A Security Analysis of Automated Chinese Turing Tests

Abdalnaser Algwil
School of Computing and Communications
Lancaster University, UK
a.algwil@lancaster.ac.uk

Dan Cireșan
IDSIA (SUPSI-USI)
Switzerland
dan@idsia.ch

Beibei Liu
South China University of Technology
China
eebbliu@scut.edu.cn

Jeff Yan*
School of Computing and Communications
Lancaster University, UK
jeff.yan@lancaster.ac.uk

ABSTRACT

Text-based Captchas have been widely used to deter misuse of services on the Internet. However, many designs have been broken. It is intellectually interesting and practically relevant to look for alternative designs, which are currently a topic of active research. We motivate the study of Chinese Captchas as an interesting alternative design – counterintuitively, it is possible to design Chinese Captchas that are universally usable, even to those who have never studied Chinese language. More importantly, we ask a fundamental question: is the segmentation-resistance principle established for Roman-character based Captchas applicable to Chinese based designs? With deep learning techniques, we offer the first evidence that computers do recognize individual Chinese characters well, regardless of distortion levels. This suggests that many real-world Chinese schemes are insecure, in contrast to common beliefs. Our result offers an essential guideline to the design of secure Chinese Captchas, and it is also applicable to Captchas using other large-alphabet languages such as Japanese.

Keywords

Chinese Captcha; Convolutional Neural Network; Deep Neural Network; security; usability

1. INTRODUCTION

Captchas have been widely deployed by websites for preventing malicious bot programs from misusing Internet resources or services. The most commonly used Captchas are text-based, in which challenge appears as an image of text and users are required to recognize and retype the text. To resist attacks from auto-recognition programs, the text in the image has to be distorted and camouflaged. However, too sophisticated distortion may also degrade the readability for humans. It is thus critical for a Captcha scheme to be well balanced between usability and security [46].

Many text Captchas have been broken, e.g. [30,44,45], and it is an active research area to explore alternative designs - recent efforts include image recognition Captchas [28], motion Captchas [11], and game Captchas [2]. However, these alternative designs have been either broken [37,43,48] or not widely deployed.

Traditional wisdom is that text Captchas should use Roman characters, which are recognizable universally and thus avoid localization issues – you do not have to design a different Captcha for a different natural language. In this paper, we will show that counter-intuitively, it is possible to design Chinese Captchas that are universally usable, even to those who have never studied Chinese language. On the other hand, given hundreds of millions of Chinese Internet users, there is a huge customer base for Chinese Captchas. Some major websites in China have used such Captchas already, for example, QQ.com (ranked by Alexa.com as Top 8 website in the world and Top 2 in China), Tianya.cn (ranked by Alexa.com as Top 60 website in the world and Top 11 in China), China.com (ranked by Alexa.com as Top 112 website in the world and top 19 in China) and Renren.com (ranked by Alexa.com as Top 1,292 website in the world and Top 187 in China). Therefore, we argue that Chinese Captchas can be an interesting alternative design, and deserve more attention in the research community than now.

Early research suggested that computers are good at recognizing single Roman characters, even if the characters have undergone sophisticated distortions [19]. This has led to an important principle: the robustness of text Captchas should rely on the difficulty of finding where the character is (segmentation), rather than which character it is (recognition). This principle, often referred to as the segmentation-resistance principle, has become the footstone for designing text Captchas.

However, a fundamental open question remains: is the segmentation-resistance principle established for Captchas using Roman characters applicable to Chinese based designs? The former have a small alphabet, but the latter a

*Corresponding author. Beibei Liu participated in this work as a postdoctoral researcher of Jeff Yan.

ACSAC '16, December 05-09, 2016, Los Angeles, CA, USA
© 2016 ACM. ISBN 978-1-4503-4771-6/16/12…$15.00
DOI: http://dx.doi.org/10.1145/2991079.2991083

much larger one. The empirical work establishing this principle does not offer any evidence to either support or disapprove the extension. In the wild, most, if not all, deployed Chinese Captchas assume either explicitly or implicitly that it is hard for computers to recognize distorted single characters from a large character set like Chinese, and the security of such Captchas relies on this unproven hardness.

In this paper, with deep learning techniques, we will offer the first evidence that computers do recognize individual Chinese characters with a high accuracy, regardless of distortion levels. However, not all machine learning techniques do well with this task. Our result suggests that many real-world schemes, including those designed for Chinese communities only, and those designed for universal usability, are insecure, in contrast to common beliefs. This result is also relevant to Captcha designs based on other large-alphabet languages such as Japanese. Overall, our work also contributes to the understanding of what can computers do well and what they cannot, and to the understanding of better Captcha design.

We review related work in Section 2. Section 3 provides a survey and an analysis of real-world Chinese Captchas. In Section 4, we show that the popular Convolutional Neural Network Le Net 5 [34] do not do well in recognizing our synthesized datasets of distorted Chinese characters, but a newer Deep Neural Network (DNN) [24] achieves impressive performance, instead. In Section 5, we show that the DNN also achieve similarly good performances on a dataset collected from a deployed, representative Chinese Captcha. In Section 6, we compare our experiments with handwritten Chinese recognition. We argue that handwritten Chinese and our synthesized datasets might appear to be slightly different, but they are significantly different image objects for computers. We conclude in Section 7 by summarizing the lessons that we have learned.

2. RELATED WORK

2.1 Text-based Captchas

By far, text-based, particularly Roman characters based Captcha, is still the major type worldwide adopted. Many studies explored how to design text Captchas properly, and most of the studies built on attacks on existing Captchas.

Early efforts to break text Captchas include [20, 38, 39]. In [38], Mori and Malik used sophisticated object recognition algorithms to break EZ-Gimpy and Gimpy, two early simple Captchas, with a success of 92% and 33% respectively. In [39], Moy et al. increased the success rate on EZ-Gimpy to 99% with their distortion estimation techniques. They also achieved a success rate of 78% on the 4-characters Gimpy-r scheme. In [20], Chellapilla and Simard worked on a variety of text Captchas taken from the Internet and reported success from 4.89% to 66.2%.

In contrast to the early works that relied on sophisticated computer vision or machine learning algorithms, Yan et al. proposed an attack that only used naïve pixel counting method and simple pattern recognition algorithms [44]. Their method achieved almost 100% success rate on a number of Captchas. The same authors have subsequently reported successful attacks on a series of Captchas designed and deployed by Microsoft, Yahoo and Google [45]. Their novel character segmentation techniques have been proven effective and of general value. In [26], they also broke a novel text Captcha deployed by the Megaupload website which featured a new anti-segmentation technique.

Bursztein et al. [18] developed an automated tool called Decaptcha that was able to break 13 out of 15 most widely acknowledged Captcha schemes. They reported $1\% - 10\%$ success rate on Baidu and Skyrock, $10 - 24\%$ on CNN and Digg, $25 - 49\%$ on eBay, Reddit, Slashdot and Wikipedia, and 50% or higher on Authorize, Blizzard, Captcha.net, Megaupload and NIH. Decaptcha failed (0% success rate) to break the Google and reCaptcha schemes. However, the two schemes were later broken by Yan's team [27].

Lately, Gao et al. [29] performed the first security analysis of hollow Captcha, one of the latest designs. A novel attack using a Convolutional Neural Network with a graph search algorithm was implemented against five hollow Captchas deployed by Yahoo, Tencent, Sina, CmPay and Baidu. Their attack achieved a success rate from 36% on Yahoo to 89% on Tencent.

Most recently, the joint team of Gao and Yan published in 2016 a surprisingly simple, low-cost, generic but powerful attack that breaks a wide variety of Roman-character based Captcha designs [30].

2.2 Alternative Captchas

Along with text Captcha research, there are many efforts to develop alternative Captcha schemes. The underlying principle is to exploit other human abilities which are believed more difficult for computers to defeat, e.g. human's capability of understanding image semantics, or instructions and fulfil a specified task accordingly.

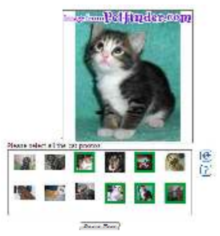
Microsoft ASIRRA [28] is a Captcha that asks users to identify cats out of a set of 12 photos of cats and dogs (see figure 1 (a)). The images used in this Captcha come from a private database of over three million images. Golle [31] reported an attack to ASIRRA. A combination of Support Vector Machine (SVM) classifiers were trained on the color and texture features extracted from the ASIRRA images. An accuracy of 82.7% was achieved in distinguishing cat from dog in a single image. But the overall success rate of attack dropped to 10.3% since there are 12 images to distinguish in each ASIRRA puzzle.

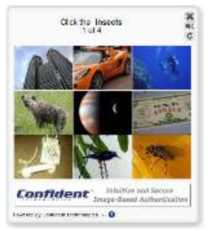
Confident Captcha [6] is another Captcha design that is built on image semantics. Users are asked to click those images out of 9 images according to a text description, as shown in figure 1 (b). The main problem of this scheme is its scalability since the images and their semantic labels have to be manually maintained.

Gossweiler et al. [32] proposed an image orientation Captcha in which the user is asked to adjust a randomly rotated image into an upward orientation, as shown in figure 1 (d). This Captcha's limitation is that the pictures used must be cautiously selected because upright positions of some images might be ambiguous.

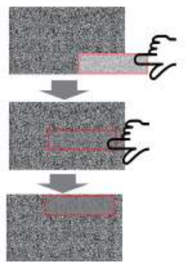
Figure 1 (c) shows a novel Captcha proposed in [40] for smartphone and tablet devices. The user is asked to drag a small noise image patch around on the background noise image until a hidden message emerges.

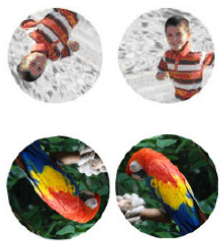
Some Captchas frame the puzzle as a game. An example is the PlayThru Catpcha [2]. As shown in figure 1 (e), this Captcha requires users to fulfil a task which is simple and funny. Unfortunately, the PlayThru Catpcha is broken in [7]. One major problem of such game Captchas is the transmission cost that fetching a game puzzle from the server is much slower than a text-based one. In [37], Mohamed et al. pro-

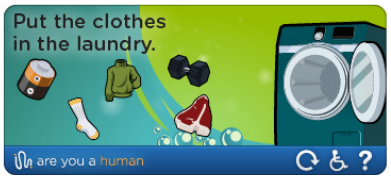

(a) Microsoft's ASIRRA.



(b) Confident CAPTCHA.


(c) CAPTCHA for Smartphone and Tablet PC by Okada and Matsuyama.


(d) Image Orientation CAPTCHA by Google.



(e) PlayThru CAPTCHA.

Figure 1: Alternative CAPTCHAs

vided a more thorough investigation into the security and usability of game Captchas.

Some other Captcha designs are: the Drawing Captcha [42] which was broken by Lin et al. [35] with an Erosion-based approach; the motion-based NuCaptcha [11] which was broken by Xu et al. [43] and Bursztein [8] separately, using different approaches.

To sum up, conventional Roman text Captchas, although having been compromised in many scenarios, still have many virtues, especially in implementation and maintenance. The various alternative Captcha designs have not completely solved the problem because most of them suffer from scalability limitation. The backend database or puzzle pool is hard to generate and update, making it difficult for wide deployment. As a result, it is of practical value to explore more alternative options for Captcha design, such as using large-alphabet language like Chinese.

3. REAL-WORLD CHINESE CAPTCHAS: A SURVEY AND ANALYSIS

Many Chinese Captchas have been deployed on the Internet. We classify them into two categories: 1) Local Chinese Captchas, which are for Chinese-speaking users. 2) Universal Chinese Captchas, which are universally usable, even to users who are illiterate in Chinese.

3.1 Chinese Characters

Chinese language has a large character set, with thousands of commonly used characters. The brute-force search space of Chinese characters is far bigger than Roman characters. Moreover, there are several other features of Chinese characters that should be taken into account for a proper Captchas design. (1) Chinese characters have no connectivity guarantee. A large number of Chinese characters comprise disconnected parts, e.g. 儿(child), 呵(breath out), 旧(old). This favoring feature naturally serves a segmentation-resistance means. Computers would easily make wrong segmentations on disconnected parts. (2) Except for a small number of simple ones, most Chinese characters are composed of radicals, and some radicals can be valid, independent characters, e.g. 加(add) = 力(force) + 口(mouth). This may cause usability problems when applying some well-established anti-segmentation techniques such as the "crowding together" and the "using variant widths". For example, the character 加 is composed of two radicals 力 and 口, which are both valid characters themselves. If the captcha engine happens to select these two characters and crowd them close, it will be confusing for human users whether to recognize them separately or as a whole. (3) Many characters are only subtly different from each other, e.g. 天(sky) vs. 夭(tender); 日(day)vs.目(eye), which demand that the distortion and camouflaging applied to a Chinese Captcha be carefully controlled to avoid poor usability.

Typing Chinese usually requires special software or plug-in which may not be accessed on every computer or device. Thus it would be of great value to design a secure Chinese Captcha scheme worldwide usable.

3.2 Local Chinese Captchas

Websites in China have been increasingly employing Chinese Captchas. They require users to recognize Chinese characters and type them to pass a test. They are designed for Chinese-speaking users. Figure 2 shows a number of such schemes taken from popular websites.

Basically, these Chinese Captchas all follow the same style as the conventional Roman Captchas. A fixed number of Chinese characters are randomly chosen and placed horizontally on a background canvas, with certain degree of distortions and camouflages applied. However, as will be discussed later, with some processing, those camouflages can mostly be removed and individual characters located. In other words, rather than adopting the segmentation-resistance principle, most of these Chinese Captchas base their security on hindering auto-recognition of individual Chinese characters.

3.2.1 Schemes (a) and (b)

BotDetect, a commercial Captcha vendor, provides a variety of localized text-based Captcha schemes, protecting over 2500 websites [3]. Figure 2 (a) and (b) give two Chinese schemes from BotDetect.

Chess Scheme

Chinese characters are embedded with a chessboard, appearing in white color on black squares and in black color on white squares. El Ahmad and Yan [26] have previously proposed a simple method to extract Latin Characters from the Chess scheme. The attack can also be applied to this Chinese Chess Captcha as follows:

- Examine each square on the chessboard. If the majority of pixels in a square are black, convert all the black pixels to white and the white to black. Nothing

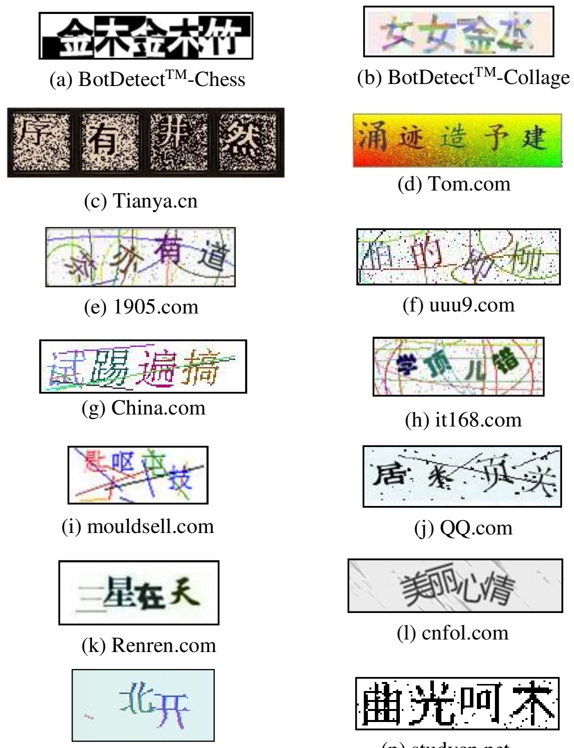

(a) BotDetect^{TM}-Chess (b) BotDetect^{TM}-Collage (c) Tianya.cn (d) Tom.com (e) 1905.com (f) uuu9.com (g) China.com (h) it168.com (i) mouldsell.com (j) QQ.com (k) Renren.com (l) cnfol.com (m) chinagames.net (n) studycn.net

Figure 2: Real-world local Chinese Captchas.

will be done if the majority of pixels in the square are white. The result of this process is demonstrated in Figure 3(b).

- Slice the image vertically into several segments using the method introduced in our previous work [44]. Each segment contains one Chinese character.

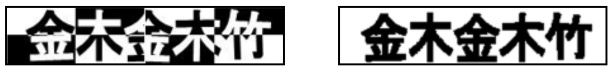

(a) Challenge (b) Extracted characters

Figure 3: Chess scheme from BotDetect^{TM}

Collage scheme

Chinese characters are rendered with colorful texture and the background is split into several vertical regions, each painted in a different pale color. The characters can be extracted with simple image processing:

- Binarize a challenge image, as shown in Figure 4 (b). This operation is possible due to the significant intensity difference between foreground characters and background canvas.
- Remove the noise-dots left after the binarization operation, as shown in Figure 4 (c). There are many techniques available for this goal. One possible way is by removing chunks of small pixel counts.
- Vertically segment the image into individual characters, using the method of [44].

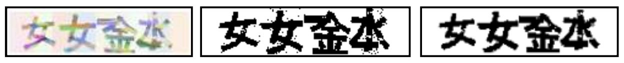

(a) Sample challenge (b) Binarized image (c) After noise removal

Figure 4: Collage scheme from BotDetect

Note that the Captcha length in BotDetect schemes can be customized. When more characters are used in the Captcha, the characters tend to be crowded together, making the segmentation task more difficult. But obviously the usability of the Captcha has to be compromised in this case.

3.2.2 Scheme (c)

Figure 2(c) is an example taken from Tianya.cn, the largest Chinese forum online in the world [13]. In this scheme each Chinese character is randomly positioned inside a dark brown square. Segmenting the image into individual character boxes is trivial, but a main hurdle here is to distinguish the character from the noise-like background in each box. It can be observed that after binarization the character boxes exhibit two styles. The style 1 box, as shown in Figure 5 (b) presents the character in black on a white background with surrounding single-pixel noise. The style 2 box, as shown in Figure 6 (a), is presented in exactly the opposite colors but can be converted to its complement form as shown in Figure 6 (c). After the two styles being unified, the noise-dots in each box can be cleared as follows:

- Clean up the character box by removing any small black chunks. The results of this operation are shown in Figure 5 (c) and Figure 6 (d) respectively.
- Draw a bounding box around the Chinese glyph and remove all the black pixels outside the boundary. The results are shown in Figure 5 (d) and Figure 6 (e) respectively.

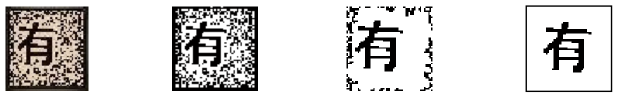

(a) (b) (c) (d)

Figure 5: Processing style 1 images. (a) Original; (b) Binarized image; (c) After noise removal; (d) After cropping and rescaling

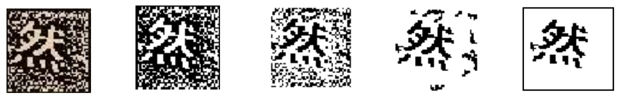

(a) (b) (c) (d) (e)

Figure 6: Processing style 2 images. (a) Original; (b) Binarized image; (c) Complement form; (d) After noise removal; (e) After cropping and rescaling

3.2.3 Schemes (d)-(j)

All the schemes shown in Figure 2 (d)-(j) implement the segmentation-resistance principle with cluttered background. The scheme in Figure 2(d) from Tom.com [14] uses a background with gradually changing colors, attempting to interfere with the automatic extraction of the characters. However, since the color applied to each character is constant and of obviously higher intensity than the background, it is not difficult to distinguish the individual characters from the background.

The schemes in Figure 2 (e)-(j), although taken from different websites such as QQ.com [12], China.com [4], 1905.com [1], uuu9.com [16], it168.com [9], and mouldsell.com [10], all use the same type of background cluttering. Specifically, one-pixel-thick arcs as well as one-pixel dots are randomly rendered as clutters on the background. These clutters as well as the target characters are in arbitrary colors, with the exception that the scheme (j) uses the same color for both the characters and the clutters. In fact, this cluttering style has been well studied in previous works. Cleaning such kinds of clutters has proved to be a trivial task. Some effective techniques can be found in [18,46] for this purpose. Moreover, using invariant number of characters in addition to distinctive color for each character makes it easy to segment and extract the individual characters [46].

3.2.4 Schemes (k) - (m)

Rather than relying on cluttered background, the schemes shown in Figure 2 (k), (l) and (m) roughly exploit the "crowding together" rule to resist auto segmentation. In these schemes, there is hardly any space between adjacent characters. In particular, the rotation in scheme (l) makes it more difficult to segment the characters by vertical slicing. However, these schemes are easy to break. This is because the Chinese characters, unlike Roman characters, do not differ much in their sizes. In other words, given the same font, the square boundaries of each Chinese character are almost the same. This feature can be used to roughly segment the characters even though they are connected to each other, especially when the number of characters is constant. To make it worse, for scheme (k) and (m), we can simply identify and separate individual characters by detecting their distinct colors.

3.2.5 Scheme (n)

The last example in Figure 2(n) hardly applies any effective technique to resist auto-extraction of target characters. The characters are well separated apart, without any clutters connecting or crossing them. The noise-dots can be easily cleared due to its slight density.

3.3 Universal Chinese Captchas

It seems impossible at first glance to develop Chinese Captchas that can be used by people who have never studied. However, there are a few such universal Chinese Captcha schemes available on the Internet. The Touclick and the CCaptcha are among the most popular ones, which will be discussed as follows.

3.3.1 Chinese Touclick

As shown in Figure 7, Touclick [15] works as follows. Two Chinese characters are randomly placed on background picture. A hint which contains the two characters is displayed below the puzzle. Users are required to find out the two characters in the puzzle and click them in the same order as they appear in the hint. The fact that no Chinese typing is required makes Chinese Touclick applicable to users who do not know Chinese at all.

The background picture is randomly chosen, most of which are arts of natural sceneries. These fancy background pictures are supposed to play the role of disturbance to stop computers from solving the puzzle. However, the appearance of scenery and a Chinese character is quite different. Shen et al. have successfully hacked the Chinese Touclick in [41], exploiting the fact that Chinese characters have much more corners than a natural scenery image. They first render two character images according to the hint and extract the corner features from them. These features are then compared to those extracted from the puzzle image to pick the characters out of the background. Since there is hardly any distortion applied to the Chinese characters in the puzzle, a simple Euclidean distance based matching process suffices to locate the target characters. The reported accuracy is 100%.

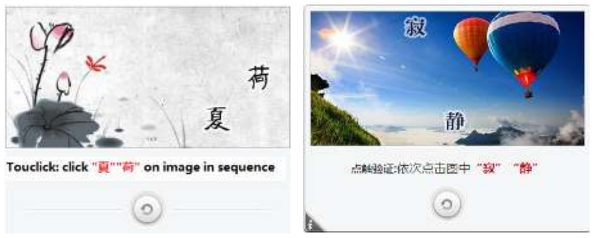

荷
夏
Touclick: click "夏""荷" on image in sequence
寂
静
点触验证:依次点击图中 "寂" "静"

Figure 7: Examples of the Chinese Touclick CAPTCHA.(a) taken from [41]; (b) from Touclick website [15]

3.3.2 CCaptcha

The CCaptcha scheme, filed as a US patent application [5], exploits a database of Chinese characters, as well as radicals of these characters. Although each puzzle could hardly be solved by the well-known OCR techniques [22], it can easily be solved via observation by human users who are not necessarily literate in Chinese. The designers performed a large scale user study, in which native speakers and foreigners who did not know Chinese achieved similarly high accuracy in solving this Captcha [21].

A CCaptcha challenge (Figure 8) is composed of 10 images. The bigger image on the left is a Chinese character that can be decomposed into elementary radicals. The other nine smaller images represent candidate radicals, among which some are real radicals from the target character and the rest are faux ones. To pass the test, a user has to recognize all real radicals and click corresponding images. Selecting any faux radical will fail the test.

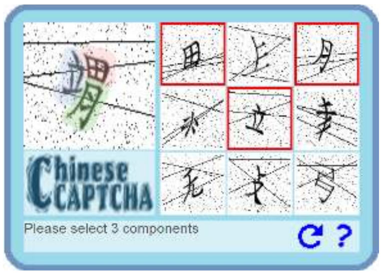


Figure 8: Example of a CCAPTCHA puzzle

The inventors applied OCR software to recognize distorted characters and radicals used in this scheme, but to no avail. And this is their main argument for the security of the scheme [22].

3.3.3 *Observations*

Clearly, these Chinese Captchas that we examined above, whether local ones or universal ones, mostly build their security on the assumed recognition-resistance of Chinese characters to computers. It is therefore natural to ask a fundamental question: *does the large Chinese character set really provide this recognition-resistance capability?* This open question motivates our following experiments.

4. CAN COMPUTERS RECOGNIZE DISTORTED SINGLE CHINESE CHARACTERS?

4.1 Prior Art

Chellapilla et al. [19] examined how good computers are at recognizing single Roman characters with the hindrance of distortions and random clutters. Their result suggests that computers are able to achieve close to 100% success in most of the situations they tested. Their recognition rate deteriorated with the increasing number of thick intersecting foreground arcs, but remained above 60% even in the worst situation where human's recognition accuracy dropped to below 10%.

This work has established the principle of segmentation resistance that has a profound impact on Roman-character based text Captchas. It suggests that the designers of Roman-character based text Captchas should not build their security on the task of character recognition, which is an easy task for computers.

However, it is unclear whether their result is relevant to Captchas that are based on a language of a large character set, such as Chinese and Japanese. As the character sets in these languages are much larger than the Roman alphabet (thousands vs. dozens), intuitively it makes it a much harder problem to automatically recognize the former than the latter.

Our analysis in the previous section suggests that the state-of-the-art Chinese Captchas mostly rely on the assumed but never tested hardness of recognizing distorted single characters by computers. To find out whether computers can be impeded by a large character set, we extend Chellapilla et al.'s experiments to Chinese characters. To the best of our knowledge, our experiment is the first of its kind, and there is no direct evidence to either support or deny the extension of the results of [19] to a language of a large character set.

4.2 Data Preparation

We use 3755 most popular Chinese characters from China National Standard GB2312.

The recognition experiments in [19] tested 7 different combinations of distortion and clutter style. For each configuration, the distortion and clutter range from easy to difficult with 4 or 5 scales, leading to a total of 33 rounds of experiments. Such settings were possible for a recognition task that involves 30 characters (uppercase letters and digits excluding a few characters that can be easily confused), but they become prohibitive when applied to Chinese character recognition experiments that involve thousands of characters.

It should be noted that our goal is not exactly the same as in [19]. Their purpose was to compare the capability of computers and humans in recognizing distorted characters. As a result, their experiment settings were pushed to extreme cases that would not be considered in real-world Captchas. However, our experiments are mainly to find out whether computers would be deterred by the large alphabet set of Chinese characters as this is not established yet. Based on the above considerations and given the time and computation capacity permitted, we decided to tailor our experiments to three configurations. Our principles are simply that when necessarily, we follow all the methods and configurations as in [19], but we also avoid design choices that are appropriate for Roman characters but not for Chinese characters.

Specifically, we followed the same geometric transformations as in [19], which combined scaling (randomly from -20 to $+20$ percent) and rotation (randomly from -20 to $+20$ degrees).

We did not separate the local warp and global warp for the sake of simplification. Instead, we generated a unified parameterized distortion field and empirically fixed the parameter to ensure a distortion that was more difficult than the state-of-the-art Chinese Captchas but still acceptable for humans.

We also discarded the use of background arcs and non-intersecting foreground arcs, because they had been proven in [19] as the least useful in confusing computers. The most challenging setting revealed in [19] was the foreground arcs intersecting with the character. We kept this setting in our experiments but only tried on two scales, i.e. 8 arcs and 16 arcs per character respectively. Note that the arcs we rendered were all one-pixel-wide. This is because within an image of 40×40 pixels (as required by our recognition engine), there is no adequate space to render thicker arcs. Given the complexity of Chinese characters, if thicker arcs were used, the image would easily be filled up, making it meaningless for either computers or humans.

Figure 9 gives examples of three groups of datasets we generate. The specifications are described as follows:

- Group1: Geometric transformation plus random warping.
- Group2: In addition to the same processing as Group 1, eight one-pixel-wide foreground arcs are randomly generated and placed on top of each character.
- Group3: The same processing as in Group 2 with the number of arcs doubled to sixteen.

All the Chinese characters are normalized to 40×40 and then centered on a 48×48 blank background canvas.

For each group, we generate for each character a training set containing 260 samples, and a test set of 60 samples. The total number of samples in each group is: $3755 \times (260+60) = 1,201,600$.

Each group of training/test dataset is converted into two files, one with the images and one with the corresponding labels as follows:

- Training set files: contain $260 \times 3755 = 976,300$ images and their corresponding labels.
- Test set files: contain $60 \times 3755 = 225,300$ images and their corresponding labels.

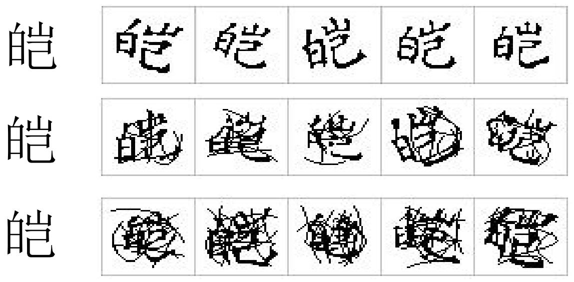


Figure 9: Examples of groups 1, 2 and 3, respectively

4.3 A Failed attempt

We conducted a number of trials using the Convolutional Neural Network (LeNet-5) which is widely used to recognize digits and Roman characters.

We first conducted our experiments on a group of 1000 Chinese characters. We used 161 training samples and 20 testing samples for each single character. On a standard desktop with 8 GB RAM and Intel(R) Core(TM) i7-3770 CPU @3.40GHz, our CNN achieved 68.18% success on the test set. It took approximately 24 epochs to train the engine; each epoch taking about 10 minutes.

We then test on a group of 3755 Chinese characters. We used 161 training samples and 20 testing samples for each single character. The desktop used for the previous trial was not up to this task, making nearly zero progress after weeks. Finally, we decided to run the experiment on an expensive huge machine with 64 GB RAM and 2 Intel(R) Xeon(R) CPU E5-2650L 0 @ 1.80GHz CPUs. This time, it took about 24 epochs to train the engine, each epoch taking about 4-5 hours. The trained CNN engine achieved less than 35% success on the test set. By using a doubled sample size (322×3755 training samples) and (40×3755 test set), the required learning time is doubled for each epoch, but the eventual recognition success decreased to less than 11% on the test set.

Our initial failure suggests that 1) the automated recognition of distorted single Chinese characters is not trivial, and 2) the complexity of recognizing such characters provides some evidence that Chinese Captchas based on recognition difficulty are more secure than their Roman counterparts.

These observations are reasonable, given that the tested Chinese alphabet (3375 classes) is over one hundred times bigger that the Roman alphabet (26 classes). This increases the difficulty of the problem by requiring both bigger networks and much more training data. Therefore, we are motivated to use deeper and wider NNs (i.e. Deep Neural Networks).

4.4 DNNs and Multi Column DNNs

Lately, Deep Neural Networks (DNN) significantly decreased the error rate on many classification tasks [23,24,33]. A DNN obtained the best results at two competitions about classification of offline (no temporal information) handwritten Chinese characters organized at ICDAR 2011 [36] and ICDAR 2013 [47]. We decided to use similar architectures for our experiments.

4.4.1 DNN architecture

A DNN uses a feed-forward architecture, without recurrent connections. It starts with an **input layer** and continues with a linear succession of various types of layers. Our DNNs use convolutional, max-pooling and fully connected layers. Each non-fully connected layer has its neurons arranged in a list of rectangular maps. Figure 11 depicts the architecture of one of the DNN we use in the paper.

The **convolutional layers** (in red in Figure 11) extract features from the maps in the previous layer by convolving their input with a small filter that contains the feature to be detected. These filters are learned during the training process.

The **max-pooling layers** (in blue in Figure 11) select the locations where the features were most active by choosing the maximum in non-overlapping square patches, usually 2×2 in size.

Convolutional and max-pooling layers repeat several times, building more complex feature detectors with each additional stage.

When the size of the maps becomes too small to add another stage of convolutional and max-pooling layers, we use **fully connected layers** to classify the features extracted by the previous layers. The very last layer uses a soft max activation function that outputs probabilities for the input to be part of a particular class.

4.4.2 MCDNN architecture

We also use a MCDNN as in [23], averaging the outputs of several independently trained DNN. A MCDNN is used only during testing, corresponding output neuron values being averaged (see Figure 10).

For simplicity, we choose to use DNNs with identical architecture. Still, they differ because they are initialized with different random weights, the image-label pairs are presented randomly during training, and the distortions applied on the input characters are also random.

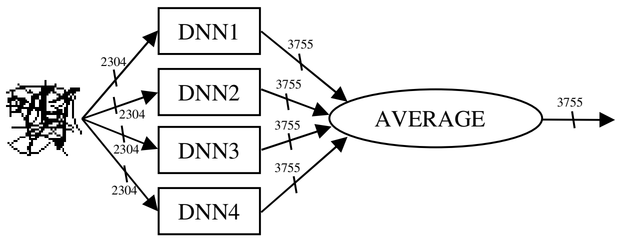


Figure 10: 4-column DNN. Each character (48x48 pixels) is sent to the input of each DNN column. After processing, the output vectors are averaged component wise.

4.5 Experiments and Results

We depict the characters on 40×40 pixels images, as done in [23]. These images are centered in a 48×48 pixel image, to allow space for small distortions which are used to augment the training set. The characters are stored in gray scale images with 0-255 for pixel intensity information. The input of the network directly maps to the pixel values, without any feature extraction. As the neurons work with real values from $[-1, +1]$, the pixels intensities are linearly translated and rescaled.

The DNN used for ICDAR competitions [25] has hundreds of maps per layer. As our characters are artificially generated, thus probably easier to classify than handwritten characters, we start with smaller architectures. We keep the

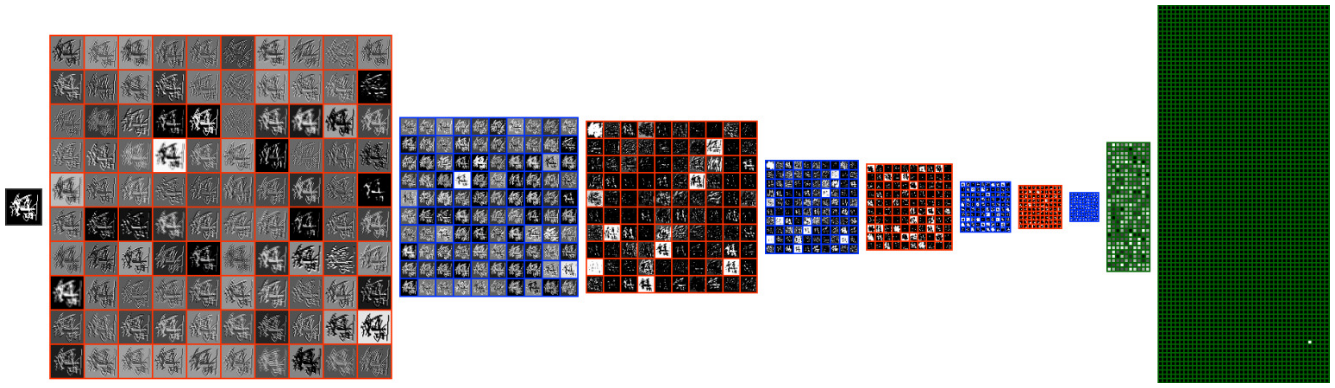


Figure 11: A Chinese character (left) is propagated through a trained DNN with architecture 48×48-300C3-MP2-300C2-MP2-300C2-MP2-300C2-MP2-300N-3755. For space reasons only the first 100 maps in each convolutional (red) and max-pooling (blue) layers are shown. The fully connected (green) layers are scaled up 4x, the rest of the net is drawn at scale. It can be observed that one of the 3755 output neurons, corresponding to the class of the image at the input, is lit up.

convolutional and max-pooling layers, but we use far fewer maps in every layer, e.g. instead of 100-450 maps, we only use 50 maps for our smallest networks. All DNN share the same template, we only modify the numbers of maps per layer and the number of neurons in fully connected layers. Due to the smallest convolutional and max-pooling kernels, this template has the advantage of creating the deepest possible network for the 48×48 input size.

A compact representation of our DNN looks like this: `48x48-mC3-MP2-mC2-MP2-mC2-MP2-mC2-MP2-nN-3755N`, where 48×48 is the input layer, m is the number of maps in convolutional and max-pooling layers, C3 denotes a convolutional layer with a convolution filter of 3×3, MP2 denotes a max-pooling layer with a kernel of 2×2, n represents the number of neurons in the hidden fully connected layer, and 3755 is the number of categories to recognize. To further clarify the notation, our bigger network is named `48x48-300C3-MP2-300C2-MP2-300C2-MP2-300C2-MP2-300N-3755` (see Figure 11 for graphical depiction). This network has 1024859 neurons, 2574455 weights and 224018555 connections.

We use a GPU implementation of the training algorithm (Stochastic Gradient Descent - SGD) described in [24]. This allows us to train and test several DNN in a reasonable amount of time. All weights are randomly initialized with values from $[-0.05, 0.05]$. The initial learning rate is 0.001 and is multiplied with 0.95 at the end of every epoch. We train the DNNs on the training dataset from each group. If we train the DNN with distortions, then the characters from the training set are always distorted prior to be passed through the network. At the beginning of every epoch, new distorted versions are generated from the originals, thus if the net is trained for 50 epochs, then it will see 50 times more characters than the characters from the training data, although they are variations of the original characters. For validation we use the original, undistorted training set. The characters from the test set are used exclusively for testing and they are not distorted. We train 10 DNNs (Table 1) using the three data groups. We also build two MCDNNs with the networks trained on Group 3.

4.5.1 *Differences between Le Net 5 and DNN*

When it was introduced, Le Net 5 was state of the art for convolutional neural networks. It represented an impressive achievement of research and engineering. More than ten years later, current DNNs use many of the ideas from Le Net 5, but complement them with state of the art layer architecture, better training algorithms and sheer size.

The biggest distinction between Le Net 5 and a DNN is their size; the latter is both deeper (more layers) and wider (more maps/neurons in each layer). The four convolutional layers of DNN compared to the only two of Le Net 5 allow for extracting more complicated features. Le Net 5 has only 6 and 16 maps in the first and the second convolutional layers, respectively. Our DNN has 50 to 300 maps in each of the convolutional layers. The easiest way to compare the complexity of the two models is to look at the number of connections: 340068 for Le Net 5 versus 224018555 for our biggest DNN (658 times more connections!).

Le Net 5 uses trainable subsampling layers. DNNs use simple but efficient max-pooling layers which perform feature selection by choosing the maximum activation from the corresponding input patch.

4.5.2 *Experiments on Group 1*

As this group is the simplest, we start with a "thin" network (DNN1) which has only 50 maps in each convolutional and max-pooling layer. This architecture seems to be sufficient as it almost fully learns the training/validation set and reaches a very low 0.342% error rate on the test set.

We try to prevent the network to overfit on the training set and make training harder by adding full distortions on the training set (please note that the training set is already composed of distorted characters as described in [19], here we generate online new versions of these characters), i.e. maximum 10% translation, maximum 10°rotation, maximum 10% scaling, and elastic distortions. Although this prevents the network (DNN2) to fully learn the training data, it does not help with generalization, as the error on the test set increases to 0.475%.

We do not experiment with wider networks because the error on test characters is sufficiently low to consider this task solved.

Table 1: DNN experiments on single Chinese characters. (Epochs: the number of epochs required to reach minimum error on validation data. Distortions: translation / rotation / scaling / elastic parameters)

DNN/ MCDNN	Architecture (maps/neurons per layer)	Data	Epochs	Distortions	Validation error [%]	Test error [%]	Test error (first 10 predictions) [%]	Training time per epoch [s]	Test time per char [ms]
1	50/200	G1	121	No	0.004	0.342		2900	1.15
2	50/200	G1	113	10/10/10/6/30	0.407	0.475		3230	1.15
3	50/200	G2	50	No	6.767	14.763		2750	1.09
4	100/200	G2	65	No	0.695	11.597		5550	1.15
5	200/200	G2	24	No	0.036	10.052		16800	2.84
6	300/300	G2	19	No	0.000	7.805		34950	5.30
7	300/300	G3	29	No	0.000	19.948	3.233	35000	5.37
8	300/300	G3	135	10/0/0/0/0	9.587	15.531	2.324	35000	5.32
9	300/300	G3	95	10/5/0/0/0	9.918	15.528	2.306	35000	5.32
10	300/300	G3	95	10/5/5/0/0	9.903	15.517	2.279	35000	5.32
11	3 nets: 8, 9, 10	G3	-	-	-	10.812	1.416	-	15.96
12	4 nets: 7-10	G3	-	-	-	**10.086**	1.291	-	21.33

4.5.3 *Experiments on Group 2*

We start with exactly the same architecture and distortions as for Group 1 (DNN1). If before the validation error was practically 0, now (DNN3) it is 6.767% showing that this dataset is much harder to learn. In the next three experiments we keep increasing the width of the networks to 100 (DNN4), 200 (DNN5) and 300 (DNN6) maps per layer. Both validation and test error are decreasing, indicating that more capacity is needed. The biggest network reaches zero error on validation and 7.805% error on test.

Distortion could be used to prevent learning the training set and improve generalization, but we decide that it is better use our limited computing time to experiment with the hardest dataset, Group 3.

4.5.4 *Experiments on Group 3*

For the hardest dataset we start directly with the same architecture as the biggest network trained on Group 2. Although this network (DNN7) manages to learn the entire training set, almost one test character out of five is misrecognized (test error is 19.948%). Looking for better generalization, we train three more networks (DNN 8, 9 and 10) with various amounts of affine distortions. All of them reach similar test error (15.5%) and improve over DNN7. After one single training epoch, DNN10 is capable of correctly classifying more than 60% of the characters from the test set (Figure 12). It needs 26 epochs to reach 20% error and 95 epochs to go to 15.517% error on the test set.

We also compute how many characters are not recognized even when looking at the highest 10 predictions the networks compute, i.e. the label is not part of the classes with the highest ten probabilities. This error is important because the characters in this list have similar appearance, thus probably they will share many of their radicals. Even if the character is incorrectly classified, the probability is high that it contains the required radicals.

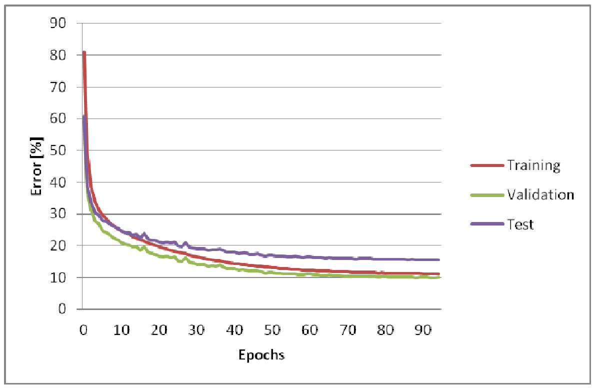


Figure 12: Error rates for training of DNN 10.

4.5.5 *Experiments with MCDNN (on Group 3)*

We tested the method from [23] by building two MCDNNs (committees of NN): one averages the networks trained with distorted images, and the other averages all DNN trained on Group 3. Both MCDNNs significantly decrease the recognition error with up to 35.01% with respect to the best DNN.

The test error computed considering the characters with the highest 10 predictions is almost halved, reaching a low 1.291%.

4.5.6 *Speed*

We use relatively old NVIDIA GTX 580 graphics cards for all experiments. It is slow to train DNNs when the datasets contain over one million characters and the networks have hundreds of millions of connections. If the small networks require less than 1 hour of training per epoch, the biggest networks need about 10 hours per epoch.

Once trained, testing is much faster at 1.1 to 5.3ms per character for various networks. The MCDNNs are slower because they need to calculate the output of four networks.

Even so, up to 50 characters can be checked each second. Individual DNNs can be computed separately, thus the classification time decreases linearly with the number of CPUs/GPUs used.

The test phase can be further optimized specifically for this task, and ported to the state-of-the-art GPUs like NVIDIA Titan X. This should decrease the time required to test a character by 5 to 10 times.

5. EXPERIMENTS ON CCAPTCHA

Sections (4.5.2 - 4.5.5) show that MCDNNs and DNNs can recognize Chinese characters with high accuracy even though the characters are simultaneous distorted (affine and elastic) and overlapped by up to 16 arcs in an attempt to make the problem even more difficult (Figure 9). Here we try to attack CCaptcha (Section 3.3.2) by the means of the same DNNs/MCDNNs.

5.1 Data

CCaptcha uses 66111 characters/radicals, thus its size is 66111 (categories) × 260 (samples per category) × 48 × 48 (character size) = 36.9GB. This exceeds by far our machines DRAM (which has to store multiple buffers related to the dataset, and several DNNs). Randomly accessing the data from a Solid State Drive slows down the training process by over one order of magnitude compared to reading the data from DRAM. We are updating our code and machines to deal with datasets of this size. For this work we decided to use a subset, specifically 3755 categories (the same as in Sections 4.5) out of the 66111. We name this set G4. This preliminary test will show if DNNs can break.

Exploiting an information leakage hole that we discovered in the design of CCaptcha (details see [17]), for each of the 3755 classes, we collect 260 samples for training and another 60 for testing. The total number of samples in G4 is: $3755 \times (260 + 60) = 1,201,600$. G4 includes a mixture of characters and radicals where:

- There are roughly over 2500 classes of characters composed of more than 3 explicit radicals.
- Less than 300 classes are definitely radicals because they could not be further broken into smaller parts.
- Some samples have only two explicit radicals.

These observations point out that this data set is largely composed of complex enough characters (some of which are even more complex than our previous data G1-G3).We can thus confirm that G4 is not biased by the complexity of the characters. That is, characters in G4 have similar or higher complexity than those from G1-G3. Thus, the results of the experiments are influenced mostly by the level of distortions and noise.

5.2 Preprocessing

The character/radical images collected from the CCaptcha server have arbitrary sizes. Their width and height have random values from and 138–166 and 64–97 pixels, respectively. The sizes of characters/radicals and their locations on the canvas vary, too. To make the data compatible with our DNN engine, preprocessing is required. We first locate the bounding box of the glyph and extract it from the original canvas. We then scale the extracted glyph to fit a 40 × 40 pixels box. The scaled glyph is then placed in the center of a 48×48 pixels patch that is cut from the original canvas. In this way, we are able to keep the original distortions and camouflages (salt/pepper noise and 3 overlapping lines) while scaling the image to satisfy the DNN specifications. Figure 13 gives examples of G4.

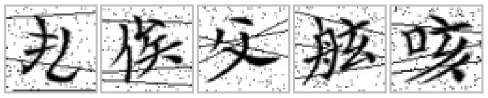


Figure 13: Examples from G4.

The training and the test datasets of G4 are converted to two files in the same way described in previous sections, one with the images and one with the corresponding labels.

5.3 Results

We trained several networks with identical architectures to those in Table 1. DNNs learn very fast to classify characters from G4. After only several epochs the error on the test set is already well under 1%. In just one epoch DNN5 and DNN6 (the biggest networks) go under 0.5%. Comparing "Test errors" in Table 1 and Table 2, we observe that G4 dataset is much easier than G1, G2 and G3. The fact that G4 is easier than G1 (compare DNN1 from Table 1 and Table 2) could seem counterintuitive because G4 has both salt and pepper noise and 3 overlapping segments. We observe that characters from G1 are overly processed and with rough edges (Figure 9). G4 has smooth strokes, as we use the original characters from CCaptcha. It turns out that the quality of the strokes has a greater positive effect on recognition than the negative effect of the noise and segments.

G2 and G3 contain 8, respectively 16, arcs intersecting the characters. The lowest error (7.805%) is obtained with DNN6 (Table 1). An identical network (DNN6 in Table 2) trained on G4 reaches 0.002% error rate, practically solving the CCaptcha problem with 3755 categories. There are only four incorrectly recognized characters (Figure 14).

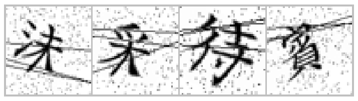


Figure 14: The four incorrectly classified characters of G4 by DNN6.

Although increasing the number of maps in each convolutional layer helps in decreasing the error, all DNN from Table 2 have very small errors, almost close to 0%. We interpret this result as an indication that DNN should work well even on the full CCaptcha with 66111 categories. For the reduced set we did not use MCDNN, but they could prove useful for the big dataset.

5.4 Breaking CCaptcha

We briefly discuss how to convert the power of DNN/MCDNN attacks to break CCaptcha.

At least several methods can break CCaptcha. One is to combine the power of DNN/MCDNN with a dictionary attack. Exploiting API vulnerabilities that we identified in

Table 2: DNN experiments on 3755 categories from CCaptcha

DNN/ MCDNN	Architecture (maps/neurons per layer)	Data	Epochs	Distortions	Validation error [%]	Test error [%]	Test error (first 10 predictions) [%]	Training time per epoch [s]	Test time per char [ms]
1	50/200	G4	62	No	0.002	0.162	0.003	2900	1.15
2	50/200	G4	113	10/5/5	0.102	0.124	0.000	3230	1.15
3	100/200	G4	32	No	0.000	0.067	0.000	5550	1.15
4	100/200	G4	31	10/5/5	0.034	0.042	0.000	5680	1.15
5	300/300	G4	8	No	0.000	0.023	0.000	35000	5.37
6	300/300	G4	31	10/5/5	0.001	0.002	0.000	35000	5.32

CCaptcha, we can build a dictionary that maps each Chinese character to a set of valid radicals (details see our previous work [17]). Or, we could build such a dictionary using general Chinese reference books. To solve a CCaptcha puzzle, we simply pick up its target character, as well as nine radicals from the traffic. We then let the trained DNN/MCDNN recognize each of them. A dictionary lookup will tell which are valid radicals for a given character.

Alternatively, we can train a DNN/MCDNN to identify each character-radical pairwise relationship. Then, to solve a CCaptcha, we pick up images from the traffic both for the target character and the 9 radicals, and then use the trained DNN/MCDNN to find valid radicals.

6. COMPARING WITH HANDWRITTEN RECOGNITION

Considering the very low error rate for G4, we can conclude that DNNs are very good image denoisers, as they can easily deal with noise (pixels and 3 random segments) and moderate distortions (affine and elastic).

We compare our results with the results obtained in [25] with identical DNN architecture, applied to handwritten Chinese characters. Our results on G4 are unexpected to us, as we expected G4 to have a similar difficulty as Handwritten Chinese Recognition (HCR) without noise.

The much higher error rate obtained for handwritten characters (4.21%, see [25]) suggests that the stroke variability and the deformation/placement of radicals present in handwritten characters are significantly harder to generalize from than the artificially generated characters from CCAPTCHA (G4), even if they are elastically distorted and contain noise. Although we had no possibility to use exactly the same categories as in [25], we checked visually that the complexity of the categories from the two datasets (handwritten and CCAPTCHA) is similar.

Only when the added noise becomes too intense as a consequence of too many arcs intersecting the characters' strokes (G2 and G3) the DNNs start to have difficulties in recognizing the artificially generated characters. This indicates that by themselves the geometric distortions used in CCAPTCHA are too low to confuse a DNN.

Handwritten Chinese and automatically distorted Chinese characters in our synthesized datasets might appear to be only slightly different, but they are significantly different image objects for computers. Figure 9 shows some distorted characters that are much harder to recognize than handwritten ones even for humans.

7. CONCLUSIONS

We for the first time have systematically analyzed Chinese Captchas, and answered a fundamental question: is the segmentation-resistance principle established for Captchas using Roman characters applicable to Chinese based designs? With deep learning techniques, we have offered the first evidence that computers do recognize individual Chinese characters well, regardless of distortion levels. This result suggests that most real-world Chinese Captchas are not secure, in contrast to common beliefs. Our result is also applicable to Captcha designs based on other large-alphabet languages such as Japanese.

In summary, we have the following lessons and guidelines for Chinese Captcha design.

- Although most such Captchas are designed for Chinese users alone, it is possible to design Chinese Captchas that are universally usable. The trick is that each Chinese character is effectively a picture, and thus it is possible to design a Captcha that is both a text scheme and an image recognition scheme, simultaneously.

- In contrast to CCaptcha's security argument, our work suggests that CCaptcha, as is, is not secure. Given its clever and counterintuitive idea, it is worthwhile to improve it into a more secure design, rather than throwing it away altogether.

- It is more demanding to break recognition-based Chinese schemes than Roman-character ones. The former's large alphabet does raise the bar for recognition attacks with machine learning techniques.

- However, the principle of segmentation resistance is applicable to Chinese Captchas. That is to say, they have to be segmentation-resistant to be secure.

It is interesting future work to systematically study how to design Chinese Captchas that are simultaneously secure and usable.

Acknowledgements

We thank the anonymous reviewers for their comments.

8. REFERENCES

[1] 1905, http://www.1905.com, accessed: 2015-05-07

[2] Are You A Human, http://areyouahuman.com/, accessed: 2013-12-06

[3] Botdetect captcha, http://captcha.com/localizations/chinese-captcha.html, accessed: 2013-12-20

[4] China.com, http://www.china.com, accessed: 2015-05-07

[5] Chinese CAPTCHA, http://crecaptcha.org, accessed: 2013-09-01

[6] Confident technologies, http://confidenttechnologies.com/products/confident-captcha, accessed: 2013-09-01

[7] Cracking the areyouahuman captcha, http://spamtech.co.uk/software/bots/cracking-the-areyouhuman-captcha/, accessed: 2013-12-06

[8] How we broke the nucaptcha video scheme and what we propose to fix it, http://elie.im/blog/security/how-we-broke-the-nucaptcha-video-scheme-and-what-we-propose-to-fix-it, accessed: 2013-09-01

[9] it168.com, http://www.it168.com, accessed: 2015-05-07

[10] Mouldsell.com, http://www.mouldsell.com, accessed: 2015-05-07

[11] Nucaptcha, http://nucaptcha.com/demo

[12] QQ, http://www.qq.com

[13] Tianya, http://bbs.tianya.cn/post-no110-342020-1.shtml, accessed: 2013-12-20

[14] Tom.com, http://www.tom.com, accessed: 2015-05-07

[15] Touclick, http://www.touclick.com, accessed: 2015-05-07

[16] UUU9, http://www.uuu9.com, accessed: 2015-05-07

[17] Algwil, A., Yan, J.: Failures of Security APIs: A New Case. In: Proceedings of Financial Cryptography and Data Security. Springer (2016)

[18] Bursztein, E., Martin, M., Mitchell, J.C.: Text-based captcha strengths and weaknesses. In: Proceedings of the 18th ACM conference on Computer and communications security. ACM (2011)

[19] Chellapilla, K., Larson, K., Simard, P., Czerwinski, M.: Computers beat Humans at Single Character Recognition in Reading based Human Interaction Proofs (HIPs). In: 2nd Conference on Email and Anti-Spam (CEAS) (2005)

[20] Chellapilla, K., Simard, P.Y.: Using machine learning to break visual human interaction proofs (hips. Advances in neural information processing systems 17 (2004)

[21] Chen, L.: Personal Communications (2014)

[22] Chen, L., Juang, D., Zhu, W., Yu, H., Chen, F.: CAPTCHA AND reCAPTCHA WITH SINOGRAPHS, uS20120023549 A1- 2012

[23] Cireșan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE (2012)

[24] Cireșan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. vol. 22, pp. 1237–1242 (2011)

[25] Cireșan, D.C., Meier, U.: Multi-Column Deep Neural Networks for Offline Handwritten Chinese Character Classification. In: International Joint Conference on Neural Networks (2015)

[26] El Ahmad, A.S., Yan, J., Ng, W.: Captcha design: Color, usability, and security. IEEE Internet Computing 16(2), 44–51 (2012)

[27] El Ahmad, A.S., Yan, J., Tayara, M.: The robustness of google captchas. Tech. rep., Computing Science, Newcastle University (2011)

[28] Elson, J., Douceur, J.R., Howell, J., Saul, J.: Asirra : A captcha that exploits interest-aligned manual image categorization. In: ACM Conference on Computer and Communications Security (2007)

[29] Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., Yan, J.: The robustness of hollow captchas. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13. pp. 1075–1086. ACM Press (2013)

[30] Gao, H., Yan, J., et al: A simple generic attack on text captchas. In: Network and Distributed System Security Symposium (NDSS). San Diego, USA (2016)

[31] Golle, P.: Machine learning attacks against the asirra captcha. In: Proceedings of the 15th ACM conference on Computer and communications security - CCS '08. pp. 535–542. ACM (2008)

[32] Gossweiler, R., Kamvar, M., Baluja, S.: What's up captcha? a captcha based on image orientation. In: Proceedings of the 18th international conference on World wide web. pp. 841–850. ACM (2009)

[33] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems NIPS 2012. pp. 1097–1105 (2012)

[34] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)

[35] Lin, R., Huang, S.Y., Bell, G.B., Lee, Y.K.: A new captcha interface design for mobile devices. In: Proceedings of the Twelfth Australasian User Interface Conference. vol. 117, pp. 3–8. Australian Computer Society, Inc. (2011)

[36] Liu, C.L., Yin, F., Wang, Q., Wang, D.: ICDAR 2011 Chinese Handwriting Recognition Competition (2011)

[37] Mohamed, M., Sachdeva, N., Georgescu, M., Gao, S., Saxena, N., Zhang, C., Kumaraguru, P., van Oorschot, P.C., Chen, W.B.: Three-way dissection of a game-captcha : Automated attacks , relay attacks , and usability. In: ASIA CCS. ACM, Kyoto, Japan (2014)

[38] Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual captcha. In: Computer Vision and Pattern Recognition. IEEE (2003)

[39] Moy, G., Jones, N., Harkless, C., Potter, R.: Distortion estimation techniques in solving visual captchas. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2004. vol. 2, pp. 23–28. IEEE

(2004)

[40] Okada, M., Matsuyama, S.: New captcha for smartphones and tablet pc. In: 2012 IEEE Consumer Communications and Networking Conference (CCNC). pp. 34–35. IEEE (2012)

[41] Shen, Y., Ji, R., Cao, D., Wang, M.: Hacking chinese touclick captcha by multi-scale corner structure model with fast pattern matching. In: Proceedings of the 22nd ACM international conference on Multimedia. pp. 853–856. ACM (2014)

[42] Shirali-Shahreza, M., Shirali-Shahreza, S.: Drawing captcha. In: 28th International Conference on Information Technology Interfaces, 2006. pp. 475–480. IEEE (2006)

[43] Xu, Y., Reynaga, G., Chiasson, S., Frahm, J., Monrose, F., Oorschot, P.V.: Security and usability challenges of moving-object captchas : Decoding codewords in motion. In: 21st USENIX Security Symposium (USENIX Security) (2012)

[44] Yan, J., El Ahmad, A.S.: Breaking visual captchas with naïve pattern recognition algorithms. In: 23rd annual Computer Security Applications Conference - ACSAC '07. IEEE computer society (2007)

[45] Yan, J., El Ahmad, A.S.: A low-cost attack on a microsoft captcha. In: Proceedings of the 15th ACM conference on Computer and communications security - CCS '08. pp. 543–554. ACM, New York, NY, USA (2008)

[46] Yan, J., El Ahmad, A.S.: Usability of captchas or usability issues in captcha design. In: Proceedings of the 4th symposium on Usable privacy and security - SOUPS '08. pp. 44–52. ACM (2008)

[47] Yin, F., Wang, Q.F., Zhang, X.Y., Liu, C.L.: ICDAR 2013 Chinese Handwriting Recognition Competition (2013)

[48] Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K.: Attacks and design of image recognition captchas. In: Proceedings of the 17th ACM conference on Computer and communications security - CCS '10. pp. 187–200. ACM, Chicago, USA (2010)