Research on Deep Learning Techniques in Breaking Text-based Captchas and Designing Image-based Captcha

Mengyun Tang, Haichang Gao, *Member, IEEE*, Yang Zhang, Yi Liu, Ping Zhang and Ping Wang

***Abstract*—The ability of hackers to infiltrate computer systems using computer attack programs and bots led to the development of Captchas, or Completely Automated Public Turing Tests to Tell Computers and Humans Apart. The Text Captcha is the most popular Captcha scheme given its ease of construction and user friendliness. However, the next generation of hackers and programmers has decreased the expected security of these mechanisms, leaving websites open to attack. Text Captchas are still widely used because it is believed that the attack speeds are slow, typically two to five seconds per image, and this is not seen as a critical threat. In this paper, we introduce a simple, generic, and fast attack on text Captchas that effectively challenges that supposition. With deep learning techniques, our attack demonstrates a high success rate in breaking the Roman-character based text Captchas deployed by the top 50 most popular international websites and three Chinese Captchas that use a larger character set. These targeted schemes cover almost all existing resistance mechanisms, demonstrating that our attack techniques are also applicable to other existing Captchas. Does this work then spell the beginning of the end for text-based Captcha? We believe so. A novel image-based Captcha named Style Area Captcha (SACaptcha) is proposed in this paper, which is based on semantic information understanding, pixel-level segmentation and deep learning techniques. Having demonstrated that text Captchas are no longer secure, we hope our proposal shows promise in the development of image-based Captchas using deep learning techniques.**

***Index Terms*—Captcha, text-based, security, deep learning, convolutional neural network, image-based.**

I. Introduction

CAPTCHA stands for "Completely Automated Public Turing Test to Tell Computers and Humans Apart". Since it was proposed by Louis Von Ahn et al. in 2003[1], Captchas have been widely used in commercial applications to provide security against malicious computer programs and bots [2]. Captchas automatically generate and evaluate a test that is difficult for a computer to solve, but easy for humans. If the success rate of solving a Captcha for humans reaches 90% or higher and computer programs only achieve a success rate of less than 1%, the Captcha can be considered good [3]. Multiple variations have been proposed, but they always use the same fundamental idea: show users an image and request that they conduct a recognition task.

The authors are with the Institute of Software Engineering, Xidian University, Xi'an, Shaanxi, 710071, P.R.China. (e-mail: hchgao@xidian.edu.cn).

This work was supported in part by the National Natural Science Foundation of China under Grant 61472311 and in part by the Fundamental Research Funds for the Central Universities.

Manuscript received April 19, 2005; revised August 26, 2015.

Since its introduction, the text Captcha has been the most widely deployed Captcha scheme [4]. However, current research threatens the security of existing text Captchas, e.g., [3], [5], [6]. A large range of early simple text Captcha schemes deployed in the wild have been broken, such as Google, Yahoo!, and Microsoft, among many others. Captcha designers typically learn from previous failures to design Captchas with increased security and usability. Kumar suggested that the robustness of text Captchas should be measured by the difficulty of segmenting characters from Captcha images rather than recognizing what each character is, because it is easy for powerful classifiers, e.g., convolutional neural networks (CNNs), K-nearest neighbor (KNN) and support vector machine (SVM), to recognize rotated or warped characters [7]. This has been referred to as the segmentation-resistance principle, and it has become the footstone for designing text Captchas. Current text Captchas are much more sophisticated than previous models. To enhance security, Captcha designers tried to add various novel resistance mechanisms to existing text Captchas, e.g., crowding characters together (CCT), noise arcs, complicated backgrounds, hollow schemes and two-layer structures. However, all of these resistance mechanisms seem to be ineffective, as studies [8], [9], [10], [11] have demonstrated. Some more-recent researchers have claimed that they can break a large group of Captchas with a variety of design features in a single step [12], [13].

Clearly, these attacks pose a realistic threat to current text Captchas. There is now a general agreement that they are no longer secure. However, text Captchas are still widely used because the attacks are slow, typically taking two to five seconds per image. In this paper, we propose a simple, fast and effective attack with deep learning techniques. It contains three main steps: pre-processing converts a color Captcha image to black-and-white and removes noise arcs or complicated backgrounds; color filling segmentation (CFS, introduced in [6]) is used to select single characters or to simply divide a Captcha into equally distributed segments according to the number of characters it contains; and recognition utilizes deep CNN to determine what each single character is, and it combines the recognition results as the final result.

We tested our attack on the text Captchas deployed by the top 50 most popular websites according to the Alexa ranking to evaluate the security. These real-world Captchas cover all resistance mechanisms, including Google's reCAPTCHA, which is derived from Google Street View, character isolated schemes, hollow schemes, CCT schemes and other schemes

with noise arcs or complicated backgrounds. Our attack has achieved success rates ranging from 10.1% to 90%, with an average attack speed of 0.03 to 0.65 seconds. Judged by the commonly used criteria presented in [3], this attack has successfully broken all these targeted Captcha schemes at a speed that allows us to claim it as a real-time attack. Although various anti-segmentation mechanisms were adopted or these schemes and our rough segmentation method cannot precisely segment a Captcha image into individual characters, our attack still achieved high success rates. This proves that the segmentation-resistance principle may no longer be applicable.

The most widely used text Captchas are usually based on English letters and Arabic numerals, which are limited to 62 character categories (26 uppercase letters, 26 lowercase letters and 10 digits). An additional type of text Captcha was developed from large-alphabet languages such as Chinese, Japanese and Korean. For example, Chinese Captcha, which has billions of users, is much more complicated than the commonly used Roman character-based Captchas. There are approximately 3755 commonly used Chinese characters, which makes the solution space larger than traditional text Captchas using English letters and Arabic numerals. This paper also analyzes the security of such large-alphabet Captchas. We collected three Chinese Captchas from Baidu and QQ as representatives and tested our attack on them. The success rates we obtained vary from 28.6% to 93.0%, indicating that large-alphabet Captchas are equally unsecure.

We propose an image-based Captcha named Style Area Captcha (SACaptcha) that is based on the neural style transfer technique. To pass the test, users are required to click foreground style-transferred regions in an image based on a brief description. Unlike earlier image-based Captchas, SACaptcha relies on human understanding of semantic information and pixel-level segmentation, which seems to be more difficult for machines to solve. With neural style transfer, labels for images will be unnecessary, and any image can serve as the input to automatically generate a Captcha. To test its usability and robustness, we conduct user study to evaluate human performance and try to attack it using three state of the art techniques. We think it is a positive attempt on applying deep learning techniques to Captcha design.

Unlike previous work, which only uses deep learning as a recognition engine for individual characters, we also utilize it to enhance the security of the image-based Captcha. Our work provides the evidence that deep learning techniques can be used not just as an attack tool to break Captchas but can also be leveraged to improve their security.

This paper provides a comprehensive analysis of text Captchas. It analyzes all resistance mechanisms used in text Captchas, presents the methods to break them and evaluates the security of both traditional Roman character-based Captchas and the newly emerged large-alphabet Captchas. Finally, based on this thorough analysis, we conclude that existing text Captchas are not robust enough to resist automatic attacks using optimized image pre-processing algorithms, advanced deep learning techniques and improved hardware equipmebluent. It is necessary to develop new Captchas with increased security to resist deep learning-based attacks.

The remainder of this paper is organized as following. Section II provides a review and an analysis of real-word text Captchas, and Section III evaluates the robustness of the most commonly used Roman character-based Captchas. In Section IV, we conduct a further study analyzing the security of Captchas using large character sets, such as Chinese Captchas, we verify the effect of network depth on our attack, and we compare it with prior attacks. A novel image-based Captcha we called SACaptcha is proposed in Section V, and Section VI concludes the paper.

II. A Survey about Text Captchas

From its inception, the text Captcha has played an important role in enhancing Internet security. It is the earliest and the most popularly deployed Captcha scheme, especially those schemes based on English letters and Arabic numerals. This widespread usage may be due to several factors [14], [7]: from the user's point of view, a Captcha is only a text recognition problem, and it is intuitive to users worldwide; for a Captcha deployer, the cost of generating a text Captcha is lower than other Captcha alternatives, e.g., image-based and audio-based Captchas.

The original text Captcha forms are relatively simple. For instance, an early Captcha scheme deployed by Yahoo! for its free email services just asks users to read a distorted word. Current Captchas are much more sophisticated, with various resistance mechanisms to enhance their security. We summarize the most commonly used resistance mechanisms as follows:

Character isolated. The character isolated Captcha scheme might be the simplest; see Figure 1 (a). CFS [6] is able to extract single characters easily, and powerful classifiers work well in recognizing these extracted characters.

Rotation and warping. Rotation and warping are the earliest and most widely used resistance mechanisms. Rotation constructs characters in a random angle, whereas warping

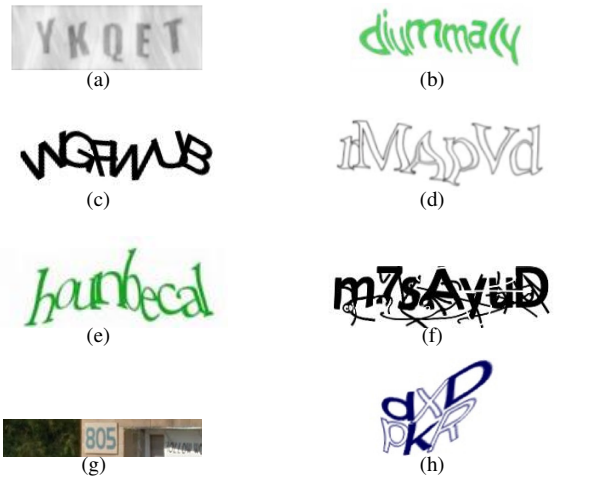


Fig. 1. Sample Captchas using various resistance mechanisms: (a) Character isolated, (b) Rotation and warping, (c) Overlapping, (d) Hollow scheme, (e) Varied Captcha string length, (f) Noise arcs, (g) Complicated background, (h) Two-layer structure.

waves the local characters or the whole Captcha string. Figure 1 (b) provides a sample Captcha. Essentially, rotation and warping increase the variations of characters, making the recognition task more difficult. However, it has already been proved that modern classifiers are able to easily recognize these rotated and warped single characters [7].

Overlapping. Overlapping removes the space between adjacent characters to crowd characters together, making it more difficult for computers to detect where each character is, as Figure 1 (c) shows. It is an effective application of the segmentation-resistance principle. However, several attacks have been proposed to solve this Captcha scheme, e.g., [12], [13]. Previous work [12] analyzed all possible ways of segmenting a Captcha and then used a classifier to pick the "best shard" among the remaining segments. [13] introduced a method based on Gabor filters to extract character strokes.

Hollow scheme. The main feature of a hollow Captcha is that it utilizes contour lines to form connected characters in order to improve security and usability simultaneously (see Figure 1 (d)). Unfortunately, attacks have demonstrated that hollow schemes are not as effective as thought in enhancing the security of Captchas. A single-step method reported in previous work [8] has broken a large group of hollow Captchas.

Varied Captcha string length. Most Captcha schemes use a varied Captcha string length (see Figure 1). This method increases the solution space of a Captcha. However, the dynamic programming (DP) algorithm introduced in [13] effectively solves Captchas with a varied Captcha string length.

Noise arcs. Adding noise arcs is another way to make characters connect with each other, as Figure 1 (f) illustrates. We classify noise arcs into two categories: thin and thick. For thin noise arcs, it is usually not necessary to change the attack process, as it has little effect on recognition. [15] noted that even with heavily added thin noise arcs, current deep learning algorithms still have the ability to recognize these characters. As for thick noise arcs, advanced pre-processing techniques are able to easily remove them. Erosion and dilation are commonly used techniques in noise removal.

Complicated background. To confuse the detection of characters, several Captcha schemes embed characters into a complicated background, e.g., a street view. The most well-known scheme is a version of Google's reCAPTCHA; see Figure 1 (g). It deploys Street View images that contain house numbers as its Captcha. However, Google itself broke this Captcha by using a deep convolutional neural network [16].

Two-layer structure. The two-layer structure was first introduced in Microsoft's Captcha in 2015; see Figure 1 (h). It can be regarded as a vertical combination of two traditional single-layer Captchas. The two-layer Captcha appears to be a more secure scheme. However, the novel envelope algorithm introduced in [11] can be used to precisely segment a two-layer Captcha into two single-layer Captchas, upon which common attacks can be utilized to break it.

In summary, these resistance mechanisms can be classified according to two aspects: anti-recognition and anti-segmentation. In practice, a Captcha scheme always combines several resistance mechanisms to guarantee its security.

Captcha designers have worked to design more-effective means to increase the difficulty for computers to solve text Captchas. However, existing image processing techniques and computer vision algorithms expose weaknesses in them and reduce their effectiveness. The increasing complexity of the Captcha schemes can increase security, but it also decreases usability for the end-user. With this loss of security, we can postulate that the era of text Captcha may be drawing to a close. To analyze this proposition, we conduct a comprehensive analysis of text Captchas.

III. Attack Process

In this section, we introduce an extremely simple, effective and fast attack on text Captchas. To evaluate the effectiveness of our attack, we test a large group of real-world text Captchas with distinct resistance mechanisms. We choose those deployed by the top 50 most popular websites according to the Alexa ranking, including Google, Baidu, Wikipedia, QQ, Microsoft, Sina, Weibo, Yandex, PayPal and Apple. In total, we collected 11 Captcha schemes from these websites. Several websites use the same Captcha scheme (e.g., Google, Facebook, YouTube and LinkedIn et al. all use reCAPTCHA), and some websites deploy other Captcha alternatives but not text Captchas (e.g., Instagram uses short message verification). The sample Captcha images we aim to test are shown in Figure 2, and Table I summarizes the resistance mechanisms each Captcha scheme has adopted. As shown in Table I, our targeted Captchas cover all popular resistance mechanisms and represent a wide spectrum of designs.

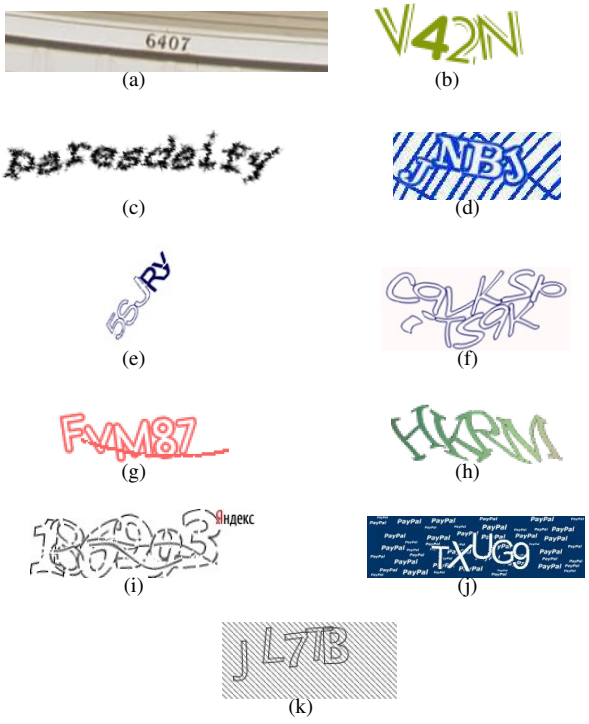


Fig. 2. Sample images of targeted schemes: (a) reCAPTCHA, (b) Baidu, (c) Wikipedia, (d) QQ, (e) Microsoft's single-layer scheme, (f) Microsoft's two-layer scheme, (g) Sina, (h) Weibo, (i) Yandex, (j) PayPal, (k) Apple.

Our attack contains three main steps: pre-processing, segmentation and recognition. Pre-processing converts a color

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2821096, IEEE Transactions on Information Forensics and Security

TABLE I
RESISTANCE MECHANISMS ADOPTED BY TARGETED CAPTCHA SCHEMES

	Website	Character isolated	Rotation and warping	Overlapping	Hollow scheme	Varied Captcha string length	Noise arcs	Complicated background	Two-layer structure
reCAPTCHA	google, facebook, youtube, linkedin, twitter, blogspot, google.co.in, wordpress et al.	√	√			√		√	
Baidu	baidu.com, hao123.com	√	√						
Wikipedia	wikipedia.org	√	√						
QQ	qq.com		√	√	√			√	
Microsoft (single-layer scheme)	live.com		√	√	√	√			
Microsoft (two-layer scheme)	bing.com		√		√	√			√
Sina	sina.com.cn		√	√	√	√	√		
Weibo	weibo.com		√	√					
Yandex	yandex.ru		√	√	√		√		
PayPal	paypal.com	√		√				√	
Apple	apple.com			√	√			√	

Captcha image into black-and-white, removes complicated backgrounds or divides a two-layer Captcha into two single-layer Captchas; segmentation picks up each isolated character or roughly segments a Captcha image into equally distributed segments according to the number of characters it contains; and recognition utilizes CNN as the classifier to identify these single characters as the final result. In the following, we explain the details of our attack and evaluate its effectiveness.

A. Attack details

1) Pre-processing: Pre-processing clarifies the background in order to obtain the characters' area and to facilitate the subsequent segmentation. For most Captchas we aim to test, e.g., Baidu, Wikipedia, Weibo and Yandex, we convert a rich-color image to black-and-white using Otsu's threshold method [17]. For other Captcha schemes, the adoption of rotation, noise arcs, a complicated background or a two-layer structure will lead to an unclear location of the text area and will have a negative effect on later segmentation. Therefore, we treat them during pre-processing as follows:

Rotation. Microsoft's single-layer Captcha scheme is heavily rotated, making the segments created by later segmentation too thin to contain enough information for each character. Therefore, we rotate the original Captcha images to upright. The pixels at the very bottom and at the right end in a Captcha image are identified in order to determine the angle, from which we can determine the rotation required to create the upright characters.

Noise arcs. The Captchas deployed by Yandex and Sina contain noise arcs. The main difference between the schemes is that the noise arcs in the Yandex scheme are within the text area, whereas Sina's extend past the text area. We remove the noise arcs in the latter case, since they cause these obtained segments to contain noise information. The noise arcs in the Sina scheme are the same color, so we detect the color of the pixels in each column, and if a column only contains one color, we set the non-white pixels in this column to white.

Complicated background. The reCAPTCHA, QQ, PayPal and Apple schemes all use a complicated background mechanism. reCAPTCHA is the most difficult scheme to break, as it uses real-world street views. We binarize the image and remove the noise blocks connected with the edge of the image. For QQ scheme, we do not remove all the background information, but we locate the text area according to the projection method introduced in [10]. This requires counting the number of non-white pixels in each column and each row and then removing all non-white pixels in these columns and the rows whose number of non-white pixels is smaller than a certain threshold. This threshold is determined by analyzing a small sample set of data. The backgrounds of PayPal and Apple can be easily deleted by removing tiny noise blocks and single-direction lines.

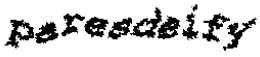
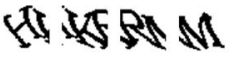
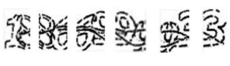
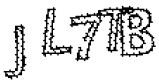
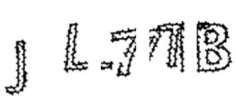
Two-layer structure. The second version of Microsoft's Captcha is two-layered. The attack for this scheme is to first segment the two-layer Captcha into two single-layer Captchas by using the envelope algorithm proposed by Gao et al. [11].

The pre-processing results of each scheme are listed in Table II. Existing advanced image pre-processing techniques are robust enough to remove the interference in all cases.

2) Segmentation: This step divides a Captcha image into mini segments that each contains a single character. According to the relationship between adjacent characters, we classify the Captchas into two categories here: the character isolated scheme and the CCT scheme. The characters in the character isolated scheme are separated, whereas the characters of the CCT are connected to each other. We leverage different approaches to obtain mini segments for these two schemes.

Character isolated scheme. For character isolated schemes, in which all characters are single connected areas, e.g., reCAPTCHA and Wikipedia, we simply utilize the CFS algorithm [6] to identify each individual character. Note that

TABLE II
ATTACK PROCESS

scheme	Sample image	Preprocessing result	Segmentation result	Recognition result
reCAPTCHA				“6407”
Baidu				“V42N”
Wikipedia				“paresdeify”
QQ				“JNBJ”
Microsoft (single-layer scheme)				“5SJRy”
Microsoft (two-layer scheme)				“c9LkspiTs9k”
Sina				“FVM87”
Weibo				“HKRM”
Yandex				“186993”
PayPal				“TXUG9”
Apple				“JL7TB”

for characters like ‘ *i* ’ and ‘ *j* ’ in Wikipedia that consist of two connected areas, we combine them into a single character according to their location relationship. That is, if one connected area is entirely on the top of another connected area, we combine them.

CCT scheme. For schemes with connected characters, we divide the Captcha image into equally distributed segments according to the number of characters it contains. However, some Captcha schemes use a varied Captcha string length, and we cannot know how many characters each Captcha image contains in advance. Using Microsoft’s single layer Captcha as an example, the string length ranges from 5 to 7. Thus, the first challenge of segmentation is to determine the number of characters in a Captcha image.

We regard this as a classification problem, and we utilize a CNN model to conduct this classification task. For example, determining the number of characters in Microsoft’s single-layer Captcha is a three-classification problem, that is, its Captcha images can be classified into three categories: images with 5 characters, images with 6 characters and images with 7 characters. The width and height of the text area provide significant information for estimating Captcha string length. To maintain the information and guarantee a standard input to the network, we remove the surrounding blank space in each Captcha image and relocate it to the center of a relatively larger blank image as the input of the network. The output is the estimated Captcha string length of this image. The CNN model we utilized here is same as the one we used for recognition and will be illustrated in detail later.

For the CCT schemes with a varied Captcha string length, our method has achieved an average accuracy of 91.5% for estimating how many characters each image contains. Compared

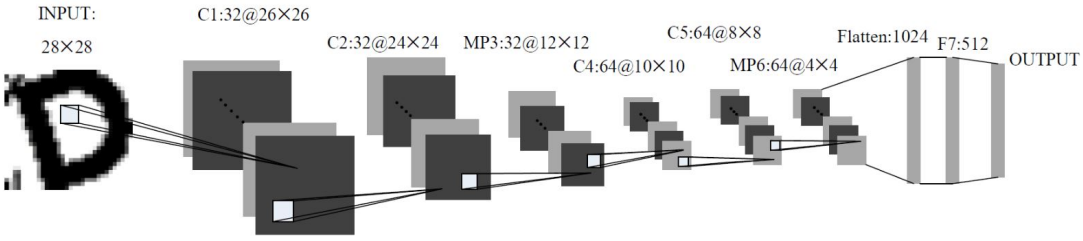


Fig. 3. Our improved LeNet-5: where **C** denotes the convolution layer, **MP** denotes the max-pooling layer, and ***n*@*a*×*a*** denotes that this layer generates ***n*** feature maps with a size of ***a*×*a***; **Flatten: 1024** denotes that the flatten layer generates a vector with 1024 factors, and **F7: 512** denotes that the 7^{th} layer is a fully connected layer containing 512 hidden neurons.

to the methods in [8] and [13], which determine the number of characters by dynamically selecting a path according to the recognition results, our method is much simpler and more effective.

Then, we vertically segment each Captcha image into c equally distributed segments, where c is the number of characters it contains, as estimated by the CNN. To guarantee that each segment contains the main information of a character, we slightly overlap them, since the width of each character is varied. Table II lists the segmentation results of each Captcha scheme. Note that we only use the CNN model to estimate the Captcha string length of the Captcha images in the test set. For the training set, we divide a Captcha image according its real number of characters.

3) Recognition: Finally, we modify the most fundamental CNN architecture, LeNet-5 [18], which was originally designed for hand-written and machine-printed character recognition, and use it as our recognition engine to recognize mini segments received by segmentation. The recognition results of all segments are combined as the final result (see the last column of Table II).

LeNet-5 uses a feed-forward architecture. It takes a character image as input, continues with a linear succession of layers and then outputs the category this character image belongs to. The original LetNet-5 contains three convolutional layers, two subsampling layers and two fully connected layers. To receive higher-level features, we modify LeNet-5 by adding an extra convolutional layer. Figure 3 illustrates this.

The mini segments are normalized to 28×28 pixels as the inputs of the network architecture. The network was trained using stochastic gradient descent (SGD), described in [19]. The original learning rate was set to 0.001. In total, we trained 50 epochs.

Although there have been many deeper network architectures for image classification, and although our CNN model is probably not optimal, we did not optimize the network, for one simple reason: our network turned out to be effective enough at recognizing the segments extracted from Captcha images. We will investigate the effect of network depth on our attack results in Section IV.

B. Experiment results

We implemented our attack and tested it on all targeted schemes. The Captcha images were processed in C# on a desktop computer with a 3.3 GHz Intel Core i3 CPU and 2 GB RAM. Our modified LeNet-5 is based on a deep learning framework called Caffe [20], which was developed by a UC Berkeley team. To speed up the training process, we trained it with an NVIDIA TITAN X GPU and 64 GB RAM.

Data collection. For each scheme listed in Table II, we collected 3400 random Captcha images from the corresponding websites, 2000 of which were used as the training set, 400 were used as the validation set, and the remaining 1000 were used as the test set. The collection of our data was carried out from 2016 to 2017.

Success rate. Our attack evaluation process follows previous practice. Table III summarizes our attack results for each Captcha scheme. Our success rates range from 10.1% to 90.0%. A standard goal for Captcha robustness is to prevent automated attacks from achieving a success rate higher than 1% [3]. Therefore, our simple attack broke all of the targeted schemes. Apart from reCAPTCHA, the success rates achieved by our attack are higher than 47%, far beyond the standard criterion. It turns out that all text Captchas deployed by the top 50 most popular websites in the world are not as secure as expected.

TABLE III
ATTACK RESULTS

Website	Success rate	Average attack speed (s)
reCAPTCHA	**10.1%**	0.50
Baidu	57.0%	0.15
Wikipedia	**90.0%**	0.46
QQ	47.2%	0.20
Microsoft (single-layer scheme)	50.9%	0.44
Microsoft (two-layer scheme)	65.8%	**0.65**
Sina	75.0%	0.14
Weibo	51.2%	0.04
Yandex	56.0%	**0.03**
PayPal	67.4%	0.41
Apple	47.3%	0.38

The success rate against reCAPTCHA is the lowest, since it uses street views containing house numbers as the Captcha. These complicated street view scenarios make the extraction of a house number extremely difficult, and there is a wide variation in the choice of fonts in real-world house numbers. Both of these reasons explain the lower success rate of our attack.

Attack speed. We ran all test images ten times and calculated the average attack speed of each scheme. As shown in Table III, our average attack speed ranges from 0.03 to 0.65 seconds, indicating that this is a real-time attack. Our data shows that humans identify the Captcha more slowly than our attack method. Prior work [21] reported an average solving time for some real-world Captchas of more than 46 seconds.

To sum up, it is clear that our attack is extremely effective, fast and poses a realistic threat to all text Captchas. What is also important to mention is that our work undermines the idea presented in the segmentation-resistance principle proposed in [7]. Even though our rough segmentation method used for most Captcha schemes cannot precisely segment all characters from each other, with deep learning techniques, our attack can still work very effectively on targeted schemes. The segmentation-resistance principle used in these schemes seems to be ineffective. A powerful recognition engine and a large amount of data can be used to compensate for the failure of segmentation. The segmentation-resistance principle may be not appropriate for current Captcha design, especially in the face of deep learning-based attacks.

IV. Discussion

A. The security of large-alphabet Captchas

The above work has shown the vulnerability of commonly used text Captchas based on English letters and Arabic numbers. Such Roman character-based Captchas, at most, contain 62 categories of characters. In fact, most Captchas use a much smaller character set than 62, since some letters are too similar for users to distinguish. Another type of text Captcha has emerged that is derived from large-alphabet languages, e.g., Chinese, Japanese and Korean. These large-alphabet Captchas usually have a large character set, which appears to increase the solution space and promise better security. Chinese Captcha, for instance, has thousands of commonly used characters. This makes the recognition task extremely difficult, especially for traditional machine learning algorithms.

To evaluate the security of large character set Captchas, we conduct a detailed analysis of Chinese Captcha as a representative. It is generally thought that only users who know or have studied Chinese can recognize Chinese Captchas. However, previous work [15] has shown that Chinese Captchas can also be designed in such a way that they are universal, meaning that they can be used by users who have never learned Chinese. Baidu and QQ are selected as the representatives, as they are the largest search engine and the most widely used communication tool in China respectively, with Alexa ranking them as the top 4 and 7 most popular websites in the world, respectively (see Table IV). The QQ and Baidu Tieba schemes both require users to click bottom characters in order to match the sequence of upper characters, while the Baidu Registration scheme needs to input the two characters shown in the Captcha image to pass the test.

In the QQ scheme, the characters' pixels are black and are surrounded by white pixels. Using this characteristic, we easily extracted the upper and lower characters from the complicated background. For Baidu Tieba and Baidu Registration, the projection method [10] is used to remove the noise arcs that extend the text area. After removing the noise arcs, the CCT sections of these two Baidu schemes are divided into equally distributed segments, as Table IV shows.

We build a new CNN architecture to serve as the recognition engine of Chinese characters, as Figure 4 shows. It contains five convolutional layers, three max-pooling layers and two fully connected layers. The convolution filters of the first two convolutional layers are 11×11 and 5×5, respectively, and each of them is followed by the rectified linear unit (ReLU) [22] non-linear activation function and by normalization. The following convolutional layers all have a filter of 3×3, and zero padding is applied to each. Apart from the stride of 4 in the first convolutional layer, the other four convolutional layers all have a stride of 1. Each max-pooling layer has a kernel of 3×3, and the stride is 2. Similar to previous experiments, the SGD algorithm [19] is used to train the network, and the initial learning rate is 0.001. In total, we finish the training after 50 epochs. The implementation framework and environment are the same as in previous experiments.

Unlike the common text Captchas, we mined 1000 random Captcha images from the corresponding websites as the test set. For the training and validation sets, we mimicked the shapes of the Chinese characters contained in these Captchas to automatically generate single-character images to train and select the model. The 3755 most commonly used Chinese characters from the China National Standard GB2312 were taken to make the samples. For each character category, we generated 200 single character images with random rotation. For the Baidu Registration scheme, we added random noise arcs to increase their variations. A total of 180 single character images were used for training, and another 20 were used for validation. In total, the training set contained 675,900 single-character images, while the validation set contained 75,100 single-character images. All images were normalized to 108×108 pixels to feed to the network.

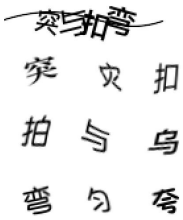
We achieved a success rate of 93.0%, 32.2% and 28.6% against QQ, Baidu Tieba and Baidu Registration, with an average attack speed of 2.816 seconds, 1.408 seconds and 0.108 seconds, respectively. Our analysis demonstrates that even with a much larger solution space, Chinese Captchas are still vulnerable to attacks based on deep learning techniques. Our results are equally applicable to Captchas derived from other large-alphabet languages, such as Japanese and Korean.

B. The effect of network depth

CNNs were originally introduced over 20 years ago [15]. However, only in the past few years have they been dramatically improved and become the main machine learning method for solving various visual computer problems.

Typically, it is considered that the deeper the network is, the higher the level of extracted features will be. To evaluate how the depth of CNNs impacts attack results, we took the Wikipedia and Sina schemes as representatives of character isolated and CCT schemes, respectively, to conduct a further study. Based on the CNN model proposed in Section

TABLE IV
ATTACK PROCESS AND ATTACK RESULTS OF CHINESE CAPTCHAS

Scheme	Sample image	Segmentation result	Success rate	Average attack speed (s)
QQ		Top: 寂然无声 Bottom: 无声寂然	93.0%	2.816
Baidu Tieba		Top: Bottom:	32.2%	1.408
Baidu Registration			28.6%	0.108

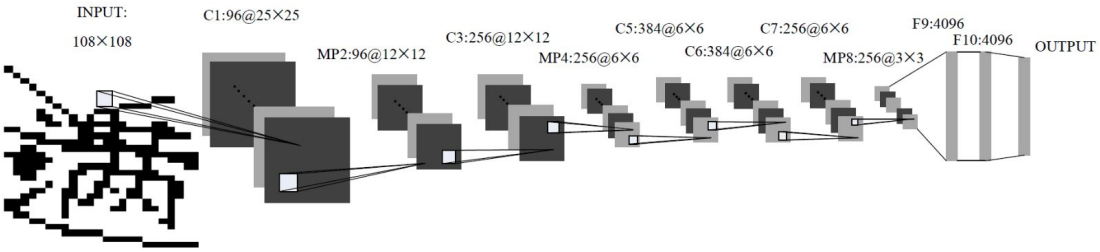


Fig. 4. CNN model used for recognizing Chinese characters: where **C** denotes convolution layer, **MP** denotes max-pooling layer, and ***n*@*a*×*a*** denotes that this layer generates ***n*** feature maps with a size of ***a*×*a***; **Fi: m** denotes that the i^{th} layer is a fully connected layer and contains m hidden neurons.

III, we built another four CNN models with 2, 3, 5 and 6 convolutional layers as the recognition engines. Table V shows the success rates achieved by the Wikipedia and Sina schemes. We highlight the original results achieved by the model proposed in Section III in red, with the best results in **boldface**.

TABLE V
THE EFFECT OF NETWORK DEPTH ON SUCCESS RATE

	2	3	4	5	6
Wikipedia	86.3%	89.8%	**90.0%**	89.4%	85.4%
Sina	59.7%	**76.7%**	75.0%	65.3%	64.0%

The success rate improves in the beginning and then starts to drop as the depth increases. This trend holds for both Captcha schemes. The reason is that the features used for character recognition may not be higher the better they are. Moreover, the features needed to obtain the best result vary with different Captcha schemes, e.g., Wikipedia achieved the best result when using the model with four convolutional layers, whereas Sina has the highest success rate using the model with three convolutional layers. Although the success rate varies with the change of network depth, the difference is not large, and all of these attacks successfully broke the Wikipedia and Sina schemes. Therefore, we selected a relatively simpler network with fewer layers as our final recognition engine.

C. Comparison with prior attacks

Text Captcha is still the most widely used Captcha form. Most of the early schemes have been broken by previous efforts, including [23], [24], [5]. In [23], Mori and Malik used sophisticated object recognition algorithms to break Gimpy and EZ-Gimpy, with success rates of 33% and 99%, respectively. In 2004, Moy et al. [24] proposed distortion estimation techniques to attack EZ-Gimpy, with a success rate of 99%. In [5], Yan et al. broke a number of Captchas using a pixel counting method. However, these attacks are neither generalizable nor robust in detecting slight changes in the Captchas, whereas our attack has broken a large group of text Captchas. Other *ad hoc* methods include [25], [26], [27], [6], of which [25], [26], [27] reported various attacks on previous text versions of reCAPTCHA, whereas [6] presented attacks against Microsoft's Captcha.

In 2010, Li et al. [28] reported a comprehensive analysis of e-banking Captchas. They built a set of image processing and pattern recognition tools, such as k-means clustering and morphological operations, and successfully broke three e-banking Captchas for transaction verification and 41 e-banking Captchas for login, with success rates either equal to or close to 100%. For each Captcha scheme, they combined different tools to attack. Therefore, it is actually a toolbox method. Decaptcha is another toolbox-based attack, proposed by Bursztein et al. [3]. They successfully attacked 13 of the 15 Captchas from popular websites and claimed that their method is generic. In fact, Decaptcha uses a five-stage pipeline: pre-processing, segmentation, post-segmentation, recognition and post-preprocessing. Similar to the attack reported by Li et al., various techniques were used for different Captcha schemes in each stage of Decaptcha. Our method uses special techniques for a few complicated Captcha schemes during pre-processing, but for the segmentation and recognition processes, we used unified algorithms on all Captcha schemes. Our method is much simpler than theirs. Decaptcha also failed to break reCAPTCHA and hollow Captcha, whereas our method can.

Gao's team first introduced a single-step attack at CCS'13 [8] that uses machine learning to solve the segmentation and recognition problems simultaneously. However, their method mainly focuses on hollow Captchas; it cannot break non-hollow schemes. The CFS algorithm they adopted to extract hollow fonts cannot extract solid fonts, so that their method cannot separate non-hollow characters that connect with each other. However, our method has broken both hollow and non-hollow schemes.

Other generic methods include [12] and [13]. The attack proposed by Bursztein et al. [12] is the second attempt to address segmentation and recognition simultaneously, and it successfully broke a group of text Captchas. Their method consists of four components: a cut-point detector, a slicer, a scorer and an arbiter. It analyzed all possible ways to segment a Captcha image and then used ensemble learning to identify among each sequence of segments the best possible one as the result. The success rates achieved by their method range from 5.33% to 55.22%. At NDSS' 16, Gao's team [13] reported another generic attack, which utilized Gabor filters to extract character components and then tries different combinations of adjacent components. Finally, when directed by a DP algorithm, the most likely combination is chosen as the final result. Both Bursztein's and Gao's methods are more complicated than ours. Although we simply used CFS to select single characters or roughly segmented a Captcha image into equally distributed segments, our method still achieved high success rates. Excluding reCAPTCHA, the attack success rates of the other schemes we tested are all higher than 47%, but only 3 of the 10 targeted schemes Gao's team tested achieved a success rate higher than 47%. What is most important to mention is that our attack speed is extremely fast. Our slowest attack on traditional text Captchas is 0.65 seconds, which is much faster than the fastest attack reported in [12] and [13]. To sum up, our attack is simple and generic, but effective and fast. We claim that it is a real-time attack.

Our work also tested the robustness of a two-layer Captcha deployed by Microsoft, similar to the scheme analyzed by Gao et al. in [11]. Again, the success rate achieved by our attack is much higher (65.8% vs. 28.0%), and it is also much faster (0.65 seconds vs. 12.56 seconds). We analyze the reasons as follows. Their method used a DP algorithm to select a path with the largest average confidence level as the final result, but this leads to numerous failures in estimating the Captcha string length; our attack instead utilizes deep learning techniques to determine the Captcha string length, achieving an accuracy of 91.5%. Additionally, their method includes many complex processes (e.g., using CFS to convert hollow characters to solid and using Gabor filters to extract character components), whereas our attack comprises three simple steps, increasing our attack speed. However, the envelope algorithm in [11] for the two-layer Captcha is indeed effective, and it provided a good basis for our attack.

Large-alphabet Captchas, as a newly developed Captcha form, are much more difficult to solve than traditional text Captchas, since they have a larger solution space. However, using Chinese Captcha as a representative, we also determined that it is vulnerable to our attack. Yan's team [15] also analyzed Chinese Captchas, but they only provided evidence that computers recognize individual Chinese characters well, whereas our attack has successfully broken real-world Chinese Captchas.

Deep learning techniques have become the main tools in analyzing the robustness of Captchas. Goodfellow et al. [16] proposed a method based on deep CNN to recognize multi-digit numbers from street view imagery, and with the use of a dataset containing tens of millions of transcribed street numbers, they achieved an extremely high accuracy (over 96%). In 2015, Colin Hong et al. [29] utilized a template-based method and a CNN-based method to break Microsoft's single-layer Captcha. Their results show that the CNN-based method performs much better. In 2016, Sivakorn et al. [30] successfully attacked the semantic image-based Captchas deployed by Google and Facebook with accuracies of 70.78% and 83.5% respectively. All of these only evaluated one or two Captcha schemes using deep learning techniques, whereas ours has carried out a more comprehensive anlysis which contains 14 Captchas. Le et al. [31] introduced a Captcha-breaking network combining CNNs and recurrent neural networks (RNNs). They used synthetic data to train the network. When tesing on the synthetic Captchas, they achieved high success rates ranging from 91.0% to 99.9%. But when testing on real-world Captchas used by Facebook and Wikipedia, they only achieved success rates of 81% and 42%, respectively. [32] proposed a network named Recursive Cortical Network (RCN) to break text Captchas and have successfully broken four schemes with success rates varing from 57.1% to 66.6%. They claimed it only needs a few clean individual characters to train the network, in fact, some parameters should be manually tuned to prepare these characters. Most importantly, their attack speed is 94 seconds, which is extremely slow.

D. Analysis

Our real-time effective attack has broken all text Captchas deployed by the top 50 most popular websites in the world

with high success rates and fast attack speeds. We also successfully attacked several Chinese Captchas, which use a large character set. Our results are also applicable to other Roman character-based Captchas and Captchas derived from other large-alphabet languages such as Japanese and Korean.

Obviously, Captcha designers have tried their best to propose various resistance mechanisms to improve the security of text Captchas, e.g., complicated backgrounds and two-layer structures. However, they are ineffective for the following reasons: first, various image pre-processing techniques proposed by early efforts can easily remove noise arcs, complicated backgrounds or other types of interference; second, with the development of deep learning techniques, advanced hardware equipment (e.g., GPU) and large datasets, character recognition has achieved extremely high accuracy, not only with Roman characters but also with Chinese characters, which have a large character set.

Therefore, we conclude that all previously proposed resistance mechanisms are vulnerable, and existing text Captchas are not as secure as expected. However, given Captcha's user friendliness, instead of discarding it completely, we are willing to develop more-effective ways to design text Captchas with higher security and better usability. Based on the segmentation-resistance principle [7], Captcha designers have focused extensively on how to make text Captchas more difficult for computers to segment a Captcha into single characters. Our work has demonstrated that even when characters are imprecisely segmented, our attack still achieves high success rates. Thus, we argue that Captcha designers should pay more attention to designing text Captchas that are more resistant to recognition, especially for deep learning-based attacks.

V. SACaptcha: A Novel Image-based Captcha

A. Other Captcha alternatives

Many other Captcha alternatives have been proposed as substitutes for traditional text Captchas, based on one fundamental idea: they are easy for humans to solve but remain difficult for computers, and they are easy to generate and evaluate.

The ASIRRA Captcha [33] (Figure 5 (a)), proposed at CCS' 07, is an early version of an image-based Captcha. It generates tests by displaying 12 images of cats and dogs from a database of three million pictures, and it requires users to select all the cat images from cats and dogs. It relies on the presumed difficulty of understanding the content of images. Unfortunately, ASIRRA has been shown to be vulnerable to attack. Golle [34] proposed an ASIRRA attack that extracts color and text features from ASIRRA images to train an SVM classifier to identify cats and dogs. The success rate was 10.3%.

The semantic image Captchas deployed by Google and Facebook can be considered variants of ASIRRA, as Figure 5 (b) and (c) show. These two Captcha schemes both use an extended set of object categories to increase the difficulty for computer programs to recognize these images. However, with deep learning techniques for extracting semantic information from images, Sivakorn et al. [30] successfully attacked the

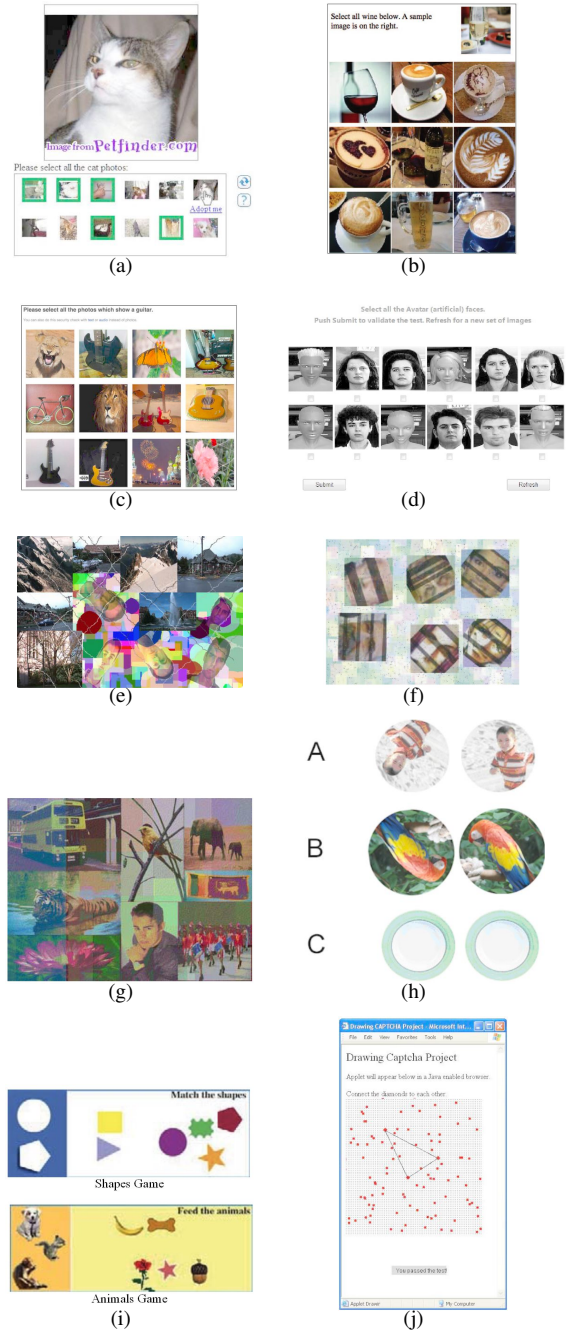


Fig. 5. Other Captcha alternatives: (a) ASIRRA, (b) Semantic image Captcha deployed by Google, (c) Semantic image Captcha deployed by Facebook, (d) Avatar Captcha, (e) FR-CAPTCHA, (f) FaceDCAPTCHA, (g) IMAGINATION, (h) What's up Captcha?, (i) DCG Captcha, (j) Drawing Captcha.

semantic image Captchas deployed by Google and Facebook with success rates of 70.78% and 83.5%, respectively. Similar schemes based on recognizing the semantic content of images also include three early image recognition Captchas introduced in [35].

There have also been trials of several Captchas based on human faces. Darryl D'Souza [21] proposed the Avatar Captcha, which asks users to identify avatar faces from a set of 12 images composed of a mix of human and avatar faces; see Figure 5 (d). These images come from a database containing human

and avatar faces, and all of them were converted to grayscale to prevent computer programs from breaking the Captcha by taking advantage of the varying color spectrum differences between human and avatar images. Avatar was broken by CNN with an extremely high success rate of 99% [36]. Goswami et al. proposed FR-CAPTCHA [37] and FaceDCAPTCHA [38]. The former requires users to find matching faces in an image, whereas the latter requires users to distinguish between actual human faces and animated human faces; Figure 5 (e) and (f) illustrate this. Unlike Avatar, the human face images presented in FR-CAPTCHA and FaceDCAPTCHA are rotated, distorted or combined with the background. Gao et al. [39] attacked FR-CAPTCHA and FaceDCAPTCHA with success rates of 23% and 48%, respectively. ARTiFACIAL [40] is a much earlier face-based Captcha scheme that allows users to first find the face in a complicated background and then click four eye corners and two mouth corners to pass the test. Zhu el al. [41] presented techniques to break ARTiFACIAL and achieved a success rate of 18.0%.

IMAGINATION [42] is a Captcha scheme that asks users to click the geometric center in a synthetic image; see Figure 5 (g). It was also involved in the analysis of image-based Captchas conducted by Zhu et al. [41]. They broke it with a success rate of 74.31%. Another image-based Captcha that should be mentioned is Google's Captcha based on image orientation [43], as Figure 5 (h) illustrates. This scheme provides a set of randomly rotated images and requests users to adjust them to an upright orientation. Aligning an image in an upright position is easy for humans but difficult for computer programs. However, a main limitation of this Captcha is that the source images used to generate these Captchas must be carefully selected, because the upright positions for some images may be ambiguous.?

The Dynamic Cognitive Game (DCG) Captcha (Figure 5 (i)) [44] is an interesting Captcha scheme. It challenges users to perform a game-like cognitive task in which they interact with a series of dynamic images. It makes Captcha solving a fun activity for the users and thus improves user experience. However, the major problem of game Captchas is that the cost of transmission is higher than traditional text Captchas.

The Drawing Captcha [45] distinguishes humans from computers by drawing lines with PDA pens. It shows users a screen with numerous dots, some of which are dissimilar from others, and users have to connect them to each other. Figure 5 (j) illustrates this. This Captcha scheme was broken by Lin et al. [46].

DeepCAPTCHA [47] is a newly proposed image-based Captcha scheme which was designed to resist deep learning attacks. It adds adversarial noise to the correctly classified images to generate adversarial examples [48], [49], [50] so that machines could be deceived. However, to our knowledge, applying adversarial examples to Captchas has two main issues: first, it works well on defending whitebox attacks, but its performance on blackbox attacks is not good, especially there has emerged a great many powerful deep learning tools and in most cases we cannot determine what models attackers is using; second, attackers also can collect images with adversarial examples to retrain their models to make them robust enough to recognize these changed images.

Other alternatives include motion-based NuCaptcha [51], which was broken by Xu's team [52] and Elie Bursztein [53], and audio Captcha, which was broken by Tamet [54].

B. SACaptcha

1) Overview: Image-based Captchas have become an active field in Captcha design. However, early proposed image-based Captchas have exposed various problems: some schemes require humans to manually select source images or add labels to images; some are based on a database, and if the database is compromised, they become vulnerable; some schemes incur a high transmission cost; and, most importantly, almost all of them have been proven to be unsecure.

To overcome these problems and try to enhance the security of Captchas, we propose a novel image-based Captcha named **SACaptcha** that is based on the neural style transfer technique [55], [56]. Neural style transfer is also a practice of deep learning techniques. It typically trains a deep CNN to reconstruct a stylized image y whose content is similar to the content image y_c, and the style is similar to the style image y_s.

SACaptcha shows a synthetic image with a size of 560×320 pixels in which some regions with different shapes are transferred using several different styles. It asks users to click foreground style-transferred regions according to a short description, detailing which shapes should be clicked. As Figure 6 shows, the left image is an original Captcha that asks users to click all foreground regions with circle, heart and pentagram shapes. The image on the right shows the answer by surrounding the shapes with red outlines. A disturbed foreground region is marked with a green outline. Essentially, solving this Captcha requires users to understand the semantic information (shapes of regions) contained in the image and to conduct a rough pixel-level segmentation.

We limit the number of foreground style-transferred regions in each Captcha image to 4 to 7, and we keep the maximum width and the maximum height of each region to less than 80 pixels. The shape of each region is randomly selected: it can be a rectangle, a triangle, a circle or other irregular shapes such as a heart, a leaf, a moon and so on. We also randomly choose the targeted shapes to generate a description from the shapes

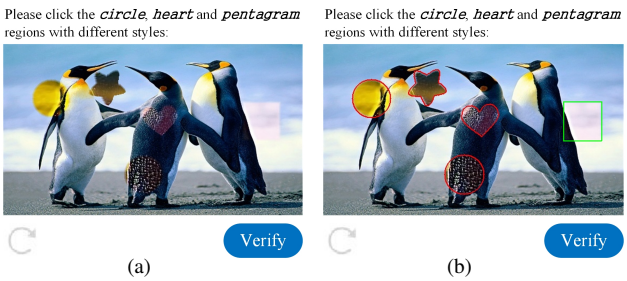


Fig. 6. SACaptcha: (a) A sample image: there are five foreground regions with shapes of circle, rectangle, pentagram and heart. (b) The answer to the sample image: a red outline marks all targeted regions that need to be clicked, and a green outline marks the disturbed region.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2018.2821096, IEEE Transactions on Information Forensics and Security

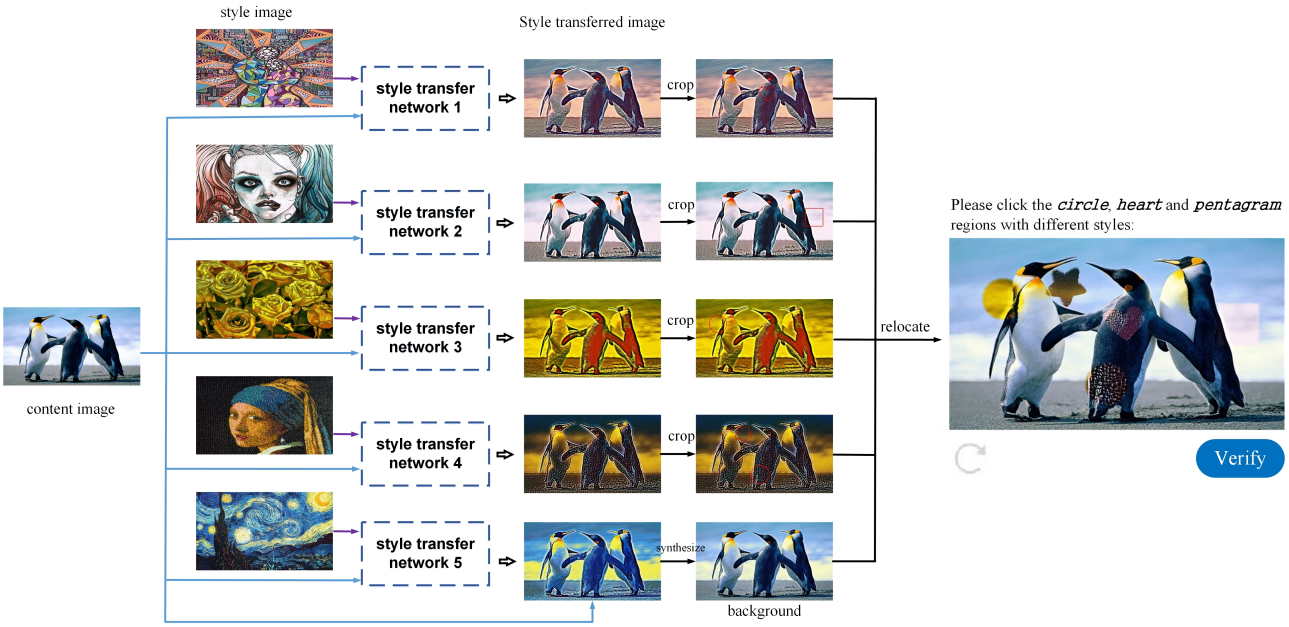


Fig. 7. The generation process of SACaptcha.

in the corresponding Captcha image. The number of targeted shapes is varied, but we expect that users will be required to click at least four regions.

2) Generation process: Figure 7 illustrates the generation process of SACaptcha, it includes four main steps:

- **Generate style-transferred images.** The network introduced in [57] has been improved to create style-transferred images. Several style transfer networks are pre-trained, and each is provided with a style. To generate a Captcha, we select an image as the content image and send it into some randomly selected pre-trained style transfer networks to generate style-transferred images, where one generates the synthetic background image and the others generate 4 to 7 foreground images. ?
- **Synthesize the background.** For the background, one of these style-transferred images is synthesized with the original image at a ratio of α, as equation 1 shows, where y denotes the style-transferred image, y_c denotes the original image (content image) and y_b denotes the final synthesized background image. Note that α is a random value with a range from 0.1 to 1.0 to make the background more varied. The larger α is, the more style information the synthetic background contains.

$$y_b = \alpha y + (1 - \alpha) y_c \tag{1}$$

- **Generate the Captcha.** We randomly crop regions with different shapes from other style-transferred images and relocate them in the synthetic background to generate a Captcha. To ease attacks based on edge detection, the edge of each region is blurred. ?
- **Generate a description.** Finally, we generate a brief description to guiding users on how to pass the test.

With pre-trained style transfer networks, it takes 0.688 seconds on average to automatically generate a Captcha image. Neural style transfer changes the features of the original image. Therefore, even if the attackers find the original image, the security of the SACaptcha is not compromised, because they do not know which style was used in a Captcha. Moreover, any image can be the resource for generating a SACaptcha, so we do not need to manually add labels to source images. In summary, the cost of generating an SACaptcha is very low.

3) Usability study: Usability and security are the most significant factors for Captcha designing. Thus, we carry out experiments to evaluate the usability of SACaptcha first. It involved 100 participants whose age ranges from 18 to 42. To investigate how the numbers of foreground shapes and styles of SACaptcha affect human perfomance, we test three versions of SACaptcha: the first one uses Captchas containing two simple forground shapes (circle and rectangle) and 11 styles; the second one uses Captchas with 25 foreground shapes but only two styles (one for the background and another for the foreground); while the last one uses 25 foreground shapes and 11 styles. For each test, every participant is required to complete 10 challenges at least, and we record the solving rate and the response time.

Table VI presents the usability study results of SACaptcha. It shows that, with the increase of the foreground shapes and styles, the solving rates of humans decrease, and the average response time rises. It takes 10.120 seconds at most for users to pass SACaptcha with a solving rate of 92.1%. It is widely accepted that a Captcha should be completed by humans no

TABLE VI
USABILITY STUDY RESULTS OF SACAPTCHA

	Version 1	Version 2	Version 3
Solving rate	94.4%	92.8%	92.1%
Average response time (s)	9.345	9.712	10.120

more than 30 seconds [2]. Besides, the lowest solving rate of SACaptcha achieved by humans is higher than that of some Captchas deployed in the wild [58], e.g., Captcha.net, Digg, Google, mail.ru, Microsoft and reCAPTCHA et al.. Therefore, the usability of SACaptcha is acceptable.

4) Security analysis: To evaluate the robustness of SACaptcha, we analyze the success rate of the random guess attack and utilize edge detection, objection detection and pixel-level segmentation techniques to attack it. We test the security on these three versions of SACaptcha introduced before. For each, 6000 SACaptchas are produced. Of these, 4000 are used as the training set, another 1000 are used as the validation set, and the remaining 1000 are used for testing.

Random guess attack. The success rate of the random guess attack on SACaptcha is less than $\frac{(80\times80\times4)}{(560\times320)} \times \frac{(80\times80\times3)}{(560\times320-80\times80)} \times \frac{(80\times80\times2)}{(560\times320-80\times80\times2)} \times \frac{(80\times80)}{(560\times320-80\times80\times3)} \approx 0.004884\%$, showing that a random guess attack is highly unlikely to pass the test.

Edge detection attack. Edge detection algorithms are always utilized to detect significant changes in an image. We also use it to determine the foreground regions in SACaptcha, because the difference between foreground and background is obvious. However, the edge detection algorithms provided by OpenCV (a library of programming functions mainly aimed at computer vision) can detect the contour lines of the objects contained in the image, but they fail to find the blurred edge of each foreground region, as Figure 8 (a) shows. Thus, we consider it a failed attempt.

Object detection attack. We attempt to detect the foreground style regions with object detection techniques, using Faster R-CNN [59]. It takes Captcha images as well as the shape and location information of each foreground region as the inputs to train the network. The location information contains the top-left pixel's coordinate, width and height of the bounding rectangle of each region. During the test process, we input a test Captcha into the pre-trained network. It returns the location and shape information of each predicted region as well as an object score that measures membership to style regions vs. the background; see Figure 8 (b).

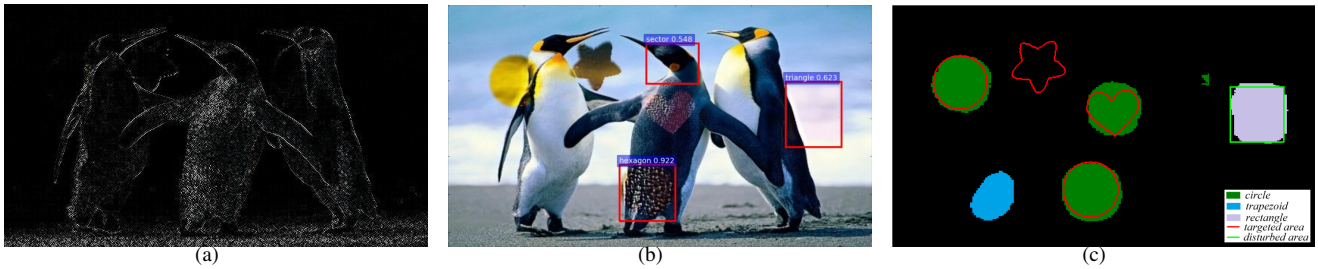
We follow a rough evaluation principle to compute the location accuracy rate and the final success rate: if the predicted region with an intersection-over-union (IoU) overlap with the ground-truth region is higher than 50% and the predicted shape information is correct, we consider it as having successfully detected the region we aimed for; if all targeted regions are successfully detected and no disturbed regions are wrongly predicted as targeted regions, we consider it a successful attack.

The object detection attack has received success rates ranging from 5.0% to 19.9% on breaking these three versions of SACaptcha. It takes 0.11 to 0.15 seconds to solve a Captcha, on average. With the increase of foreground shapes and styles, the success rate decreases, but it has litttle effect on the average attack speed. In most cases, Faster R-CNN omits some regions in an image or incorrectly detects the background region as the foreground region. It also performs poorly on guessing the shapes of regions, as Figure 8 (b) shows. What should be mentioned is that the real success rate of an object detection attack will be lower than the value we reported, because object detection techniques only return a rough location of each region (a rectangle); they cannot return a pixel-level location, especially if randomly generated irregular shapes are used.

Pixel-level segmentation attack. In essence, SACaptcha highly depends on the pixel-level segmentation problem, so we utilize pixel-level segmentation techniques to evaluate its robustness as well. A fully convolutional network (FCN) [60] was chosen for our pixel-level segmentation network. It was originally designed for semantic segmentation, in which each pixel was labeled with the class of its enclosing object or region. For each Captcha in the training set, we create a mask image in which all background pixels are marked with black and all pixels belonging to the ground-truth style regions are marked with distinguishing colors indicating different shapes. Both Captcha images and masks are sent into the network for training. While evaluating the effectiveness of pixel-level segmentation attack, we input test Captchas into the pre-trained network, which outputs a predicted mask image for each Captcha indicating the predicted result of each pixel. Figure 8 (c) depicts the attack result of the sample Captcha. To make sure that readers can easily understand the attack result, we also mark targeted and disturbed regions using red lines and green lines in Figure 8 (c). The figure shows that FCN correctly detected two foreground regions with a circle shape and one with a rectangle shape, but it incorrectly predicted a heart-shaped foreground region as a circular one. It also wrongly predicted a background region as a foreground region and omitted a pentagram-shaped foreground region as well.

We follow a same evaluation principle as object detection attack. As shown in Table VII, the success rates received by pixel-level segmentation attack range from 4.5% to 43.6%. The average attack speeds vary from 2.16 to 2.21 seconds, which are slower than object detection attack's. Again, the more the foreground shapes and styles were applied, the lower the success rate was received. Therefore, given acceptable usability, it is better to use more foreground shapes and styles in SACaptcha. FCN is effective in predicting the style region pixels, but it always wrongly selects a large number of background pixels as foreground pixels. While FCN works well on guessing standard shapes, e.g., circles and rectangles, it fails to predict special shapes such as hearts and pentagrams.

Further study. To investigate how the size of training set impacts on the success rates, we take the third version of SACaptcha as a representative and enlarge the training sets from 4000 to 10,000 to retrain the deep learning models. With the enlargement of the training set, the success rates received by object detection attack and pixel-level segmentation attack increase to 7.2% and 11.8%, respectively. But from the point of view of attackers, the cost of manually adding labels to images has also increased dramatically. Above attack results we achieved are based on the accurate labels we specifically generated with the SACaptcha production process, e.g., the location information for object detection and the mask image for pixel-level segmentation. Compared with text Captchas and early image-based Captchas based on image classification, it is more difficult to manually add labels to SACaptcha, especially



(a) (b) (c)

Fig. 8. Attack results of the sample SACaptcha: (a) Edge detection, (b) Object detection, (c) Pixel-level segmentation.

TABLE VII
ATTACK RESULTS OF SACAPTCHA

		Version 1	Version 2	Version 3
Object detection attack	Success rate	19.9%	12.8%	5.0%
	Average attack speed (s)	0.12	0.11	0.15
Pixel-level segmentation attack	Success rate	43.6%	17.5%	4.5%
	Average attack speed (s)	2.16	2.21	2.20

using pixel-level segmentation techniques to attack.

Above results indicate that we can further increase the categories of foreground shapes and styles to enhance the robustness of SACaptcha under the premise of ensuring usability. However, in fact, it is hard to strike the right balance between security and usability in Captcha design [4]. We also cannot exhaust all possible attacks here. Thus, we leave the more comprehensive security analysis as an open problem and share it with all research communities.

Obviously, designing a Captcha that is completely resistant to all known attacks is extremely difficult. We proposed a novel method of designing image-based captcha using deep learning techniques, although its security does not seem ideal. Overall, this work is a positive attempt, and our use of deep learning techniques to enhance the security of Captchas is a promising direction.

VI. SUMMARY AND CONCLUSION

We have systematically provided a comprehensive analysis of text Captchas. To evaluate their security, we proposed a simple, effective and fast attack on text Captchas. Using deep learning techniques, we have successfully attacked all Roman character-based text Captchas deployed by the top 50 most popular websites in the world and achieved state-of-the-art results. Our success rates range from 10.1% to 90.0%. The average speed of our attack is much faster than earlier reported attacks. We also used Chinese Captcha as a representative to analyze the security of Captchas using a large character set. It was still vulnerable to our attack, despite the larger solution space. We have successfully broken three Chinese Captchas collected from QQ and Baidu with success rates of 93.0%, 32.2% and 28.6%.

The attack presented in this paper has provided powerful evidence that existing text Captchas are not robust enough, neither for traditional text Captchas based on English letters and Arabic numerals nor for Captchas originating from large-alphabet languages. Our work counters the concept of the segmentation-resistance principle introduced in [7]. Even though various mechanisms are adopted to increase the difficulty of determining where each character is in a Captcha image, and although the rough segmentation method we used cannot precisely segment characters from each other, our attack was effective on all targeted schemes. The segmentation-resistance principle may no longer be appropriate in current Captcha design.

Text Captcha, as the most widely used Captcha scheme, has played an important role in distinguishing humans from computers for a long time. Although previous work has shown that it is not as secure as expected, text Captcha is still widely used, as previous attacks have been slow. However, our attack has created a real-time threat to real-world text Captchas despite the various resistance mechanisms used. It is therefore natural to ask: is the era of text Captcha at an end? Our attack might suggest that the current common use of text Captcha designs is doomed, and yet we hesitate to pronounce a death sentence on text Captcha altogether. It is highly likely the new generation of text Captchas will be designed to be resistant to deep learning attacks. We expect that our work will promote more-effective ways to enhance the security of text Captchas.

We also proposed a novel image-based Captcha named SACaptcha using neural style transfer techniques. Most early image-based Captchas are based on the problem of image classification, whereas SACaptcha relies on problems of semantic information understanding and pixel-level segmentation. This is a positive attempt to improve the security of Captchas by utilizing deep learning techniques.

In this paper, deep learning techniques play two roles: as a character recognition engine to recognize individual characters and as a powerful means to enhance the security of the image-based Captcha we proposed. This work seems to provide an evidence that deep learning is a double-edged sword. It can be either used to attack Captchas or improve the security of Captchas.

We hope our work has indicated a direction for future

Captcha study: existing text Captchas are no longer secure, and other Captcha alternatives are worthy of investigation as substitutes. The question of whether other Captcha alternatives are robust and whether the designs of new Captchas can be simultaneously secure and usable are still open problems and are part of our ongoing work.

REFERENCES

[1] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in CryptologyłEUROCRYPT 2003*. Springer, 2003, pp. 294–311.

[2] L. Von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Communications of the ACM*, vol. 47, no. 2, pp. 56–60, 2004.

[3] E. Bursztein, M. Martin, and J. Mitchell, "Text-based captcha strengths and weaknesses," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 125–138.

[4] J. Yan and A. S. El Ahmad, "Usability of captchas or usability issues in captcha design," in *Proceedings of the 4th symposium on Usable privacy and security*. ACM, 2008, pp. 44–52.

[5] J. Yan and A. S. E. Ahmad, "Breaking visual captchas with naive pattern recognition algorithms," in *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual*, 2007, pp. 279–291.

[6] J. Yan and A. S. El Ahmad, "A low-cost attack on a microsoft captcha," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 543–554.

[7] C. Kumar, L. Kevin, S. Patrice, Y., and C. Mary, "Computers beat humans at single character recognition in reading based human interaction proofs (hips)," in *CEAS 2005 - Second Conference on Email and Anti-Spam, July 21-22, 2005, Stanford University, California, USA*, 2005.

[8] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow captchas," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 2013, pp. 1075–1086.

[9] A. S. El Ahmad, Y. Jeff, and T. Mohamad, "The robustness of google captchas," 2011.

[10] H. Gao, W. Wang, Y. Fan, J. Qi, and X. Liu, "The robustness of "connecting characters togethe" captchas." *J. Inf. Sci. Eng.*, vol. 30, no. 2, pp. 347–369, 2014.

[11] H. Gao, M. Tang, Y. Liu, P. Zhang, and X. Liu, "Research on the security of microsoft's two-layer captcha," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 7, pp. 1671–1685, 2017.

[12] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based captchas." in *WOOT*, 2014.

[13] H. Gao, J. Yan, F. Cao, Z. Zhang, L. Lei, M. Tang, P. Zhang, X. Zhou, X. Wang, and J. Li, "A simple generic attack on text captchas." in *NDSS*, 2016.

[14] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Building segmentation based human-friendly human interaction proofs (hips)," in *Human Interactive Proofs*. Springer, 2005, pp. 1–26.

[15] A. Algwil, D. Ciresan, B. Liu, and J. Yan, "A security analysis of automated chinese turing tests," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 520–532.

[16] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082*, 2013.

[17] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[18] Y. LeCun *et al.*, "Lenet-5, convolutional neural networks," *URL: http://yann. lecun. com/exdb/lenet*, 2015.

[19] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1. Barcelona, Spain, 2011, p. 1237.

[20] V. Turchenko and A. Luczak, "Caffe: Convolutional architecture for fast feature embedding," *Eprint Arxiv*, pp. 675–678, 2014.

[21] D. D'Souza, P. C. Polina, and R. V. Yampolskiy, "Avatar captcha: Telling computers and humans apart via face classification," in *Electro/Information Technology (EIT), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1–6.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[23] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual captcha," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE, 2003, pp. I–134.

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

[25] C. Cruz-Perez, O. Starostenko, F. Uceda-Ponga, V. Alarcon-Aquino, and L. Reyes-Cabrera, "Breaking recaptchas with unpredictable collapse: Heuristic character segmentation and recognition," *Pattern Recognition*, pp. 155–165, 2012.

[26] P. Baecher, N. Büscher, M. Fischlin, and B. Milde, "Breaking recaptcha: a holistic approach via shape recognition," *Future challenges in security and privacy for academia and industry*, pp. 56–67, 2011.

[27] O. Starostenko, C. Cruz-Perez, F. Uceda-Ponga, and V. Alarcon-Aquino, "Breaking text-based captchas with variable word and character orientation," *Pattern Recognition*, vol. 48, no. 4, pp. 1101–1112, 2015.

[28] S. Li, S. Shah, M. Khan, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz, "Breaking e-banking captchas," in *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 2010, pp. 171–180.

[29] C.-P. Karthik and R. A. Recasens, "Breaking microsoft's captcha," Tech. Rep, Tech. Rep., 2015.

[30] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am robot:(deep) learning to break semantic image captchas," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE, 2016, pp. 388–403.

[31] T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood, "Using synthetic data to train neural networks is model-based reasoning," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017, pp. 3514–3521.

[32] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang *et al.*, "A generative vision model that trains with high data efficiency and breaks text-based captchas," *Science*, vol. 358, no. 6368, p. eaag2612, 2017.

[33] J. Elson, J. R. Douceur, J. Howell, and J. Saul, "Asirra:a captcha that exploits interest-aligned manual image categorization," in *ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, Usa, October*, 2007, pp. 366–374.

[34] P. Golle, "Machine learning attacks against the asirra captcha," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 535–542.

[35] M. Chew and J. D. Tygar, "Image recognition captchas," *Lecture Notes in Computer Science*, vol. 3225, pp. 268–279, 2004.

[36] B. Cheung, "Convolutional neural networks applied to human face classification," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 2. IEEE, 2012, pp. 580–583.

[37] G. Goswami, B. M. Powell, M. Vatsa, R. Singh, and A. Noore, "Fr-captcha: Captcha based on recognizing human faces," *PloS one*, vol. 9, no. 4, p. e91708, 2014.

[38] ——, "Facedcaptcha: Face detection based color image captcha," *Future Generation Computer Systems*, vol. 31, no. 1, pp. 59–68, 2014.

[39] H. Gao, L. Lei, X. Zhou, J. Li, and X. Liu, "The robustness of face-based captchas," in *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2248–2255.

[40] Y. Rui and Z. Liu, "Artifacial: Automated reverse turing test using facial features," *Multimedia Systems*, vol. 9, no. 6, pp. 493–502, 2004.

[41] B. B. Zhu, J. Yan, Q. Li, C. Yang, J. Liu, N. Xu, M. Yi, and K. Cai, "Attacks and design of image recognition captchas," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 187–200.

[42] R. Datta, J. Li, and J. Z. Wang, "Imagination: a robust image-based captcha generation system," in *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM, 2005, pp. 331–334.

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

[44] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. van Oorschot, and W.-B. Chen, "A three-way investigation of a game-captcha: automated attacks, relay attacks and usability," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*. ACM, 2014, pp. 195–206.

[45] M. Shirali-Shahreza and S. Shirali-Shahreza, "Drawing captcha," in *Information Technology Interfaces, 2006. 28th International Conference on*. IEEE, 2006, pp. 475–480.

[46] R. Lin, S.-Y. Huang, G. B. Bell, and Y.-K. Lee, "A new captcha interface design for mobile devices," in *Proceedings of the Twelfth Australasian User Interface Conference-Volume 117*. Australian Computer Society, Inc., 2011, pp. 3–8.

[47] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo, "No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2640–2653, 2017.

[48] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[49] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[50] S. Gu and L. Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[51] NUCAPTCHA, "Nucaptcha & traditional captcha," http://nudatasecurity.com, Accessed on May, 2016.

[52] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. C. van Oorschot, "Security and usability challenges of moving-object captchas: Decoding codewords in motion." in *USENIX security symposium*, 2012, pp. 49–64.

[53] E. Bursztein, "How we broke the nucaptcha video scheme and what we proposed to fix it," *See http://elie.im/blog/security/how-we-broke-the-nucaptcha\-video-scheme-and-what-we-propose-to-fix-it/, Accessed March*, 2016.

[54] J. Tam, J. Simsa, S. Hyde, and L. V. Ahn, "Breaking audio captchas," in *Advances in Neural Information Processing Systems*, 2009, pp. 1625–1632.

[55] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 262–270.

[56] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[57] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," *arXiv preprint arXiv:1603.08155*, 2016.

[58] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving captchas? a large scale evaluation," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 399–413.

[59] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[60] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

Mengyun Tang is a master degree candidate in computer science at Xidian University. Her current research interest is Captcha.

Haichang Gao is a professor of the Institute of Software Engineering at Xidian University and a member of the IEEE. He has published more than 30 papers. Now he is in charge of a project of the National Natural Science Foundation of China. His current research interests include Captcha, computer security and machine learning.

Yang Zhang is a master degree candidate in computer science at Xidian University. Her current research interest is Captcha.

Yi Liu is a master degree candidate in computer science at Xidian University. His current research interest is Captcha.

Ping Zhang is a master degree candidate in computer science at Xidian University. His current research interest is Captcha.

Ping Wang is a doctor degree candidate in software engineering at Xidian University. Her current research interest is Captcha.