CAPTCHA image recognition

Zizhao Zhang u5925324

A report submitted for the course
COMP4560 Advanced Computing Project
Supervised by: Ramesh Sankaranarayana
The Australian National University

June 2020

© Zizhao Zhang 2020
Except where otherwise indicated, this report is my own original work.

ZiZhao Zhang
12 June 2020
Acknowledgments

Firstly, I would like to show my deepest gratitude to my supervisor, Dr Ramesh Sankaranarayana. Thank you for your advices and instructions throughout the project. I also appreciate my tutor, Alex Niven, also provides and shares useful ideas on the early stage of the project. Thank you for your help as well.

Finally, I wish to thank my parents and my friends for their sincere encouragement and support during the project. Without their help, I will not finish my project.
Abstract

The CAPTCHA is an automated program to distinguish between humans and machines. One common CAPTCHA type is the text-based CAPTCHA. It consists of several random characters and adds noise to increase the difficulty of recognizing by program. Not only ordinary web pages use text-based CAPTCHA to protect themselves, but the darknet market also uses the CAPTCHA to avoid attacks and supervision by law enforcement officials on the darknet. Compared with the CAPTCHAs on common websites, the CAPTCHAs of the darknet have more types, are replaced more frequently and more difficult to break. The project proposes a series of breaking methods for several types of darknet CAPTCHAs, and achieves high accuracy on some kinds of darknet CAPTCHA.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>v</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Project Scope</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Report Outline</td>
<td>2</td>
</tr>
<tr>
<td>2 Background and Related Work</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Background</td>
<td>3</td>
</tr>
<tr>
<td>2.2 Related work</td>
<td>3</td>
</tr>
<tr>
<td>3 Darknet CAPTCHA</td>
<td>5</td>
</tr>
<tr>
<td>3.1 CAPTCHA without noise</td>
<td>5</td>
</tr>
<tr>
<td>3.2 CAPTCHA with noise</td>
<td>5</td>
</tr>
<tr>
<td>3.3 Maths CAPTCHA</td>
<td>6</td>
</tr>
<tr>
<td>3.4 Other CAPTCHA</td>
<td>7</td>
</tr>
<tr>
<td>3.5 Summary</td>
<td>7</td>
</tr>
<tr>
<td>4 Methodology</td>
<td>9</td>
</tr>
<tr>
<td>4.1 Binarization</td>
<td>9</td>
</tr>
<tr>
<td>4.2 Noise Removal</td>
<td>10</td>
</tr>
<tr>
<td>4.2.1 Neighbor detection</td>
<td>10</td>
</tr>
<tr>
<td>4.2.2 Projection</td>
<td>10</td>
</tr>
<tr>
<td>4.2.3 Erosion</td>
<td>12</td>
</tr>
<tr>
<td>4.3 Segmentation</td>
<td>13</td>
</tr>
<tr>
<td>4.4 Support vector machine</td>
<td>15</td>
</tr>
<tr>
<td>4.5 Other method</td>
<td>16</td>
</tr>
<tr>
<td>5 Result and Analysis</td>
<td>19</td>
</tr>
<tr>
<td>5.1 Overall Accuracy</td>
<td>19</td>
</tr>
<tr>
<td>5.2 Analysis</td>
<td>19</td>
</tr>
<tr>
<td>6 Conclusion</td>
<td>21</td>
</tr>
<tr>
<td>6.1 Conclusion</td>
<td>21</td>
</tr>
<tr>
<td>6.2 Future Work</td>
<td>21</td>
</tr>
</tbody>
</table>
Bibliography 23

Appendices 25

.1 Appendix 1 ................................. 25
.2 Appendix 2 .................................. 25
.3 Appendix 3 ................................. 26
.4 Appendix 4 ................................. 26
List of Figures

3.1 CAPTCHA 1 images ................................................. 5
3.2 CAPTCHA 2 images ................................................ 6
3.3 CAPTCHA 3 images ................................................ 6
3.4 CAPTCHA 4 images ................................................ 6
3.5 CAPTCHA 5 images ................................................ 7
4.1 Before and after OHSU binarization .............................. 10
4.2 Pixels on a straight line ............................................ 10
4.3 Projection in x-axis .................................................. 11
4.4 Damaged character .................................................. 12
4.5 Erosion principle .................................................... 13
4.6 Dilation on hollow CAPTCHA .................................. 13
4.7 Segmentation on CAPTCHA .................................... 14
4.8 Drop fall algorithm ................................................ 14
4.9 Drop fall route ....................................................... 15
4.10 Collection of letter 1 ................................................. 16
4.11 Token and grammar in PLY ...................................... 16
4.12 The cluster contains letter ....................................... 17
5.1 CAPTCHA predicted wrong ...................................... 20
1 Independent Contract 1 .............................................. 27
2 Independent Contract 2 .............................................. 28
3 Program structure ..................................................... 29
4 Readme.md .............................................................. 30
List of Tables

5.1 Over all accuracy on CAPTCHAs. .................................. 19
Chapter 1

Introduction

The word CAPTCHA is the acronym for “Completely Automated Public Turing test to tell Computers and Humans Apart.” The CAPTCHA can generate a series of short tests: these tests are often easier for humans to pass, but it is very difficult for machines to pass [Von Ahn et al., 2003].

Since the CAPTCHA can distinguish between humans and machines, it can establish a security barrier to prevent malicious registration, stealing information and other malicious behaviors such as DDoS attack [Morein et al., 2003].

The CAPTCHA has been widely used in many aspects. For example, Paypal requires people to enter characters in distorted images during the payment process [Stringham, 2015] and the CAPTCHA in the online polling system can help identify whether the machine or person is voting [Basso and Miraglia, 2008]. On the other hand, the CAPTCHA also helps some illegal websites to avoid supervision. An example of the darknet is introduced in section 1.1.

1.1 Motivation

The darknet is a branch of the deep web. An important feature of it is the high level of privacy. It cannot be accessed by ordinary search engines and its visitors are hard to track. The privacy of the darknet has led to many darknet markets being established and conducting illegal transactions [Gayard, 2018]. Therefore, law enforcement officials need to access and supervise information on the suspicious darknet market. A common attack method is DDoS attack. However, many dark web markets have used DDoS protection: Visitors will be led to a guard site with a custom CAPTCHA. Only after the CAPTCHA is passed, the visitor will be linked to the real market [Ball et al., 2019]. Compared with the common CAPTCHA, the CAPTCHA of the darknet is more complicated, hard to collect, and the update frequency is higher. It means that in addition to accuracy, the darknet CAPTCHA also requires cracking efficiency.
1.2 Project Scope

The project focuses on the solving the text-based CAPTCHA. This type of CAPTCHA is often composed of 4 or more random characters. These characters may be rotated or connected together, and random noise will be added to the image to increase the difficulty of cracking. The project will mainly study effective methods for noise removal and character segmentation on different types of darknet CAPTCHAs, including some special types. The methods can also be applied to normal text-based CAPTCHA or general image processing.

1.3 Report Outline

Chapter 1 is the introduction chapter which introduces the origin and application of the CAPTCHA and the motivation to solve the darknet CAPTCHA.

Chapter 2 is the background. In this chapter, CAPTCHA development and CAPTCHA breaking technology are introduced.

Chapter 3 introduces the darknet market CAPTCHA dataset used in this project and analyzes the characteristics on them.

Chapter 4 introduces the methodologies to solve the CAPTCHA step by step.

Chapter 5 shows the predict accuracy on CAPTCHAs. Based on the accuracy, the chapter analyzes the advantage and disadvantage on the methods.

Chapter 6 is the conclusion of the project and the future work.
Chapter 2

Background and Related Work

2.1 Background

In 2003, [Von Ahn et al., 2003] formally proposed the word "CAPTCHA" to describe the concept of CAPTCHA and proposed that CAPTCHAs can protect network security by its abilities to distinguish people and machines. The CAPTCHA at that time consisted of only a few letters and numbers. At that time, the CAPTCHA type is only the text-based CAPTCHA. Afterwards, it was found that the text-based CAPTCHA at that time had disadvantages: the generated image had similar characteristics, and it was easy to find the features and break them automatically [Bursztein et al., 2011]. In addition, too much noise will cause difficulties for people to complete the CAPTCHA. Therefore, other types of CAPTCHAs began to appear, such as image based CAPTCHAs, audio based CAPTCHAs and video based CAPTCHAs [Ragavi and Geetha, 2011]. However, compared with these CAPTCHAs, text-based CAPTCHA are more convenient and welcome. Although the technology for breaking CAPTCHA continues to improve, the generation of reliable CAPTCHAs is also improving. Therefore, the text-based CAPTCHA is still an important type of CAPTCHA and won’t be break easily up until now. The next section will introduce the methods to break text-based CAPTCHA.

2.2 Related work

There are two main types of methods for breaking text-based CAPTCHA: traditional machine learning and neural network. Support vector machine(SVM) is an efficient method on breaking CAPTCHA [Bursztein et al., 2011]. The hard point for machine learning is the segmentation. If the CAPTCHA is difficult to segment, then the inaccurate segmentation will seriously affect the accuracy of machine learning. [Bursztein et al., 2014] proposed an new machine learning algorithm dealing with the segmentation problem and achieve high accuracy at the CAPTCHAs on the major website. Nevertheless, the segmentation technique between characters is still a problem.

The convolutional neural network(CNN) is considered suitable for breaking the CAPTCHA in the past few years. Different from machine learning, CNN doesn’t need to segment the characters. It can consider the connected characters as a whole
Background and Related Work

part to train the model. [Stark et al., 2015] However, CNN costs much more time to
train the model than svm. In addition, the cost of training characters as a whole is
that a large number of training sets are required, otherwise the accuracy will drop
a lot. This project chooses svm as the first choice. One reason is collecting large
amount of training images is not easy on some darknet markets. In addition, CNN
costs much longer time, and the project wants to implement a general segmentation
algorithm.
This chapter will introduce several types of CAPTCHA images and their characteristics in the darknet market. In each section, the characteristics, difficulties, and reasons for selecting this CAPTCHA will be introduced. Most CAPTCHA figures in the project are from the darknet market website. The sources of CAPTCHAs are referenced in Appendix .3. The size of CAPTCHA 1 dataset is 9957, since it comes from the Internet and easy to collect and label. The size of CAPTCHA 2,3,4,5 dataset is 493,235,91,34 respectively. These datasets are collected and labeled from the darknet. CAPTCHA 5 is collected in 2019, and the darknet market regenerate the CAPTCHA, so it is not available now.

### 3.1 CAPTCHA without noise

CAPTCHA 1 are the common 4-letter CAPTCHA images, not from the darknet market. This type of CAPTCHA image has no noise, but there are connections between characters, which have certain requirements for the accuracy of segmentation. By analyzing its recognition results, we can only analyze the accuracy of the segmentation algorithm without considering the effect of preprocessing.

![CAPTCHA 1 images](image)

**Figure 3.1:** CAPTCHA 1 images

### 3.2 CAPTCHA with noise

CAPTCHA 2 are the CAPTCHA images from the Yellow Brick Market. It is characterized by particularly dense noise. A large number of random points and lines appear evenly in the picture and are close to the characters, so it is difficult to separate them perfectly. In addition, the characters contain numbers, uppercase letters and
lowercase letters, too many types of characters will increase the difficulty of machine learning. The advantage is that the proportion of tightly connected characters is low.

![Figure 3.2: CAPTCHA 2 images](image)

CAPTCHA 3 are the CAPTCHA images from Cypher Market. Compared with CAPTCHA 2, the distance between characters and the distance between characters and random curves are closer, so it is more difficult to remove noise. Some random curves have the same color as characters, but some are completely different, which makes binarization difficult. The thickness of these CAPTCHAs also varies greatly. After preprocessing with relatively thick characters, thin characters will be damaged.

![Figure 3.3: CAPTCHA 3 images](image)

### 3.3 Maths CAPTCHA

The Maths CAPTCHA not only requires you to recognize each character, but also the result of these characters [Hernandez-Castro and Ribagorda, 2010]. CAPTCHA 4 is a relatively simple type of semantic CAPTCHA derived from Verified Forum in darknet. When each character in "4+3=" is recognized, the CAPTCHA can only be passed after answer 7 is calculated. A lexer parser would be constructed to solve the CAPTCHA. This CAPTCHA doesn’t contain noise, but actually on the darknet market, Maths CAPTCHA is always combined with noise, such like the noise in 3.2 and 3.4.

![Figure 3.4: CAPTCHA 4 images](image)
3.4 Other CAPTCHA

CAPTCHA 5 are from Empire Market. A large number of disturbing background colors make it impossible for the CAPTCHA to separate characters using conventional methods. Its length is not fixed. Such CAPTCHAs require special preprocessing which is introduced in the next chapter.

Figure 3.5: CAPTCHA 5 images

3.5 Summary

This chapter introduces some darknet CAPTCHA types, features and difficulties, and shows the corresponding CAPTCHA image. The next chapter will explain how to deal with these CAPTCHAs.
In order to improve the difficulty of recognizing CAPTCHAs to distinguish computers from humans, many CAPTCHAs add some noise such like random points and random dots on the characters to interfere with the computer recognition algorithms. These random noise will badly affect the recognition accuracy. Several methods to remove the noise are introduced in this chapter. Afterwards, the segmentation algorithm is necessary to segment the connected characters. Preprocessing for a specific type of verification code is relatively simple, because it can be processed according to the characteristics of letters. This chapter will try to use a relatively general method to deal with different styles of characters and noise.

There are a few type of captchas that require special processing, such as removing borders and setting the non-white background color to white. These methods will not be included in this chapter.

4.1 Binarization

The usual binarization method is to set a threshold: if the gray value of the pixel is greater than the threshold, then the value of this pixel is set to 255 (white) Otherwise, the value is set to 0 (black). However, the color and brightness of the CAPTCHAs are different, and the most suitable threshold for different CAPTCHA varies greatly. In CAPTCHA image, the color of the character and the color of the noise usually have a certain degree of distinction to facilitate human recognition. Otsu’s method [Otsu, 1979] is applied to the CAPTCHA binarization. In this algorithm, the threshold divides the pixels into two groups that are greater than the threshold and less than the threshold according to the gray value, and the optimal threshold minimizes the gray value variance within the group. The main color in 4.1 is green, so the threshold OSTU method returns is close to the gray value of green. The green noise line can’t removed in this section, but lines of other colors have obviously faded or disappeared. After binarization, all the pixels is set to 0 or 255.
4.2 Noise Removal

In general, there are two main types of noise: random lines or curves, random pixels or blocks of pixels. This section will introduce several methods to remove them.

4.2.1 Neighbor detection

In CAPTCHAs with sparse noise points, median filter or Gaussian filter can remove noise points. When the noise is dense, the filter can’t work. To determine which pixels are noise pixels, the algorithm is to check the 8 pixels around each black pixel in the image. If less than or equal to two pixels are black, this pixel will be considered as the noise pixel and set to white. Meanwhile, 8 pixels around this pixel should be updated: the sum of surrounding black pixels is reduced by 1 and it needs to check whether it will be set to white.

The algorithm stops if all black pixels have been checked and no new pixels turn white. This method may cause minor damage to some very thin CAPTCHA. When applied to the darknet CAPTCHA, it can be assumed to be safe: almost no darknet CAPTCHA are pretty thin. The thin noise lines can also be removed. On the straight line in Figure 4.2, there are no more than two black pixels around each black pixel, so they will all become white.

4.2.2 Projection

Eliminated the interference of noise points, the next goal is to remove the line between characters. Let the horizontal direction be the x-axis and the vertical direction be the y-axis. The projection in the x-axis direction of the image helps. [Huang et al., 2008] uses projection method to help with the segmentation. Figure 4.3 shows the histogram.
of the CAPTCHA after removing small noise points. The value of each line at the coordinate $x$ of the histogram represents the number of black pixels in the picture whose horizontal coordinate is $x$. It is obvious to see that the areas of the connection between the characters in the CAPTCHA image are at the trough of the histogram, because the black pixel source of these areas is only from noise, so it is much less than other areas.

![Figure 4.3: Projection in x-axis](image)

Regardless of the pixel of the character, in the ideal case, the projection values of multiple lines from left to right on the x axis are almost the same. This is because the value of a line projected on each coordinate $x$ is equal to the thickness of the line on the $x$ coordinate. As long as the thickness of this line does not change much, its projection on the x-axis is almost equal, and the same is true for multiple lines. However, this inference will be affected by the noise area consisting of a small number of pixels around the line. Noise areas that are completely separate from the line are easy to remove: calculate the sum of the pixels of each individual area. If they are less than the threshold, then this area cannot be a character, and every pixel will be turned white. The size of the threshold is related to the character size. In this verification code, the threshold is set to 80.

Next, in order to determine the position of the connection of the character, a relatively safe method is taken: take 5 adjacent points on the x-axis and get their projection values. If these 5 values meet the following 3 conditions, they will be regarded as the connection between characters, and they will all be set to white.

1. The maximum value of them is no more than the threshold.
2. The range of them is no more than the threshold.
3. The variance of them is no more than the threshold.

The first threshold is related to the thickness of the character. If the threshold is set too low, the character itself will be damaged. Figure 4.4 shows that the H is
Methodology

damaged with small threshold. The second and third conditions are to prevent parts outside the character body from being mistaken for lines. For example, the part of "r" that deviates from the vertical line. The sum of his pixel values is not large, so condition 1 does not necessarily work. But unlike the line, his range and variance are relatively large. Without these two thresholds, this part will be mistaken for noise.

The general algorithm of noise removal is:
Step 1. Check the pixels’ neighborhood;
Step 2. Turn the pixels with no more than 2 neighbors to white;
Step 3. Update the surrounding pixels.
Step 4. Remove the individual areas whose sum of pixels is less than threshold;
Step 5. Check if any new pixel and area can be removed. If yes, repeat step 1-4. Otherwise the algorithm terminate.

4.2.3 Erosion

The previous steps will still leave some lines intersecting the characters. These lines and characters have formed a whole and cannot be removed by the previous method. Erosion is the one of the method to removing thinner lines [Liu et al., 2012].

Figure 4.5 shows a 3*3 erosion kernel. The red circle is the original point. Its 8 neighbors are all black pixels. Check each pixel in the left image. The pixel remains black only if its 8 neighbors are all black under the 3*3 erosion kernel. When applying the kernel on the image, the pixels on the red line should be eroded and turn white. The eroded pixel is actually the outer contour of the black area, and the 3*3 kernel has the effect of eroding a layer of outer contour.

Dilation is the opposite operation of erosion. The dilation operation of the same scale as 3*3 erosion is equivalent to adding a layer of black pixels to the outer contour of the image. When we apply erosion then dilation in Figure 4.5, the black pixels on the left line will not change back to black due to dilation, since there is no black pixel as the original point. The right part of black pixels will return to the original shape.
After this operation, the black pixels on the left line become white, but the main area is not affected at all. This is the basic principle of erosion. In addition, the sequence of dilation and then erosion can fill with hollow verification codes; the reason is similar. Figure 4.6 is an example.

Finally, the edge of the CAPTCHA image after preprocessing is often uneven due to the processing of pixels. 3x3 median filter is used for the final smoothing will be better for the following step.

### 4.3 Segmentation

Figure 4.7 shows the CAPTCHA image after the preprocessing steps. Based on the projection on x-axis, the characters have been separated to 5 areas between the red line. The next step is to segment the connection between the area "OC". The basic idea is to segment the characters where the projection value is minimum with vertical line, as the connection of characters is often the place with the fewest pixels. The idea works in this case, however, when the characters are slightly rotated or twisted, the
Methodology

pixels at the connection are not necessarily the least positions, and the vertical lines cannot accurately cut the connection.

Figure 4.7: Segmentation on CAPTCHA

[Congedo et al., 1995] proposed a drop fall segmentation method. This algorithm treats the black pixel as an obstacle and the initial point of the cut as a water drop. From the starting point, the cut marks completely imitate the trajectory of the water drop falling along the obstacle. The specific algorithm is the following step:

In Figure 4.8, s is the current position of the water drop and n1-n5 are the pixels to be detected in this period of step. n=0 means black pixels and n=255 means white.

<table>
<thead>
<tr>
<th>n1</th>
<th>s</th>
<th>n5</th>
</tr>
</thead>
<tbody>
<tr>
<td>n2</td>
<td>n3</td>
<td>n4</td>
</tr>
</tbody>
</table>

Figure 4.8: Drop fall algorithm

Step 1. Check whether n3=255. If yes, next position is n3. Otherwise go to step 2.
Step 2. Check whether n2=255. If yes, next position is n2. Otherwise go to step 3.
Step 3. Check whether n4=255. If yes, next position is n4. Otherwise go to step 4.
Step 4. Check whether n5=255. If yes, next position is n5. Otherwise go to step 5.
Step 5. Check whether n1=255. If yes, next position is n1. Otherwise go to step 6.
Step 6. If n1-n5 are all black, go to n3. If the water drop flow left and right repeatly in steps 4 and 5, then the drop go to n3 at either boundary in step 4 and 5.

The step follows the route of the drop. For instance, step 1 means if no obstcale below the drop, the water drop will fall down due to gravity. Step 2 and 3 simulate the water drop falling along the slope.

This algorithm has high requirements on the initial point. When the initial point is not suitable, the split route will run to the leftmost or rightmost edges. The algorithm itself requires that there’s a black pixel on the right side of initial points. The projected value is used again to determine the appropriate coordinate x of the
initial point. The connection between characters is where the projected value changes the most, because this area is transferred from one character to another. For each coordinate $x$, let $P = (\text{projection}(x) - \text{projection}(x-1)) + (\text{projection}(x) - \text{projection}(x+1))$. The coordinate $x$ with minimum $P$ is chosen. If it is finally detected that the end point of the route is too far left or right, this segmentation algorithm will not be suitable, and the common segmentation method will be used instead. 4.9 is a segmentation route of the algorithm.

4.4 Support vector machine

Support vector machine is a kind of linear classifier that performs classification on data by supervised learning. In traditional machine learning methods, support vector machine algorithm is very suitable for the classification of CAPTCHA characters, and the accuracy rate is relatively high [Bostik and Klecka, 2018]. Neural Network is the popular method to train CAPTCHA, but its disadvantage is that it requires a large number of training sets and training time to ensure accuracy. The special environment of the darknet makes it difficult to obtain a large number of datasets, and some darknet markets update CAPTCHAs frequently, so the cracking efficiency is an important factor. For these reasons, support vector machine is the preferred training method for this project.

In the last section, each divided character is saved in a separate folder, then reshape to the same size, as shown in the Figure 4.10. The feature is the value of pixels in the image. The value of black pixel is set to 1, and the value of white is set to 0. Then, from top to bottom, from left to right, collecting the value of all pixels and adding them to a list with the label. For example, a image of letter c consists of $3 \times 3$ black pixels, the label is c and the list is $[1,1,1,1,1,1,1,1,1]$. support vector machine is good at processing high-dimensional samples, and the feature of the image are the values of many of pixels, which form a higher dimension. This model can be directly used to predict the characters of the generated verification codes on the darknet market, and can also predict the CAPTCHAs of other markets with similar shapes and sizes, although the accuracy rate will decrease.
4.5 Other method

To solve the Maths CAPTCHA like Figure 3.4 in the darknet, a lexer parser is needed to analyze each character then calculate. This project uses Python Lex-Yacc tool [Beazley, 2001] to help solve the Maths CAPTCHA. Each character in the input is considered as a token, and the compiler will take specific actions and calculate when the symbol stack receive the input token based on the pre-set grammar. When all tokens are read and the stack is empty, calculation is over. For this case, the token "+", "-", "\(\)", "/", "(\(\)\)" and the number is enough to solve the CAPTCHA. Figure 4.11 is the basic token and grammar from the PLY tool.

The CAPTCHA in Figure 3.5 is really challenging. A large number of color lines in the picture are noise, while the really useful CAPTCHA characters are only in a small area. Binarization cannot extract characters at all, let alone the subsequent processing. Humans can barely recognize characters because the color of the characters and the surrounding colors have a certain color difference, and they are connected in coordinates. Image segmentation [Dhanachandra et al., 2015] is a useful method to find out those connected area with same colors.

In a color image, each pixel has a position coordinate (x, y) and \((r, g, b)\) value, where \(r, g, b\) are in the range of 0-255. [Dhanachandra et al., 2015] uses k-means algorithm to divide the color pixels to several clusters. A modification is applied on the CAPTCHA: the distance between pixels with value \((x1, y1),(r1,g1,b1)\) and \((x2,y2),(r2,g2,b2)\) is:

\[
m = \sqrt{(x1 - x2)^2 + (y1 - y2)^2} + \sqrt{(r1 - r2)^2 + (g1 - g2)^2} + (b1 - b2)^2
\]

where \(m\) is the variable to control the weight between coordinate distance and
§4.5 Other method

color distance. The greater the value of m, the greater the proportion of coordinate distance. What’s more, kmeans++ [Arthur and Vassilvitskii, 2006] is applied to reduce random factors of kmeans algorithm. Figure 4.12 shows a cluster when k=12 and m=0.05. Unfortunately, although all the clusters with letter is segmented, it is not easy to locate them among the clusters and image area, so the accuracy still remains low.

Figure 4.12: The cluster contains letter
Chapter 5

Result and Analysis

This chapter will show the recognition accuracy of the CAPTCHA dataset in Chapter 3 and discuss the results. Pytesseract OCR (referenced in Appendix 3) is an optical character recognition project to recognize the characters. Its accuracy in recognizing standard characters without noise is very high. Therefore, comparing its accuracy can be used to judge whether the effect of preprocess on noise is good or not.

5.1 Overall Accuracy

Table 5.1 shows the overall accuracy on the CAPTCHA datasets. The letter accuracy is the value (correct letters/total letters). For example, if the CAPTCHA has 4 letters, than the total letter are 4 times the number of CAPTCHA. The image accuracy is (correct CAPTCHAs/total CAPTCHAs), where correct CAPTCHA requires all 4 letters are correct. CAPTCHA 4 is 100% accuracy on svm indicates the lexer parser works well as the character is easy to segment. OCR can’t solve Maths CAPTCHA so the area is empty. Although the image segmentation method works on CAPTCHA 5, but there is no general method to locate the position of characters, so it is not included in the table.

5.2 Analysis

Overall, the svm accuracy is much better than OCR accuracy. In CAPTCHA 2, the svm image accuracy is 20% higher than letter accuracy. Because only 6 letters are all

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>CAPTCHA 1</th>
<th>CAPTCHA 2</th>
<th>CAPTCHA 3</th>
<th>CAPTCHA 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM image accuracy</td>
<td>84.5%</td>
<td>61.2%</td>
<td>3.2%</td>
<td>100%</td>
</tr>
<tr>
<td>SVM letter accuracy</td>
<td>93.3%</td>
<td>84.7%</td>
<td>33.1%</td>
<td>100%</td>
</tr>
<tr>
<td>OCR image accuracy</td>
<td>69.6%</td>
<td>48.8%</td>
<td>0%</td>
<td>-</td>
</tr>
<tr>
<td>OCR letter accuracy</td>
<td>72.1%</td>
<td>62.7%</td>
<td>36%</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 5.1: Over all accuracy on CAPTCHAs.
correct, this image can be recognized correctly. This condition is harder than 4-digit CAPTCHA. Figure 5.1 shows a image of CAPTCHA 2 predicted wrong. The noise removal is good, but the connection between N and M is too close. In CAPTCHA 3, the situation becomes worse. This type of CAPTCHA often connects all three or four characters together, making it difficult to correctly select three accurate split points at the same time. That’s why the accuracy in CAPTCHA 3 is much lower.

![Figure 5.1: CAPTCHA predicted wrong](image)

It can be concluded that the segmentation method of this project performs well when there are not many character connections. CAPTCHA 1 and CAPTCHA 2 both have some connections between 2 letters and the accuracy is good. It means slight connection can be segmented properly, but the method can’t be applied on more than 3-4 connections. The connected characters should be trained as a whole via svm, which greatly increases the amount of calculation. In such case, Neural Network could be a better choice.
Chapter 6

Conclusion

6.1 Conclusion

This project first introduces the origin of the CAPTCHA and its application on the network. While the CAPTCHA provides protection to the website, the criminals also use the CAPTCHA to protect themselves from attacks on the darknet. Then the project proposed a series of methods and steps for cracking conventional text-based CAPTCHA by analyzing the characteristics of several types of darknet CAPTCHA. Finally, in the accuracy test, it is found that this method does not work well when multiple characters that are tightly connected. Finally, in the future work, some new research directions of darknet CAPTCHA are proposed.

6.2 Future Work

One important future work is to solve CAPTCHA 5. One current idea is to select a rectangular frame in the image, and the size ratio is approximately similar to the characters of this type of CAPTCHA image. Traverse all possible positions of this rectangular frame. If the proportion of pixels in a box at a certain position exceeds a threshold, then the pixels in the frame may be a character. Record this position and the shape of the pixels in the frame. Afterwards, all the shapes recorded in the frames will be further analyzed.

Improve the training efficiency of svm is also a research direction in the future. Currently we extract the features by reshaping the image to a small shape as we can’t train the data with hundreds of pixels multiply hundreds of pixels. The reshape method may lose information and there are still many pixels left. An idea is to sum the pixels in each row and column and use these values as the features. Normally, the sum of different characters, rows and columns will not be totally same. But the centrally symmetrical character is an exception: the sum of rows and columns is totally same on "/" and "\" as they are symmetrical. Therefore, how to extract the feature from the image efficiently without losing accuracy is a very interesting problem.
**Bibliography**


**Ball, M.; Broadhurst, R.; Niven, A.; and Trivedi, H., 2019.** Data capture and analysis of darknet markets. *Available at SSRN 3344936*, (2019). (cited on page 1)


**Bursztein, E.; Aigrain, J.; Moscicki, A.; and Mitchell, J. C., 2014.** The end is nigh: Generic solving of text-based captchas. In *8th {USENIX} Workshop on Offensive Technologies ({WOOT} 14)*. (cited on page 3)


**Gayard, L., 2018.** *Darknet: Geopolitics and Uses*. John Wiley & Sons. (cited on page 1)


**Huang, S.-Y.; Lee, Y.-K.; Bell, G.; and Ou, Z.-H., 2008.** A projection-based segmentation algorithm for breaking msn and yahoo captchas. In *ICSIE’08: Proceedings of*


Appendices

.1 Appendix 1

Project Description: This project is mainly used for text-based CAPTCHA recognition. The project can be used to effectively and quickly pass the darknet market CAPTCHA test, which is convenient for regularly collecting and supervising darknet data by law enforcement officials.

This project summarizes the characteristics of darknet market CAPTCHA, and analyzes the application of image preprocessing, image segmentation, and SVM classifier based on these characteristics. These methods are mainly designed based on the characteristics of the CAPTCHA on the darknet market, so it will be more accurate and effective for cracking the CAPTCHA on the market. However, the project can also be applied to general CAPTCHA. The test of the usual 4-digit CAPTCHA also achieves a high accuracy rate.

Project outcome:

- Effective verification code image processing and training methods.
- 93% accuracy on common 4-digit CAPTCHA.
- 84% accuracy on darknet noise CAPTCHA.
- Application of lexer parser on Math CAPTCHA.
- Application of image segmentation method in complex darknet CAPTCHA.

.2 Appendix 2

Independent Study Contract Figure 1 and Figure 2
.3 Appendix 3

Figure 3 is the program structure.

OCR.py, svm.py, drop.py, denoise.py are my work. In OCR.py, installation of pytesseract OCR project is needed, install from https://pypi.org/project/pytesseract/
calclex.py and yacc.py are outer programs from http://www.dabeaz.com/ply/ply.html. They are not my work.

The source of CAPTCHA dataset:
CAPTCHA1: CAPTCHA from Kaggle. https://www.kaggle.com/genesis16/captcha-4-letter
CAPTCHA5: Empire market. http://i5kjii2y2jumlye6etmouksvdhech357urmj4txctrneedl4vfkfbsqd.onion (not available now)

More details are in readme.md.

The Hardware platform is: Windows 10 laptop, Intel(R) Core(TM) i7-8565U CPU @1.80GHz 1.99GHz RAM: 16.0GB 64 bit operation system. Software is Pycharm 2019.3.3.

.4 Appendix 4

Figure 4 is the Readme.md:
INDEPENDENT STUDY CONTRACT
PROJECTS

Note: Enrolment is subject to approval by the course convenor

SECTION A (Students and Supervisors)

UniiD: __u5925324__________________

SURNAME: __Zhang_________________ FIRST NAMES: __Zizhao_________________

PROJECT SUPERVISOR (may be external): __Ramesh Sankaranarayana_________________

FORMAL SUPERVISOR (if different, must be an RSSCS academic): __Ramesh Sankaranarayana_________________

COURSE CODE, TITLE AND UNITS: __COMP4560, 12 units_________________

COMMENCING SEMESTER: □ S1 □ S2 YEAR: _____ Two-semester project (12u courses only): ✔

PROJECT TITLE:

CAPTCHA Image recognition

LEARNING OBJECTIVES:

1. Implement a CAPTCHA image recognition project
2. Be able to recognize noisy CAPTCHA with various interference
3. Be able to recognize semantic CAPTCHA and calculate the answer

PROJECT DESCRIPTION:

A CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart") is a test used in computing to determine whether a user is human. CAPTCHAs are an important tool online as their ability to differentiate between human users and automated systems can assist in restricting access to content, creating secure barriers that can stop information retrieval and other malicious actions such as DDOSing. While widely used, these systems are not infallible, as shown by researchers, developers and hackers ("Bypassing Captcha Like a Boss", 2018; Sivakorn, Polakis & Keromytis, 2016).

Breaking the CAPTCHA defence of the systems allows for an increase in efficiency in collecting data from the websites. This project will focus on recognizing the noisy CAPTCHA with various interference. Another kind of CAPTCHA is semantic CAPTCHA. We will recognize of symbols and using the symbols to perform calculations. We may find and recognize more types of CAPTCHA in the further work.
## ASSESSMENT
(as per the project course's rules web page, with any differences noted below).

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: style: _____________</td>
<td>(min 45, def 60)</td>
<td></td>
<td>examiner</td>
</tr>
<tr>
<td>(e.g. research report, software description...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artefact: kind: _____________</td>
<td>(max 45, def 30)</td>
<td></td>
<td>supervisor</td>
</tr>
<tr>
<td>(e.g. software, user interface, robot...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation:</td>
<td>(10)</td>
<td></td>
<td>course convenor</td>
</tr>
</tbody>
</table>

## MEETING DATES (IF KNOWN):

**STUDENT DECLARATION:** I agree to fulfil the above defined contract:

\[ \text{Signature} \quad \text{27/07/2019} \]

<table>
<thead>
<tr>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
</table>

**SECTION B (Supervisor):**

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project. I nominate the following examiner, and have obtained their consent to review the report (via signature below or attached email)

<table>
<thead>
<tr>
<th>Signature</th>
<th>Date</th>
</tr>
</thead>
</table>

**Examiner:**

Name: ___________________________ Signature: ___________________________

*(Nominated examiners may be subject to change on request by the supervisor or course convenor)*

**REQUIRED DEPARTMENT RESOURCES:**

\[ \text{Research School of Computer Science} \]

*Form updated Jun 2018*

---

**Figure 2: Independent Contract 2**
Figure 3: Program structure
Figure 4: Readme.md