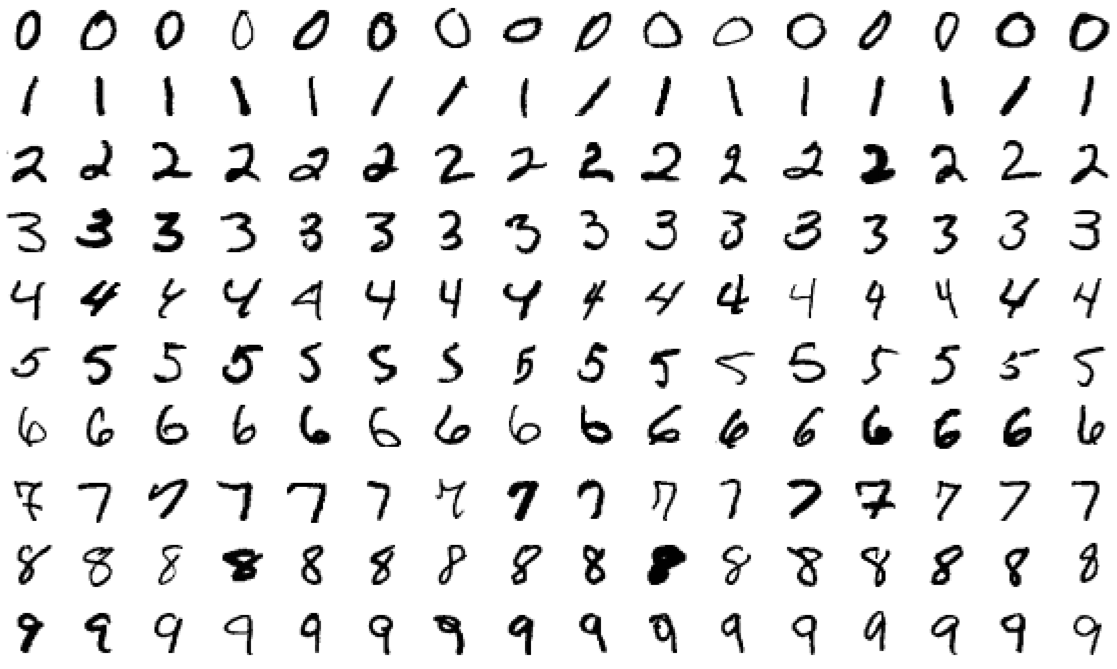


Figure 6.1.1. Example of the MNIST dataset [15]

5.2. Reference Set

To be able to recognize the digit represented by a certain image, it is required to compare it with other images containing known digits to be able to make the decision. For that it is necessary to create a reference set which will contain all these images.

That is to say, each image we would want to recognize is to be compared to the images in the reference set. The image with the highest match is the one that represents the right number. Since handwritten digits differ from a person to another, the reference set needs to have digits with different fonts. That is why, we have created six images of each digit using the online image editor *pixlr.com*, each one with a different font. The reference set contains images with the same dimensions as the ones in the MNIST dataset, i.e., 28 by 28 pixels. Furthermore, these images have a black background and a white font, which made it easier to use and manipulate them later on using Octave. Furthermore, to make the comparison easier, we have regrouped each six images representing the same digit under one file. So the resulting reference set was ten files, each one representing a digit from zero to nine, and containing six images of that digit in different fonts.

The pixels of these images are then changed into zeros and ones, which makes the overlapping of the images easier. The black background was initially represented as zeros, so it is left the same. As for the pixels of the white font, each one of them was represented with a different non zero value depending on the shade of white. These non zero values are all converted into ones. The following image displays the digit “2” reference set. Rest of the reference sets are in Appendix A.

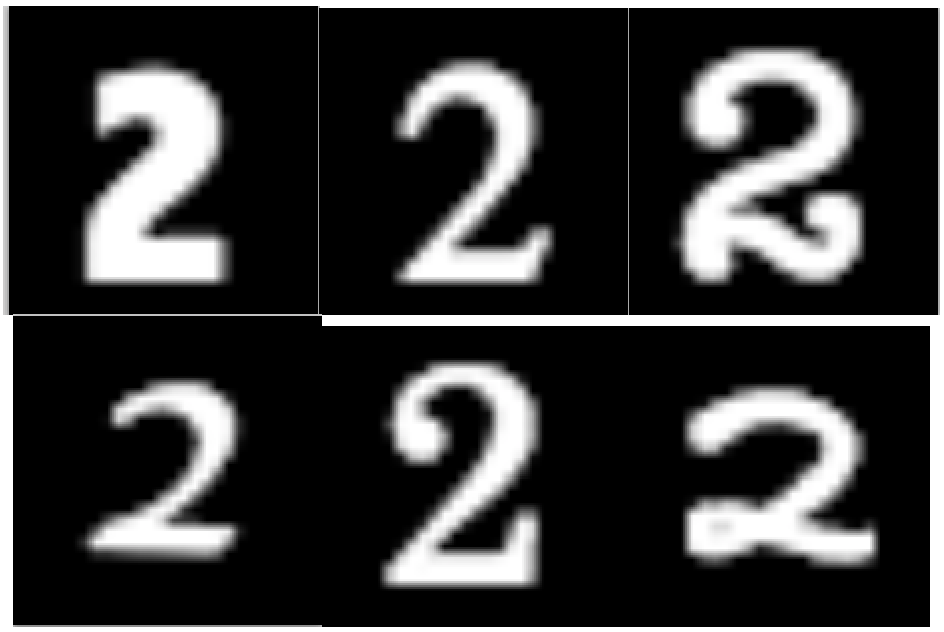


Figure 6.2.1. File of 2’s in Reference Set

5.3. Test Set

The program to be developed needs to be tested against some images that contain handwritten digits so as to be able to assess its performance and calculate its success rate. That is why it is very necessary to create a test set. The test set represents an example of the images containing the handwritten digits which will have to be compared to the images in the reference set so as to identify them.

This set was formed using the file from the MNIST database. The original file contained 60,000 images representing different digits. This made it difficult to look for each number using the label for the testing of the program. In order to make it easier to access each digit we want, we have decided to store a number of images from each digit in a separate file. That is why we have stored 20 images of each digit in ten different files. That is to say, the resulting test set was in the form of ten files, each one of them represents a digit and contains 20 images of it. These images were extracted from the initial file by reading them and their labels using Octave.

In order to make the manipulation of the matrices/images easier, we had to make some modifications in the elements of all the matrices representing the test set as well. The black pixels were originally represented as zeros, so they were left the same. As for the white ones, each of them had a different non zero number, so we turned them all into ones.

6. Implementation

In this project we aim at building a mechanism that would recognize handwritten digits from the MNIST dataset. We have opted for the Image Correlation technique, also referred to as Matrix Matching. The goal of this project is to use the basic and simple concepts of this methods and see how good can we make the accuracy rate without using complex techniques such as machine learning.

We have started the implementation of the program using a very simple and basic method, which is explained in further details under the section Version 1, then we have calculated its error rate. Afterwards, we have tried to spot the problems in the mechanism and find the limitations of the technique in order to improve it, and that is how we ended up with a second version of the program. We kept doing the same thing, each time trying to improve the previous version, which enabled us to keep improving the program and reach a higher accuracy and performance.

6.1. Version 1 - First Maximum

The first version of our mechanism was very straightforward. We have applied the concept of matrix matching literally and just consider the digit with highest match as the one represented by the test image.

In this version, we have take the test image and compare it with every image we have in the reference set. We have ten files as a reference set, each one of them contains six images that we have created before. Since the images are read as matrices, whose elements are zeros and ones, for each file the test image is overlapped with all six images, using the dot product. Therefore, by summing the result of the dot product of both the test and the reference image, we get the number of pixels in which they overlap. We, then, take the maximum overlap out of the six results, and store it in a different matrix M. Each column in the matrix M contains the maximum overlap with a certain digit, the first element being the maximum overlap with the zeros, the second one representing the overlap with the ones, and so on, until the overlap with the nine. Finally, the result is the digit corresponding to the maximum pixels overlap.

This technique, although it might seem logical, resulted in a very low accuracy rate, which is of 12.5%. This represents a reliability rate slightly higher than the random guess, which is 10%, since we have ten digits.

After testing this technique with the 200 images that we have in the test set and looking at the results, we were able to see the kinds of problems that caused the error rate to be very high. The first thing we have noticed is that the majority of the digits we are trying to identify match with the eight the most. This could be explained by the eight being the digit with the most number of pixels, which would make it have a very large overlap with the test images. Also, the shape of this digit allows most of the test images to fit with it.

If the digits in the test images have a very large overlap with the eight but represent another digit, there is a high chance that the number of pixels left out of the reference image, representing the eight, is a very large number. This problem could occur in other cases with the opposite case, where the number of pixels left out of the test image is significantly higher than the one left out of the reference image. That is why we thought about not only considering the number of pixels that do overlap, but investigate the number of pixels left out of both the test and the reference image.

6.2. Version 2 - First and Second Maximum with Rules

In the first version of the program, we have noticed that most of the unknown digits match with the eight. In order to avoid this issue, we have decided to consider the second maximum as well. By just looking at the second maximum and considering it as the number we are trying to identify, the accuracy rate is 22%. In this case we would not take into consideration all the numbers which had a correct match with the first maximum.
For this reason we have decided to introduce some rules that would help us distinguish if the digit in the test image should be mapped with its first or second maximum. In this technique we do not only look at the number of pixels that overlap, but we investigate the pixels left out from both the test and the reference image as well.

The following is a rule that we have concluded from the results of the first trial, which helped us improve the accuracy of this version:

- ❑ If the first maximum is 8, and the second maximum is 2, then we consider the overlap with the reference image of the 5 as well. That is because the shape of the five can match more with the 8 rather than the 2. Therefore, If the difference between the match of the test image and the reference image of 2 and that of the test image and the reference image of the 5 is very low, then it is more likely to be 8, otherwise, we consider it as 2.

Introducing this rule helped us slightly increase the success rate of the program from 22% to 24.5%. This percentage is still very small and needs to be further improved. That is why, we thought that one way to reduce the margin of error in our program is to change the reference set, since we are relying on images that contain typed digits in order to recognize the handwritten ones.

6.3. Version 3 - Upper Left Adjusted Images

In this version of the program, we have tried to solve the issue related to the digits that are either shifted or rotated by an angle. The approach we have considered in order to overcome this challenge is to shift and adjust all the non zero pixels to the upper left of the image (Appendix B). The figure that follows gives an example about this approach with the two different digits 1 and 4.

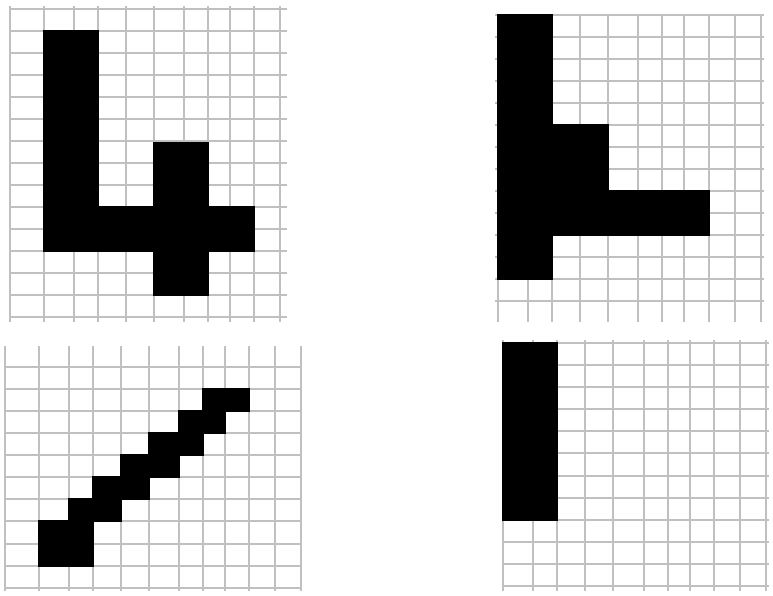


Figure 7.3.1. Illustration of the Upper Left Adjusted Approach

This method did not give us much accuracy, and unfortunately, we did not investigate it more so as to find a way to improve it due to time constraints.

6.4. Version 4 - Extracting the Reference Set from the MNIST Database

The initial reference set we have worked with contained images of types digits with different fonts. The decision of taking different fonts was mainly because to maximize the chances of a correct match, since the handwritten digits are written in different manners.

However, this was not a very accurate approach since the typed digits differ a lot from the handwritten ones. This approach resulted in a very low performance. That is why, we have decided to change the reference set, from the set of files containing the typed digits, into a set of images from the initial MNIST database, but not the same ones we have in the test set (Appendix C).

By following this change of reference set, and by taking just the first maximum without introducing any rules to it, we have an accuracy rate of 52.5%. We can see that the performance has changed drastically by changing the reference set.

In this approach, we would not only take the first maximum as the perfect match for the unknown digit, but we, also, take a look at the second maximum and analyse the relationship between the test digit and these two maximums as well as the pixels left out of all of them in every overlap so as to conclude the rules to be used to improve our mechanism.

That is why, the first step we did is to display matrices containing the maximum overlap with each digit along with the number of pixels left out from the test image and the number of pixels left out from the reference image (Appendix D). These results were used so as to be able to generate the rules.

After looking at the results of the first and second maximum, we have printed the pixels left out from both the test and the reference image, in order to be able to see the relationships between them and how they can help improve the efficiency of the program. We ended up extracting the following rules:

- ❑ If the first maximum is 8, and the second maximum is 1, then we look at the number of pixels left out from reference image of the 8 after the overlap with the test image of 1. If this number is significantly high, then we consider that the unknown digit is a 1, otherwise it is considered to be 8.
- ❑ If the first maximum is 6, and the second maximum is 3, then we look at the number of pixels left out from the reference image of the 6 after overlapping it with the test

image of the 3. If this number is significantly high, then we consider the unknown digit to be a 3, otherwise we take it as a 6.

- If the first maximum is 8, and the second maximum is 7, then we consider the number of pixels left out from the reference image representing the 8. If this number is significantly high, then we consider the unknown digit to be 7, otherwise, we consider it as 8.
- If the first maximum is 7, and the second maximum is 6, then we there is a possibility it is either a 7 or 9. That is why we look at the number of pixels left out from both the reference and the test image of the 9. If the number of pixels left out from the reference image of 9 is almost half the number of pixels left out from the test image of 9, then it is most likely to be a 9, otherwise, we consider it as a 7.
- If none of these rules apply, then we take the first maximum as the digit we are trying to identify.

By applying these rules the accuracy rate of the program went from 52.5% to 57%.

7. Analysis of the Social and Psychological Impact

Implementing this technology on a large scale could evidently bring about several repercussions. If we were to begin with an impact of this approach, the most obvious one is the social effect. Recognizing handwritten digits, if not used with good intentions, could lead to the manifestation of several social issues.

First of all, this method could lead to the increase of theft in societies. That is to say, if someone was to take a picture of a code or pin number of a private account, which maybe written by the owner, or available in a scanned document, the image could be used, even if it is not of a high resolution as it could be taken by a camera located far away from the original document, in order to be able to segment it and extract the code from it. This problem could appear in several other cases. For example, thieves or other people with malicious incentives, could use this technique in order to get some important codes and keys of important ciphers, so as to serve political or personal needs.

This kind of images can also be extracted from social media accounts or from phones. Nowadays, with the increase of technological advancements, the use of social media became very popular, and people started sharing many information through private direct messages, which could also include this types of codes and ciphers. If someone is spying on these people or could penetrate their private accounts in a way or another, they could get them and extract the information they need in order to serve their personal motives.

In addition to that, recognizing handwritten digits could be used in a bad way so as to break the captcha code. This code is used in order to make sure that the entity who is accessing a certain website or is trying to get a benefit from a certain service is a human being and not a computer. The captcha is usually put as an image to make it hard for computers to read it, which will make sure that only human beings can be granted access. However, if digits recognition is used in this case by computers, then they would be able to appear as humans.

Furthermore, these issues could increase the lack of trust in societies, therefore, affect the social behavior of people. As a consequence of the bad uses and applications of this technique, people in different societies could become more suspicious and doubtful of everyone around them. If people notice that this kind of methods could be used in spying and theft, then there will no longer be trust inside societies and the problem of wariness will end up being elevated. This problem will affect the behaviors of these people towards each other.

Consequently, this would lead to the increase of the lack of communication as well as misunderstandings.

If everybody could see how this technology could be used against their own benefit and used to serve some immoral motives this would lead to the creating an environment of lack of trust can have a psychological impact on human beings. As an example, this problem could lead to emotional distress as people would be worried, all the time, that someone is spying on them or trying to use their personal and private information for their own incentives.

If this kind of behavior is proven to be true, then it would raise some questions and concerns related to psychology. In other words, when people strive to look for other people's personal information in order to use it for their own benefits or to serve some other party's evil intentions this means that there is a whole psychological factor behind it that is pushing them towards this behavior. If these psychological problems could be traced, it could be possible to reduce them or prevent them in the future in order not to fall in this kind of problems to begin with.

8. STEEPLE

8.1. Social

Image processing in general can cause the opposition of some people from different societies. The reason behind this is because this field can be further developed for malicious intentions, perhaps in the military field to be able to target certain people, and many other similar possibilities. That is why if image processing is not well managed and used with a wise mentality and good intentions, it could represent a threat to societies.

8.2. Technological

The output of this project is a program that can read images of an unknown handwritten digits and be able to identify. Therefore, it can be useful in the future to identify images containing handwritten digits. Moreover, it provides some techniques that could be further developed in order to be able to recognize all types of characters in images or scanned documents.

8.3. Economic

This kind of projects can be used by companies and could be implemented in some softwares, which could be able to reduce their costs. In 1997, HWAI, developed by the US Postal Services in order to recognize handwriting in letters was very successful in cost reduction, as they have reached eight billion dollars in cumulative [16].

8.4. Environmental

This project does not have any effect on the environment, therefore, does not need to take into consideration any environmental factors.

8.5. Political

This project does not have any direct impact on the political factors. However, if we take it into the broader context of image processing as a whole, then we can say that this field could be used by some political parties to reach their objectives, either with good or bad motives, as mentioned earlier in the societal section.

8.6. Legal

Image processing needs to be taken into consideration by all the authorities, as they need to enforce some laws on its uses and applications to ensure that none of them could be used against the well-being and serenity of humanity.

8.7. Ethical

This project may open the discussion to a very large ethical dilemma related to the reduction of human labor due to computerization. The wider field of image processing may appear as a risk of having a lower need for human beings in some fields. Going back to the example of HWAI by USPS, this technology was able to reduce 100 million dollars of labor costs, which means it has reduced the number of employees. Moreover, this field could raise some ethical questions about its applications. Since using image processing can enable us not only to develop characters recognition but facial recognition as well, in addition to multiple other applications, people could be worried about preserving their privacy and security.

9. Challenges and Limitations

This project was my first encounter with Optical Character Recognition (OCR). That is why, while working on this project I was faced with many challenges and issues. First of all, it took me a long while to understand all the concepts and get familiar with them, from image processing to OCR, to all of the techniques and algorithms used in it. Furthermore, the data we were dealing with was very problematic in terms of the way the digits are written. Since some of the digits were rotated by an angle, some of them were thicker or thinner than the rest, and some of the digits were not well centered or were written in confusing ways. In addition to that, overcoming these challenges was not easy since we were only using basic image correlation techniques. We have tried to maximize the success rate, but we have only reached 57%, which is not a very high performance.

10. Conclusion and Future Work

Optical Character Recognition is a very broad field concerned with turning an image or a scanned document containing a set of characters into an encoded text that could be read by machines. In this project, we have attempted to build a recognizer for handwritten digits using the MNIST dataset. The challenge of this project was to be able to come up with some basic image correlation techniques, instead of some sophisticated algorithms, and see to what extent we can make this mechanism accurate. We have tried several versions and kept trying to improve each one in order to reach a higher performance rate. The last version has reached a rate of 57% accuracy. Unfortunately, we could not compare the performance of the mechanism we have built to some others that have already been designed and/or implemented before because we did not find any academic paper that tackles this method. The performance we have reached is far less than that of machine learning, which reaches a performance rate of 99.3%; however, it could be further improved and made into a better one. The goal of this project was to explore the field of OCR and try to come up with some techniques that could be used without going into deep computations, and even if the final result is not very reliable, it still provides an accuracy way better than random.

The future steps that to go for would be having a closer look at the results of all the versions in order to find new rules. By extracting and implementing them, we will be able to enhance the performance of these versions. Moreover, it would be good if we could make some modifications to both the reference set and the rules in order to make our program more general and able to identify both typed and handwritten digits.

Furthermore, in the future, we could make a great use of the matrices that indicate the first maximum overlap of each test image with the reference images, along with the number of pixels left out from both. These matrices could be used with some clustering algorithms to build a program able to recognize handwritten digits with a very high efficiency.

Last but not least, we thought about using linear or high level regression in the versions we have developed in order to create more rules. As regression could be used for binary classification and is not very suitable to classify a digit out of ten, this technique could be used in order to tell which digit is the most suitable, the first maximum or second maximum, which will enable us to generate more rules; thus, reach a higher efficiency.

11. References

[1] G. Anbarjafari, “1. Introduction to image processing,” Sisu@UT. [Online]. Available: https://sisu.ut.ee/imageprocessing/book/1.

[2] S. Tanner, “Deciding whether Optical Character Recognition is feasible” [Online]. Available: http://www.odl.ox.ac.uk/papers/OCRFeasibility_final.pdf

[3] H. F. Schantz, The history of OCR, optical character recognition. Manchester Center, VT:

Recognition Technologies Users Association, 1982.

[4] History of Computers and Computing, Internet, Dreamers, Emanuel Goldberg. [Online]. Available: https://history-computer.com/Internet/Dreamers/Goldberg.html.

[5] “Optical Character Recognition: What you Need to Know”. Phoenix Software International. Available: http://www.phoenixsoftware.com/pdf/ocr_data_entry.pdf

[6] S. N. Srihari, & E. J. Kuebert. Integration of Hand-Written Address Interpretation Technology into the United States Postal Service Remote Computer Reader System. Available: http://www.cedar.buffalo.edu/papers/articles/HWAI-RCR97.pdf

[7] V. Sharma, S. Rai, & A. Dev (2012). International Journal of Advanced Research in Computer Science and Software Engineering. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.9353&rep=rep1&type=pdf

[8] V. Shrivastava, & N. Sharma (2012). Artificial Neural Network Based Optical Character Recognition. Available: https://pdfs.semanticscholar.org/e0df/4d4c89af84b6caa250ba26e7f355258968de.pdf

[9] P. Y. Simard, D. Steinkraus, & J. Platt. Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. Available: https://www.microsoft.com/en-us/research/publication/best-practices-for-convolutional-neural-networks-applied-to-visual-document-analysis/?from=http%3A%2F%2Fresearch.microsoft.com%2Fapps%2Fpubs%2F%3Fid%3D68920

[10] H. Byun, & S. W. Lee (2012). Applications of Support Vector Machines for Pattern Recognition: A Survey. Available:

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.723.5893&rep=rep1&type=pdf

[11] R. Cintron, & V. Saouma (2008). Strain Measurements with the Digital Image Correlation System Vic-2D. Available: https://nees.org/site/resources/pdfs/cintron_final_paper.pdf

[12] P. Singh, & S. Budhiraja. Feature Extraction and Classification Techniques in O.C.R. Systems for Handwritten Gurmukhi Script – A Survey. Available: http://www.ijera.com/papers/Vol%201%20issue%204/BQ01417361739.pdf

[13] “About”. GNU Octave. Available: https://www.gnu.org/software/octave/about.html

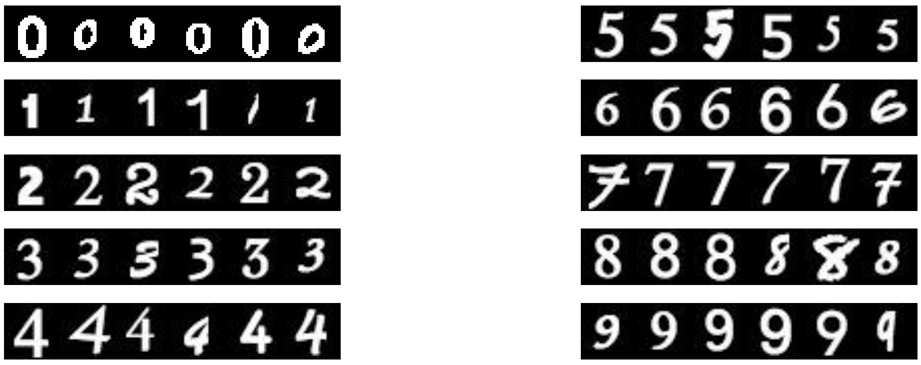
[14] Y. LeCun, C. Cortes, C. J. C. Burges. The MNIST Database of Handwritten Digits. Available: http://yann.lecun.com/exdb/mnist/

[15] “MNIST database,” Wikipedia, 24-Apr-2018. [Online]. Available: https://en.wikipedia.org/wiki/MNIST_database.

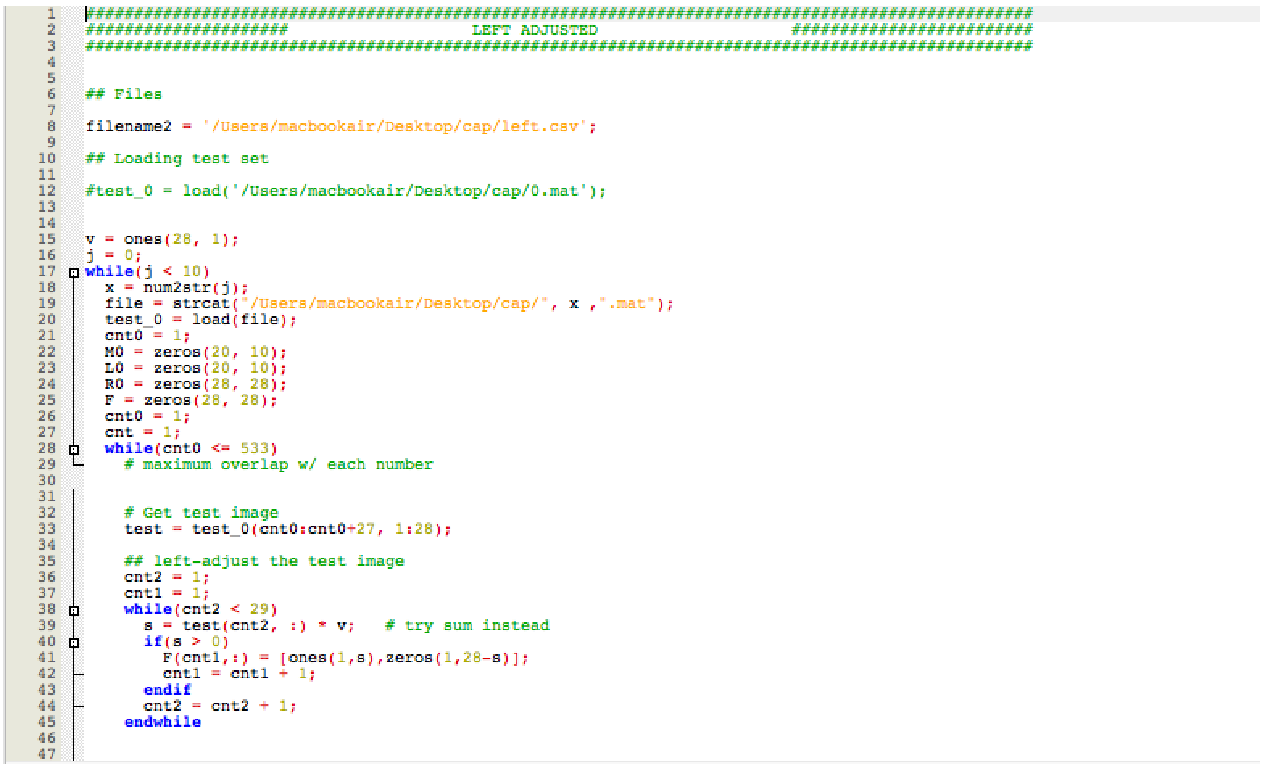
[16] “Landmarks in Postal Research at CEDAR”. CEDAR University of Buffalo. Available: http://www.cedar.buffalo.edu/~srihari/PostalResearch.pdf

12. Appendices

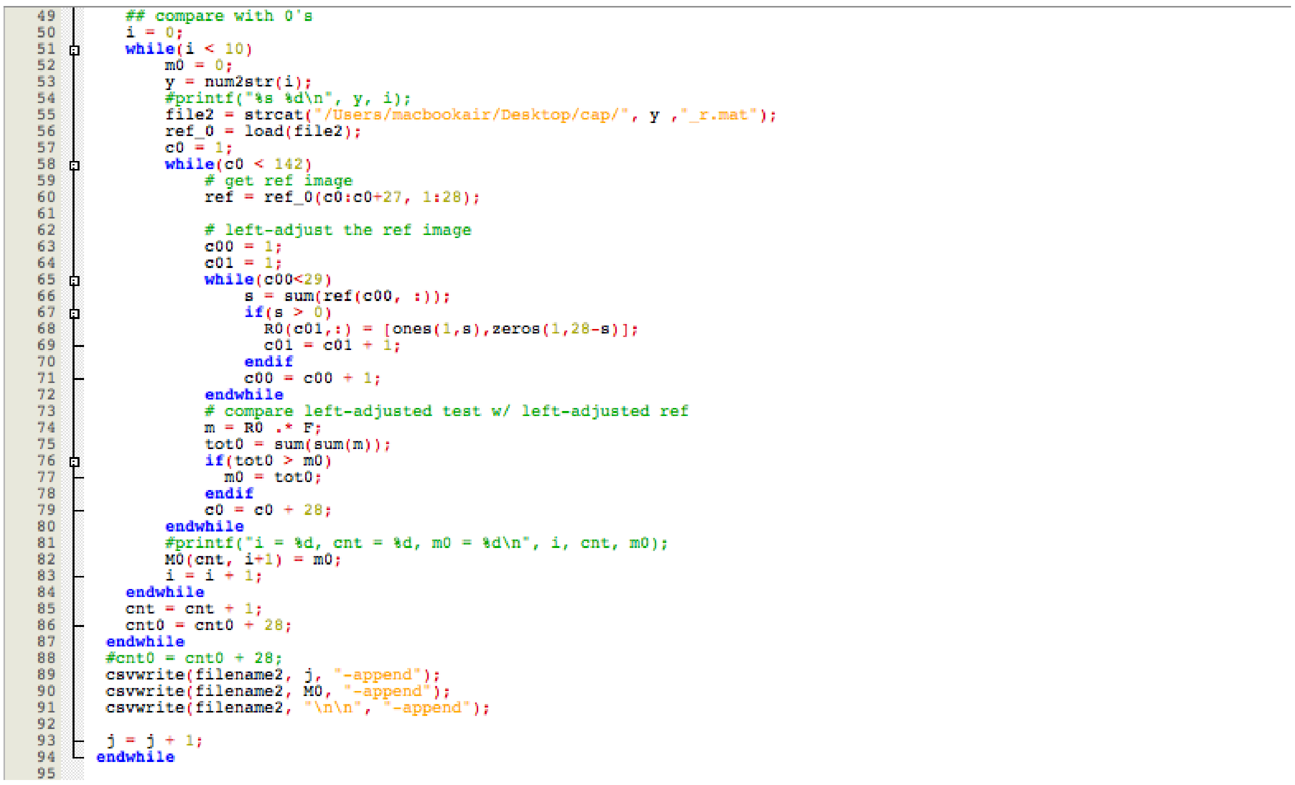
APPENDIX A. Reference Set



APPENDIX B. Version 3 - Upper Left Adjusted

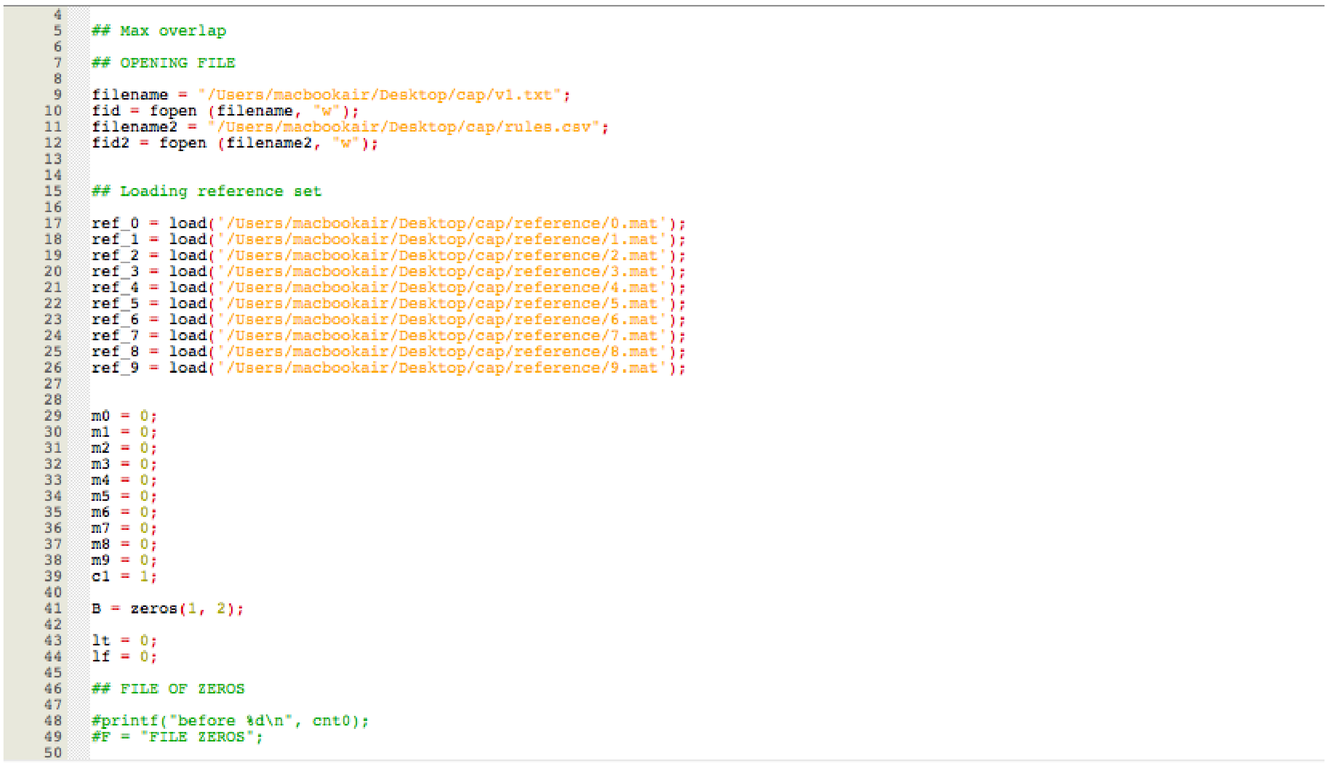


```
##################################################################################################
#####################                   LEFT ADJUSTED                    #########################
##################################################################################################


## Files

filename2 = '/Users/macbookair/Desktop/cap/left.csv';

## Loading test set

#test_0 = load('/Users/macbookair/Desktop/cap/0.mat');


v = ones(28, 1);
j = 0;
while(j < 10)
  x = num2str(j);
  file = strcat("/Users/macbookair/Desktop/cap/", x ,".mat");
  test_0 = load(file);
  cnt0 = 1;
  M0 = zeros(20, 10);
  L0 = zeros(20, 10);
  R0 = zeros(28, 28);
  F = zeros(28, 28);
  cnt0 = 1;
  cnt = 1;
  while(cnt0 <= 533)
    # maximum overlap w/ each number


    # Get test image
    test = test_0(cnt0:cnt0+27, 1:28);

    ## left-adjust the test image
    cnt2 = 1;
    cnt1 = 1;
    while(cnt2 < 29)
      s = test(cnt2, :) * v;   # try sum instead
      if(s > 0)
        F(cnt1,:) = [ones(1,s),zeros(1,28-s)];
        cnt1 = cnt1 + 1;
      endif
      cnt2 = cnt2 + 1;
    endwhile

```

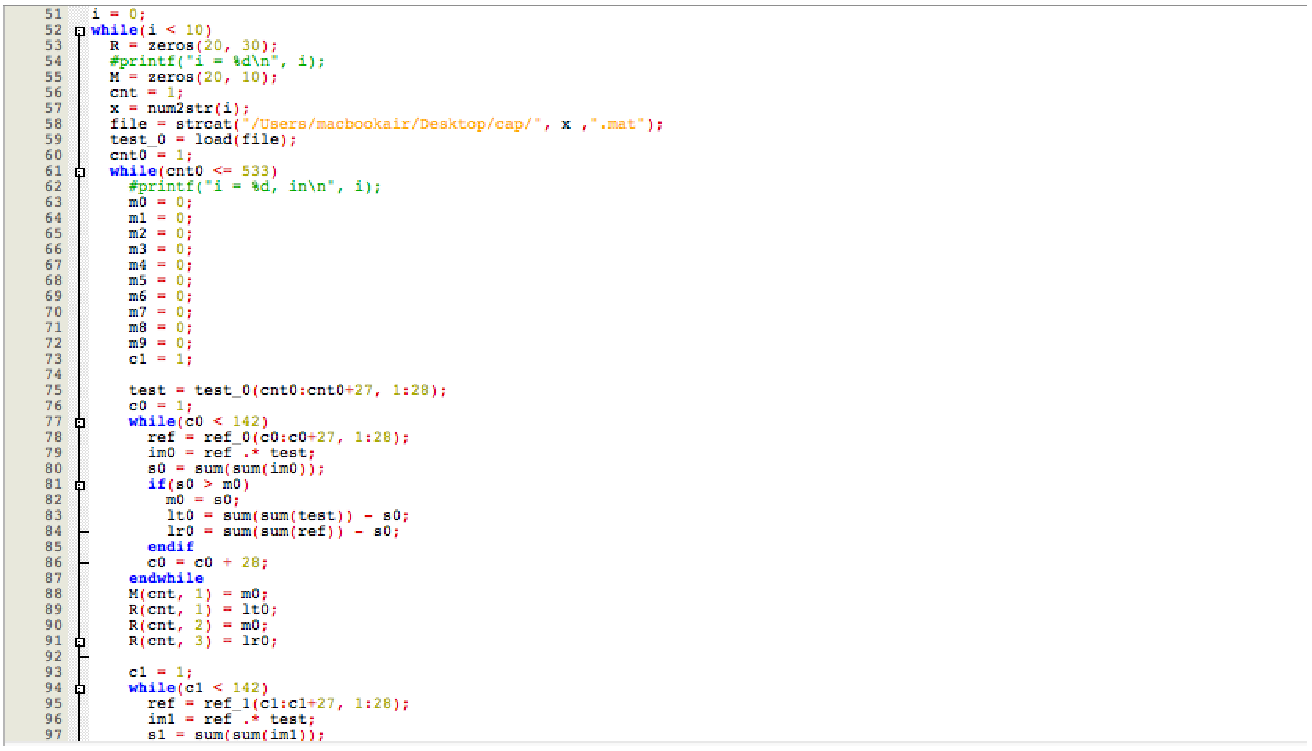


```
    ## compare with 0's
    i = 0;
    while(i < 10)
        m0 = 0;
        y = num2str(i);
        #printf("%s %d\n", y, i);
        file2 = strcat("/Users/macbookair/Desktop/cap/", y ,"_r.mat");
        ref_0 = load(file2);
        c0 = 1;
        while(c0 < 142)
            # get ref image
            ref = ref_0(c0:c0+27, 1:28);

            # left-adjust the ref image
            c00 = 1;
            c01 = 1;
            while(c00<29)
                s = sum(ref(c00, :));
                if(s > 0)
                  R0(c01,:) = [ones(1,s),zeros(1,28-s)];
                  c01 = c01 + 1;
                endif
                c00 = c00 + 1;
            endwhile
            # compare left-adjusted test w/ left-adjusted ref
            m = R0 .* F;
            tot0 = sum(sum(m));
            if(tot0 > m0)
              m0 = tot0;
            endif
            c0 = c0 + 28;
        endwhile
        #printf("i = %d, cnt = %d, m0 = %d\n", i, cnt, m0);
        M0(cnt, i+1) = m0;
        i = i + 1;
    endwhile
    cnt = cnt + 1;
    cnt0 = cnt0 + 28;
  endwhile
  #cnt0 = cnt0 + 28;
  csvwrite(filename2, j, "-append");
  csvwrite(filename2, M0, "-append");
  csvwrite(filename2, "\n\n", "-append");

  j = j + 1;
 endwhile

```

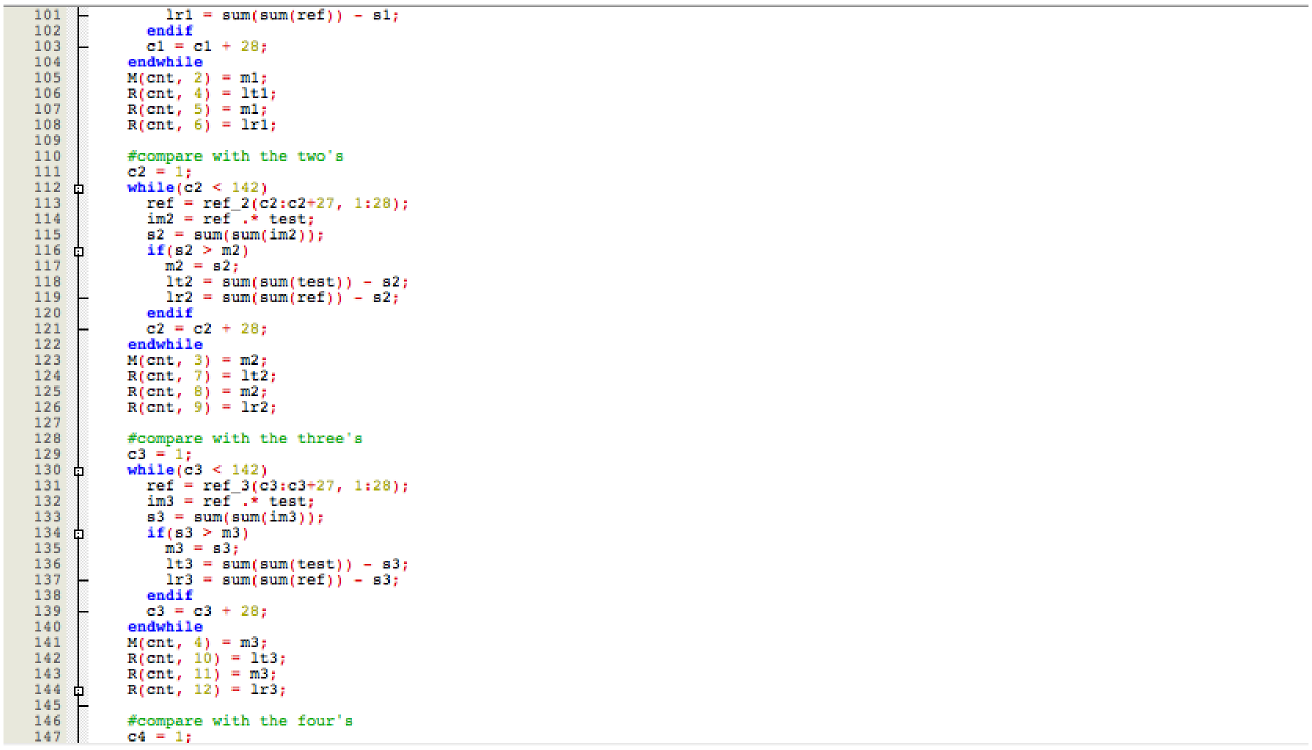
APPENDIX C. Version 4 - Extracting the Reference Set from the MNIST Database



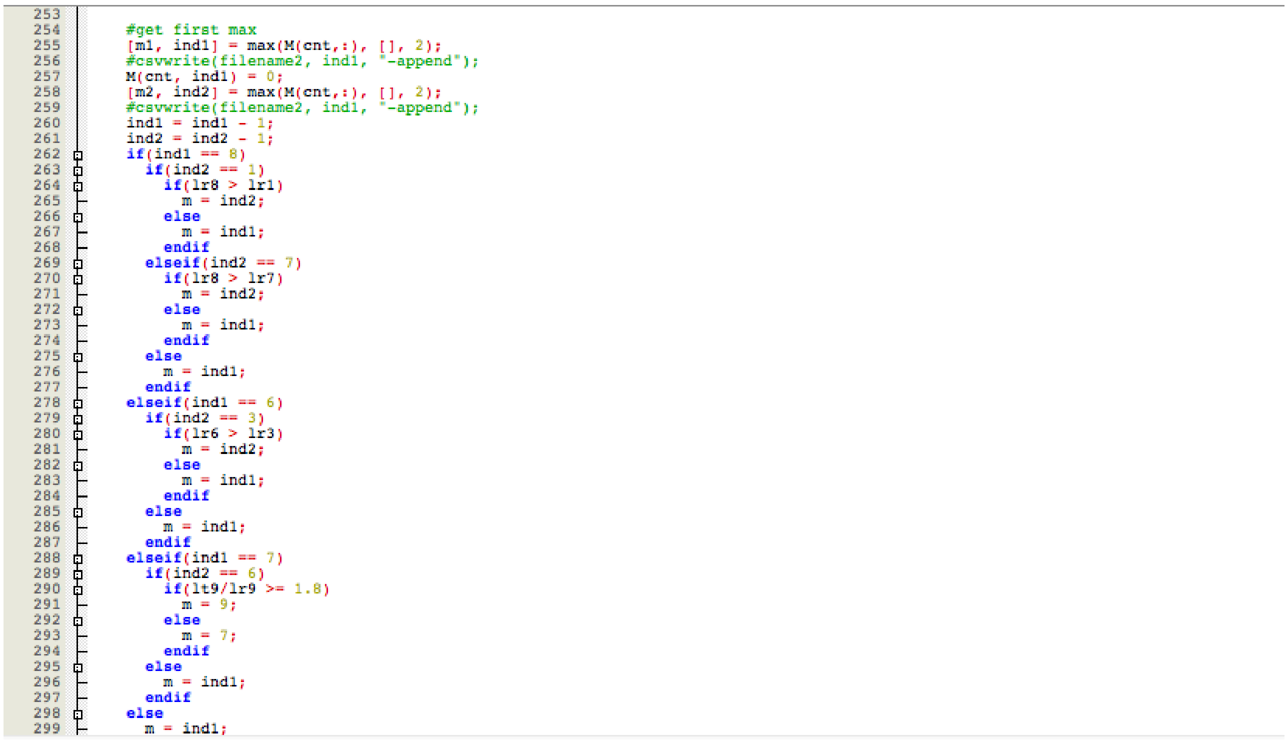
```

## Max overlap

## OPENING FILE

filename = "/Users/macbookair/Desktop/cap/v1.txt";
fid = fopen (filename, "w");
filename2 = "/Users/macbookair/Desktop/cap/rules.csv";
fid2 = fopen (filename2, "w");


## Loading reference set

ref_0 = load('/Users/macbookair/Desktop/cap/reference/0.mat');
ref_1 = load('/Users/macbookair/Desktop/cap/reference/1.mat');
ref_2 = load('/Users/macbookair/Desktop/cap/reference/2.mat');
ref_3 = load('/Users/macbookair/Desktop/cap/reference/3.mat');
ref_4 = load('/Users/macbookair/Desktop/cap/reference/4.mat');
ref_5 = load('/Users/macbookair/Desktop/cap/reference/5.mat');
ref_6 = load('/Users/macbookair/Desktop/cap/reference/6.mat');
ref_7 = load('/Users/macbookair/Desktop/cap/reference/7.mat');
ref_8 = load('/Users/macbookair/Desktop/cap/reference/8.mat');
ref_9 = load('/Users/macbookair/Desktop/cap/reference/9.mat');


m0 = 0;
m1 = 0;
m2 = 0;
m3 = 0;
m4 = 0;
m5 = 0;
m6 = 0;
m7 = 0;
m8 = 0;
m9 = 0;
c1 = 1;

B = zeros(1, 2);

lt = 0;
lf = 0;

## FILE OF ZEROS

#printf("before %d\n", cnt0);
#F = "FILE ZEROS";

```



```
i = 0;
while(i < 10)
  R = zeros(20, 30);
  #printf("i = %d\n", i);
  M = zeros(20, 10);
  cnt = 1;
  x = num2str(i);
  file = strcat("/Users/macbookair/Desktop/cap/", x ,".mat");
  test_0 = load(file);
  cnt0 = 1;
  while(cnt0 <= 533)
    #printf("i = %d, in\n", i);
    m0 = 0;
    m1 = 0;
    m2 = 0;
    m3 = 0;
    m4 = 0;
    m5 = 0;
    m6 = 0;
    m7 = 0;
    m8 = 0;
    m9 = 0;
    c1 = 1;

    test = test_0(cnt0:cnt0+27, 1:28);
    c0 = 1;
    while(c0 < 142)
      ref = ref_0(c0:c0+27, 1:28);
      im0 = ref .* test;
      s0 = sum(sum(im0));
      if(s0 > m0)
        m0 = s0;
        lt0 = sum(sum(test)) - s0;
        lr0 = sum(sum(ref)) - s0;
      endif
      c0 = c0 + 28;
    endwhile
    M(cnt, 1) = m0;
    R(cnt, 1) = lt0;
    R(cnt, 2) = m0;
    R(cnt, 3) = lr0;

    c1 = 1;
    while(c1 < 142)
      ref = ref_1(c1:c1+27, 1:28);
      im1 = ref .* test;
      s1 = sum(sum(im1));
```



```
    lr1 = sum(sum(ref)) - s1;
  endif
  c1 = c1 + 28;
endwhile
M(cnt, 2) = m1;
R(cnt, 4) = lt1;
R(cnt, 5) = m1;
R(cnt, 6) = lr1;

#compare with the two's
c2 = 1;
while(c2 < 142)
  ref = ref_2(c2:c2+27, 1:28);
  im2 = ref .* test;
  s2 = sum(sum(im2));
  if(s2 > m2)
    m2 = s2;
    lt2 = sum(sum(test)) - s2;
    lr2 = sum(sum(ref)) - s2;
  endif
  c2 = c2 + 28;
endwhile
M(cnt, 3) = m2;
R(cnt, 7) = lt2;
R(cnt, 8) = m2;
R(cnt, 9) = lr2;

#compare with the three's
c3 = 1;
while(c3 < 142)
  ref = ref_3(c3:c3+27, 1:28);
  im3 = ref .* test;
  s3 = sum(sum(im3));
  if(s3 > m3)
    m3 = s3;
    lt3 = sum(sum(test)) - s3;
    lr3 = sum(sum(ref)) - s3;
  endif
  c3 = c3 + 28;
endwhile
M(cnt, 4) = m3;
R(cnt, 10) = lt3;
R(cnt, 11) = m3;
R(cnt, 12) = lr3;

#compare with the four's
c4 = 1;
```



```

#get first max
[m1, ind1] = max(M(cnt,:), [], 2);
#csvwrite(filename2, ind1, "-append");
M(cnt, ind1) = 0;
[m2, ind2] = max(M(cnt,:), [], 2);
#csvwrite(filename2, ind1, "-append");
ind1 = ind1 - 1;
ind2 = ind2 - 1;
if(ind1 == 8)
  if(ind2 == 1)
    if(lr8 > lr1)
      m = ind2;
    else
      m = ind1;
    endif
  elseif(ind2 == 7)
    if(lr8 > lr7)
      m = ind2;
    else
      m = ind1;
    endif
  else
    m = ind1;
  endif
elseif(ind1 == 6)
  if(ind2 == 3)
    if(lr6 > lr3)
      m = ind2;
    else
      m = ind1;
    endif
  else
    m = ind1;
  endif
elseif(ind1 == 7)
  if(ind2 == 6)
    if(lt9/lr9 >= 1.8)
      m = 9;
    else
      m = 7;
    endif
  else
    m = ind1;
  endif
else
  m = ind1;
```

APPENDIX D. Matrix of Overlapped and Left Out Pixels of all the 8’s in the Test Set

(Each row represents an image of an eight in the test set)

Left out from the 0	Overlap with 0	Left out from the 8	Left out from the 1	Overlap with 1	Left out from the 8	Left out from the 2	Overlap with 2	Left out from the 8
58	103	142	75	86	12	59	102	106
62	113	110	91	84	14	51	124	84
52	110	113	78	84	14	56	106	102
53	102	143	82	73	25	63	92	56
77	110	135	108	79	19	74	113	95
56	106	117	97	65	33	53	109	99
53	105	140	81	77	21	57	101	123
56	119	126	97	78	20	57	118	90
49	113	132	83	79	19	55	107	101
49	80	143	45	84	12	46	83	141
56	82	152	82	56	42	45	93	131
48	126	119	100	74	24	43	131	77
44	89	134	81	52	46	50	83	141
53	114	131	102	65	31	47	120	104
56	91	132	74	73	25	48	99	109
54	111	112	83	82	16	61	104	104
40	146	77	113	73	25	47	139	69
52	65	158	52	65	33	37	80	128
45	113	110	89	69	29	43	115	93
61	130	93	116	75	23	60	131	77

Left out from the 3	Overlap with 3	Left out from the 8	Left out from the 4	Overlap with 4	Left out from the 8	Left out from the 5	Overlap with 5	Left out from the 8
61	100	68	74	87	50	48	113	83
68	107	61	91	84	53	71	104	92
50	112	49	64	98	39	35	127	69
55	100	75	74	81	56	40	115	81
87	100	68	93	94	43	63	124	72
52	110	32	73	89	48	45	117	79
57	101	41	75	83	54	39	119	77
65	110	32	82	93	44	47	128	68
52	110	44	68	94	43	32	130	66
33	96	58	53	76	61	37	92	104
42	96	79	75	63	66	52	86	110
61	113	48	93	81	53	71	103	93
45	88	73	44	89	48	32	101	95
61	106	69	77	90	39	51	116	80
57	90	78	72	75	62	49	98	98
54	111	57	75	90	47	41	124	72
61	125	36	85	101	29	51	135	61
41	76	92	30	87	50	37	80	116
56	102	59	71	87	50	41	117	79
70	121	21	88	103	34	57	134	62

Left out from the 6	Overlap with 6	Left out from the 8	Left out from the 7	Overlap with 7	Left out from the 8	Left out from the 8	Overlap with 8	Left out from the 8	Left out from the 9	Overlap with 9	Left out from the 9
54	107	107	68	93	33	32	129	58	51	110	28
72	103	111	87	88	66	37	138	28	74	101	37
42	120	94	72	90	62	12	150	37	60	102	36
39	116	98	73	82	72	26	129	58	68	87	36
73	114	100	77	110	44	50	137	50	67	120	18
59	103	111	81	81	71	34	128	64	72	90	48
42	116	98	61	97	29	22	136	51	45	113	25
61	114	100	80	95	59	36	139	48	65	110	28
42	120	94	83	79	73	24	138	49	74	88	50
34	95	119	47	82	35	10	119	68	32	97	41
42	96	118	78	60	92	24	114	78	70	68	70
57	117	97	89	85	69	42	132	34	82	92	46
21	112	102	50	83	48	14	119	68	47	86	37
46	121	93	80	87	67	35	132	60	70	97	41
54	93	121	74	73	81	21	126	40	55	92	46
53	112	102	84	81	45	30	135	52	61	104	34
62	124	90	97	89	63	45	141	25	82	104	34
24	93	121	42	75	77	11	106	81	43	74	64
53	105	109	75	83	69	27	131	25	63	95	43
66	125	89	91	100	52	52	139	48	77	114	24