

# When Deep Learning Meets Captcha

**Abstract**—CAPTCHA is a standard secure mechanism to automated tell computers and humans, and is applied to prevent against malicious bot programs. Current text-based Captcha mechanism combines many anti-segmentation and anti-recognition features such as sophisticated noisy background, character rotation, distortion and overlapping. Such mechanism is widely deployed by Google, Microsoft, Baidu, Alibaba. Many attacks on text-based Captcha have been proposed. However, most of these fine prior works focus on a unique Captcha mechanism, making them have a limited applicability. In this paper, we demonstrate a novel generic attack on current text-based Captcha. We employ the Generative Adversarial Networks (GANs) to translate the original Captcha to the regular one. Using the translated regular Captcha to train a Convolutional Neural Network (CNN) model, our approach is able to accurately identify the Captcha. We thoroughly evaluated our approach using real-world Captchas, and achieve a success rate of over XX with an average speed of 76ms on an ordinary server (with a 3.2-GHz Inter Xeon CPU with 100-GB RAM and a TITAN Xp GPU). We discovered that, the Captcha with complex noisy background do not defend against segmentation under our attack. This is demonstrated by the fact that we are able to remove the complex background from the Captcha image. Since our attack can identify a widely text-based Captchas, this paper calls the community to revisit the risks of using text-based Captcha to defend against malicious bots.

**Keywords**—*Captcha, Security, Generic Attack, Text-based*

## I. INTRODUCTION

A CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Human Apart) is a automated test that humans can pass but computer programs cannot [23]. It provides a effective approach for automatically distinguishing humans from computer systems, and therefore is used to defend against automatic spam, registration or malicious bots [20, 22].

The most widely deployed CAPTCHA is the so-called text-based scheme [28], which mainly consists of distorted English letters and Arabic numerals. The popularity of this scheme is due to its obvious advantages [4, 5]: (1) most people around the world can recognize English letters and Arabic numerals; (2) the space of the text-based Captcha are huge so that the brute-force attack can be defeated. Given its pervasive usage, a security breach of the text-based Captcha could lead to serious consequences.

The robustness of text-based Captchas is the significant concern in the research communities. Over the past decade, researchers have uncovered a number of ways to recognize text-based Captchas. Many fine prior researches just focus on attacking an unique Captcha scheme [7, 8, 17, 27]. This limits their applicability. Recently the generic attacks have been proposed by Gao *et al.* [9] and Bursztein *et al.* [1, 2]. They claim that they can break a wide range of text-based Captchas using a generic method. However, these defeated Captchas possess relative simple noisy background or uniform style. Although the security of text-based Captchas have been proven frail, many companies such as Google, Microsoft and Baidu still use such scheme as current text-based Captcha scheme have more complex background or distorted characters. This makes previous attacks invalid. Recent studies [3, 21] also demonstrate that text-based Captcha is still a secure mechanism.

The key factor of previous attacks on text-based Captcha lies in the difficulty of segmenting characters [5]. Therefore, the basic design principle of text-based Captchas should be anti-segmentation. To this end, current text-based Captchas with more complex noisy background and distorted characters have been proposed and deployed by many companies. In order to increase the difficulty of finding where each character is, such Captcha shorten the distance between characters. All attacks proposed above were invalid due to cannot successfully segment the characters. It is therefore driving us wondering: is current Captcha scheme as secure as it is expected? This fundamental question precipitated our study.

In this paper, we present a novel generic attack on current text-based Captchas using deep learning technology. Our attack employs a variant of the generative adversarial network (GAN) [11] to transform the distorted captchas to the regular ones. The former serval layers of the GAN is used to remove the complicated noisy background. The middle layers aim to enlarge the distance between adjacent characters and output the distorted captchas with larger inter-character distance which are translated to regular ones by the last layers. At last, the transformed regular captchas are recognized by a Convolutional Neural Network (CNN).

We thoroughly evaluate our approach using real-world captchas collected from some websites. We show that our approach is effective in recognize current text-based captchas and as a results, we can defeat almost all current text-based captchas with a success rate range from XX% to XX%. We demonstrate that, the sophisticated noisy background cannot offer stronger protection in term of anti-segmentation under our attack. Our finding suggests that text-based captchas are insecure under the age of artificial intelligence.

**Contributions** This paper makes the following specific contributions:

In order to design a more security Captcha scheme, cur-

rently Captchas deployed by many companies

## II. BACKGROUND

### A. Text-based Captchas

CAPTCHA is also called inverse Turing test [18], and aims to determine whether or not the user is human. The most popularity of Captchas is text-based Captchas. Initially, they were composed of deformed English letters and Arabic numerals which human can recognize while computers cannot. The usage of English letters and Arabic numerals makes text-based Captchas being welcomed world-wide [4, 5]. Due to such advantages, text-based Captchas have been widely used in various of Internet security access tasks [23]. These tasks include defending against malicious scraping web contents, preventing automatic registration of free accounts for spam posting or forums, or mitigating the impact of DDoS attacks.

1) *Captcha Security Features*: To summary out a representative security features, we collected a wide range of Captchas that had ever been or being used by the top 20 most popular web sites which are listed by Alexa<sup>1</sup>. Figure 1 gives some representative examples of our collected Captchas<sup>2</sup>. It obviously shows that current text-based Captchas have more complex security features than previous Captchas. These features can be classified into two categories: anti-segmentation and anti-recognition features. The anti-segmentation feature aims to increase the difficulty of character segmentation. In order to implement such feature, the Captcha is needed to be confused on the whole with sophisticated background, extra connecting lines and overlapping characters. Thus, the anti-segmentation features can be regarded as the global security features as it makes the Captcha complex on the whole. In contrary, the anti-recognition features belong to local features. It can defense against character recognition by changing the font style of Captchas. We list the two fratures as follows:

**Anti-segmentation features** 1. **Sophisticated Noisy Background** Try to make the background have little difference with the text for confusing the solver. 2. **Connection Lines** Add extra lines on the text to prevent solver from automatic finding the single character. 3. **Overlapping** Decreasing the space between the adjacent characters to make them collapsed.

**Anti-recognition features** 1. **Character Set** Which charset the text-based Captchas scheme uses. Some schemes only include English letters while some schemes consist of both English letters and Arabic numerals. 2. **Font Style** Using multiple font styles such as solid font, hollow font or font compromise of dots. 3. **Font Size** Using random font size when generating Captcha. 4. **Font Color** Using variable font colors. 5. **Distortion** Distorting characters using attractor fields. 6. **Rotating** Rotating characters with random angles. 7. **Waving** Distorting the characters in a wave fashion.

2) *Previous Captcha Scheme*: According to the [9], the previous text-based Captchas can be classified into three categories: character isolated Captcha (Figure 1 (a)), hollow character Captcha (Figure 1 (c)) and Connecting Characters Together (CCT) Captcha (Figure 1 (b) and (d)) based on

font style and the space between adjacent characters. Obviously, most of previous text-based Captchas use a single anti-segmentation or anti-recognition features. For examples, Figure 1 (a) only uses one single anti-recognition feature (slight rotating) and Figure 1 (b) uses one anti-segmentation feature (character overlapping) and one anti-recognition feature (character waving). Of course, some Captcha schemes use a simple combination of the above security features such as Figure 1 (f) uses rotating, background and connection lines. Although using multiple anti-segmentation and anti-recognition features, the text have the same font style or obvious difference comparing to the background.

Thus, the robustness of text-based Captchas is an active topic in the research communities. Researchers have comprehensively evaluated the security of text-based Captchas [1, 2, 9]. They demonstrate that text-based Captchas are vulnerable under their attacks as they are able to effectively segment each character. Although the security of text-based Captcha scheme are suspicious by research communities, such scheme is still primary authentication mechanism and are widely deployed most mainstream websites. This wide-spread usage, we think, is due to the following reasons: (1) the written text on Captcha image may be presented in many styles such as distorted, rotated, hollow, or overlapping characters, complex noisy background or noise interference. Each of these styles can be designed more complex than previous. For examples, the value of character's rotating angle can be set more large larger than previous, and the background can be became more complex. (2) The combination of these styles is able to increase the security strength of Captchas so that it can defeat against current attacks. Current Captchas (Figure 1 (2)) combine many anti-segmentation and anti-recognition features than previous Captchas (See Figure 1 (1)). This increases the security of text-based Captchas. Recent studies suggests that text-based Captchas are still a security mechanism if they are properly designed [3, 21]. In the following, we will introduce the current Captcha scheme.

3) *Current Captcha Scheme*: Current Captcha scheme combines many anti-segmentation and anti-recognition features. This makes current text-based Captchas more complex than previous Captcha schemes. Figure 1 (2) shows some representative current Captcha schemes. Obviously, it presents that each current Captcha scheme is compromised of many security features such as different font style, character overlapping, rotating and distorting, connecting lines and so on. Take Figure 1 (k) for an example, this Captcha using two anti-segmentation features (complex background and connecting lines) and five anti-recognition features (Character Set: both English letters and Arabic numerals, Font Style: both solid font and hollow font, Font Size: different font size, Font Color: different font color, Rotating: rotating character with different angles.). Likewise, other Captchas presented in Figure 1 also uses many anti-segmentation and anti-recognition features.

Previous attacking methods on Captchas cannot directly crack current complex text-based Captchas. The pixel counting approach proposed by Yan *et al.* [26] will lose their efficiency in current Captchas. This is because the character sizes of current Captchas are different, which leads to the number of pixels is not fixed. Color Filling Segmentation (CFS) method proposed by Yan and used by Gao [8, 27] are also

<sup>1</sup><http://www.alexa.com/topsites>

<sup>2</sup>Some previous used Captchas come from the related works [2, 8, 9].

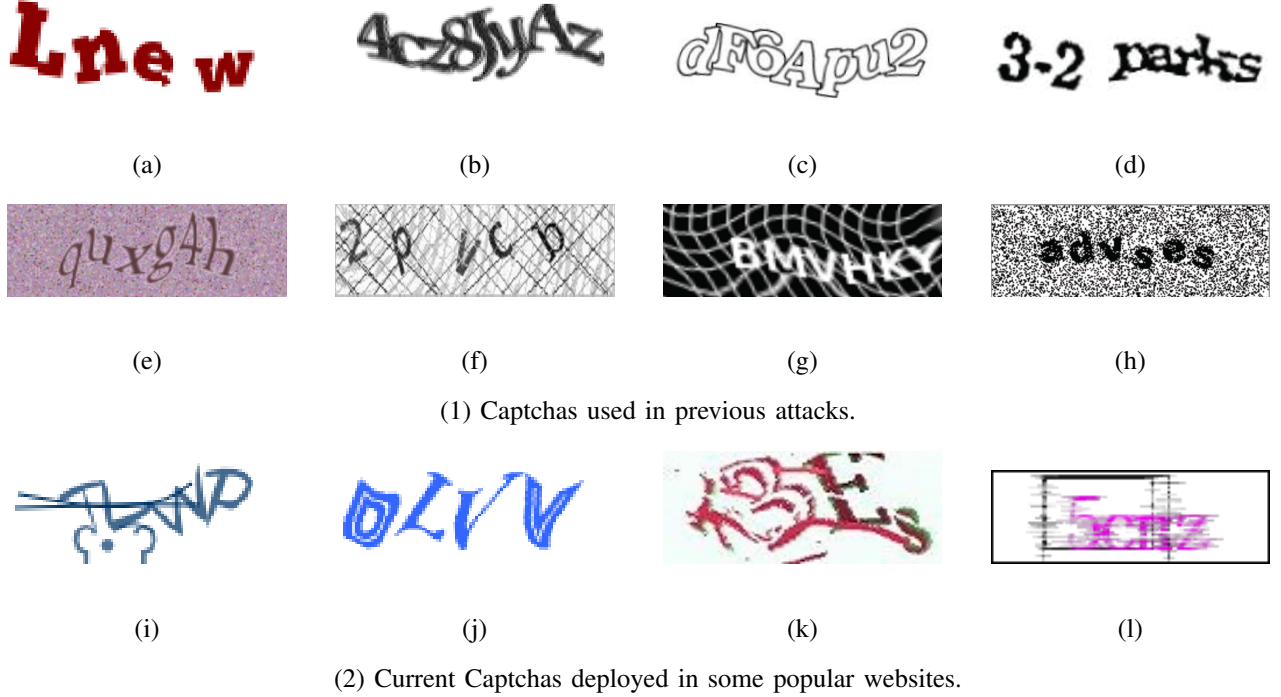


Figure 1. This shows the Captchas used in prior works and current Captchas deployed in some famous websites.

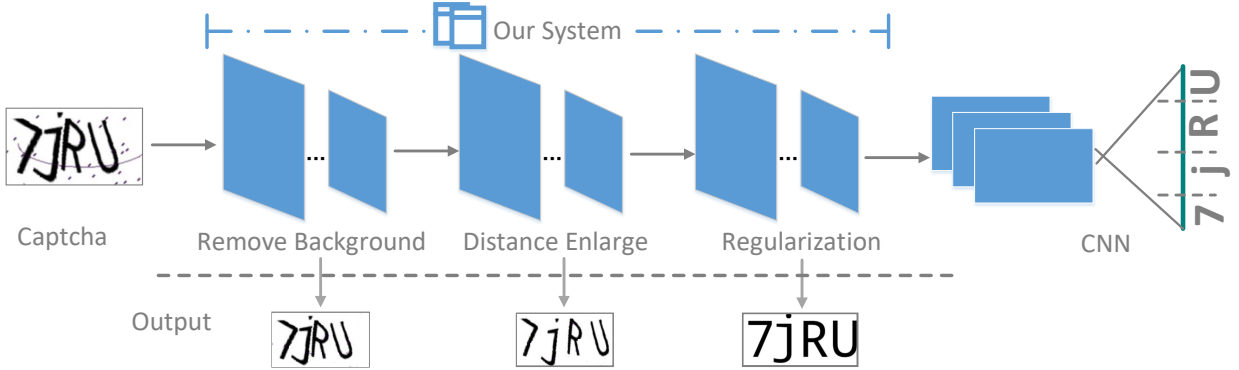


Figure 2. The overview of our attack.

cast into the shade as sometimes the characters on current Captcha are overlapped, and CFS method cannot segment the connected character. The powerful generic attacks which claim to crack a wide range of text-based Captchas have been proposed [1, 9]. The attack described in [1] is a state-of-the-art way to segment the characters. However, this approach cannot effectively segment the characters when the space between adjacent characters is -3 pixel which is more than the space (generally about -4 pixel) between the adjacent characters of current Captcha. The work in [9] exploits Gabor filters to extract character components, and then rank the adjacent character components to recombine the individual characters. The major issue of this method is the high error ranking rate whiling cracking hollow characters. This is because the hollow characters will be extracted many possible tiny components, which results in many candidate combinations, increasing the error rate. Thus, comparing to previous Captchas, current Captcha scheme deployed by many popular web sites are more

robustness, and can defeat against previous attacks.

In terms of the above complex Captcha scheme, this paper present a novel generic attack method using deep learning technology. Unlike previous attacks, our method is able to effectively clean up the complex background and connecting lines, and then the space between adjacent overlapping characters can be expanded as much as possible. At last, we explore to translate the distorted character to a regular one for the purpose of improve the success rate using CNN recognition engine. Our work aims to call the community to revisit the security of text-based Captchas under the age of Artificial Intelligence.

### B. Generative Adversarial Networks

Generative Adversarial Networks (GANs) is a special type of Artificial Neural Network (ANN), and it was first proposed by Goodfellow in 2014 [10]. GANs consists of two neural



Table I. MAPPINGS FROM SECURITY FEATURES TO ITS CORRESPONDING NUMBERS

Security Features	Anti-segmentation Features			Anti-recognition Features						
	Complex Background	Connection Lines	Overlapping	Character Set	Font Style	Font Size	Font Color	Distortion	Rotating	Waving
Feature Number	①	②	③	④	⑤	⑥	⑦	⑧	⑨	

networks competing with each other: one is the generative model  $G$  that generates the data which similar to the true data; the other is the discriminative model  $D$  which validate whether the inputting data comes from the training data or  $G$ . By repeating this competing procedure iteratively, this game will be terminated if the discriminative model  $D$  cannot identify the inputting data comes from training data or the generative model  $G$ .

The adversarial trait of GANs makes it has been widely used in video predicting [24], image processing [11, 30], natural language processing [13, 29], code security [15, 25]. Among these fine studies, Phillip *et al.* [11] proposed an image-to-image translation method named *Pix2Pix* [6] using the conditional GANs<sup>3</sup> (cGANs) [16]. In their work, they can translate an image from one style to another. Differ from typical GANs, cGANs consists of conditional generator and discriminator. Both the conditional generator and discriminator take the observed image as an input image so that cGANs can converge to a stable state.

This work inspired us that can it translate the distorted Captcha image with complex background and overlapping characters to the regular one with non-overlapping characters and without background? To do this, we design an effectively segment method for Captchas using a variant of *Pix2Pix* [6]. Our method is able to clean up the complex background and connecting lines on Captcha image, expand the space between the adjacent characters and translate the distorted character to the regular one. The preliminary experiments prove this method is valid. To our best knowledge, this is the first work to attack Captchas using GANs.

### III. CAPTCHA GENERATOR

A key success factor for deep learning system is large amount of data to train. Intuitively, the simple yet direct approach in data collecting is to mine the Captchas from target websites and then paid to mark the labels to them artificially. Obviously, it is a time- or financial-consuming work with a high error rate during marking labels. Further, some type of Captchas are difficult to mine as the corresponding websites limit the accessing frequency. In addition, there is never such a public Captcha generator which can produce enough different styles of Captchas that match the requirement of the training process. Thus, the first concern of our work is to develop a flexible, well-calibrated training data generator.

To achieve it, we design an generation model to imitate Captchas production process, and automatically generate different styles of Captchas that are similar to those deployed in real-world websites. Given a unique text-based Captcha scheme  $x$ , we first manually analyze the number of the characters and their corresponding security features,  $S_{1:N}$ , with

<sup>3</sup>Conditional GANs learn a mapping from observed image with random noise to output image while GANs learn the mapping from only random noise to output image.

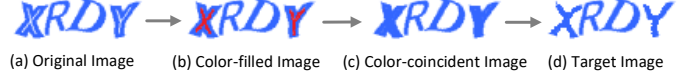


Figure 3. The procedure of Captcha preprocessing.

$N$  parameters such as font style, size and color, rotating, distortion and waving *et al.* described in Section II-A1. Here we ranked the from No.1 to 10 shown as Table I. We specifics this analysis results as  $\{M, S_{1:N}\}$ , where  $M$  and  $N$  respectively are the number of characters and its styles. Then given the content of Captcha, our generation model is able to automatically produce Captchas image  $y$ , which accords with the above analyzed style. We define our generation model as follows:

$$y | x \sim G_s(x), x = \{M, N, L_{1:M}, S_{1:N}\} \quad (1)$$

Where  $G_s(x)$  is the generation model parameterized by unique style  $s$ , that can generate the Captcha image  $y$  based on the unique Captcha scheme  $x$ .  $M$  is the number of characters on the Captcha image and  $L_{1:M}$  represents the content of the characters. For each character, our generation model individually generates corresponding style defined by  $S_{1:N}$ , because each character on some Captchas may have different style.

**Example:** We use the Captcha image depicted in Figure 2 as an example to describe our Captcha generator. This Captcha scheme consists of both English letters and Arabic numerals and the number of characters are fixed (4 characters). The content of the Captcha is  $\{7, j, R, U\}$ . It uses 2 anti-segmentation features (Complex Background and Connection Lines) and 4 anti-recognition features (Character Set, Font size, Rotating and Distortion). So the collection of security features number is  $\{①, ②, ③, ④, ⑤, ⑥, ⑦\}$ . Considering these parameters, the variance  $x$  is  $\{4, 6, \{7, j, R, U\}, \{①, ②, ③, ④, ⑤, ⑥, ⑦\}\}$ . For each character in the collection  $\{4, 6, \{7, j, R, U\}\}$ , our generator product the subgraph correspond to its security features. At last, the generator aggregates the subgraph to produce a Captcha image.

### IV. OVERVIEW

This section gives an overview of our attacking system which exploits Deep Learning techniques to recognize current Captchas. The system takes in a distorted text-based Captcha with complex noisy background. It automatically generated the regular Captcha corresponding to the distorted one. At last, the regular Captcha is cracked by CNN recognition engine. Figure 2 depicts the three steps of our attack:

#### Step 1. Captcha Preprocessing

The attack begins from inputting the distorted Captchas with complex background and overlapping characters. Given the characters on Captcha have different styles (see Figure 3 (a), it includes both hollow and solid characters), it is necessary

---

**Algorithm 1** Unifying the character font style

---

**Input:***CI*: Captcha Image*numChar*: The number of characters on Captchas**Output:***UC*: Captcha with the same font style

```
1: grayCI  $\leftarrow$  getBinaryImage(CI)
2: positions[]  $\leftarrow$  getHollowPositions(grayCI)
3: LEN  $\leftarrow$  getPositionLen(positions[], numChar)
4: meanThick  $\leftarrow$  getMeanCharThick(CI, positions[])
5: for i = 1 : LEN do
6:   cFI  $\leftarrow$  FillRedColor(CI, positions[i])
7:   cCI  $\leftarrow$  ChangeFillColor(cFI, positions[i])
8:   charThick  $\leftarrow$  getCharThick(cCI, positions[i])
9:   if charThick > meanThick then
10:    UC = cropChar(cCI, positions[i], meanThick)
11:   end if
12: end for
```

---

to uniform the style of these characters for further processing. We have shown that the system can automatically translate the hollow character to the solid one (Figure 3).

**Step 2. Generate Regular Captcha**

Once the style of character on Captcha image is uniformed, a deep learning algorithm will be applied to generate the regular Captcha shown as Figure 2. We archive this through hierarchical approaches described as the following three steps:

① **Remove Noisy Background:** To generate the regular Captcha, the first step is to clean the complex background, and produce the Captcha with white background. To do so, a deep learning algorithm will be applied to remove the complex noisy background stay on Captcha. For each Captcha scheme, this algorithm needs to train the model for cleaning our the background.

② **Expand Space Between Adjacent Characters:** This step aims to increase the distance between two characters on Captcha. We use the same algorithm as step 2 to enlarge the inter-character distance and generate a new Captcha. Keep in mind that the Captcha generated at this stage is distorted.

③ **Regularization:** In this step, our system is able to automatically translate the distorted Captcha to a regular one.

**Step 3. Recognize Captcha**

In this final step, we use a radical CNN model as the recognition engine to identify the text of the regular Captcha translated from last step.

## V. IMPLEMENTATION DETAILS

### A. Captcha Preprocessing

Given some current text-based Captchas mainly consist of both solid character and hollow character (see Figure 1 (j) and (k)), the first step of our attack is to unify the different font styles to a fixed style. Here we aims to fill the hollow character with solid core due to the following two reasons: (1) the hollow character can be easily transformed to the solid one but not the opposite. (2) the solid character can be extracted more stable

features than hollow character at the following step according to our preliminary experiments.

To do so, we first convert the colorful image to black-and-white using the standard threshold selection method proposed by Otsu [19]<sup>4</sup>. Then the Color Filling Segmentation (CFS) [27] is used to fill the hollow character with the red color (see Figure 3 (b)). Next, the color-filled image should be convert to color-coincident image by changing the red filled area to the original character color (see Figure 3 (c)). At last, the thick filled characters on the color-coincident image need to be cropped as the original character (see Figure 3 (d)).

Our method for unifying the character font style is described in Algorithm 1. The input to the algorithm is the original Captcha image with different font styles and the number of characters on this Captcha, and the output of the algorithm is the Captcha with solid characters. To locate the hollow characters, we first convert the colorful original image to binarized image (line 1). According to the binarized image, we can get the positions of the each hollow characters on the colorful Captcha image (line 2) because the size of colorful image is the same as the binarized image. For each hollow character, we use CFS method described above to fill it with red color (line 6) and then replace the red filled color with the character color to get the color-coincident image (line 7). Finally, we crop the filled character on the color-coincident image to unify the font style of characters (line 10).

### B. Generate Regular Captcha

After unifying the character font style, we need to generate regular Captchas for better recognizing. We achieve this by employing an image-to-image translation algorithm called Pix2Pix [6]. This algorithm automatically translate the image from the original style to the target style. In our case, the images to be translated are the Captchas with complex noisy background, overlapping and distorted characters (original style). These are supplied to the algorithm by a Captcha generator developed using a simple script (Section III). The algorithm tries to generate regular Captcha with appropriate character spacing and clean background (target style).

In order to generate regular Captcha, we propose the hierarchical methods that employ a variant approach of *Pix2Pix* to complete the progressive tasks. These hierarchical methods share the same methodology of the variant algorithm. The main differences of these hierarchical approaches are the input and output images. The inputs of the first steps are the images with complex background, overlapping and distorted characters. These inputting images are eliminated the complex background by the variant approach which outputs the Captcha images with white background, overlapping and distorted characters. According to the outputs, the second step is to expand the space between the adjacent overlapping characters using the same variant method, and produces the Captcha images only with distorted characters. Finally, the distorted characters are translated to the regular ones using the variant method.

**Hierarchical Methods.** The hierarchical methods are comprised of three sequenced models, and they can respectively

---

<sup>4</sup>Note that we only use the binarized image to locate the position of the hollow part of the character other than furthering processing.

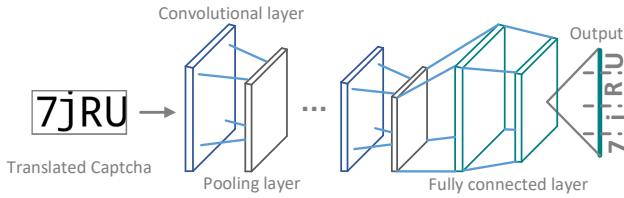


Figure 4. CNN recognition engine. The input of the recognition engine is the regular Captcha image, and it outputs the text of the Captcha image.

achieve the tasks of removing complex background, expanding space between adjacent overlapping characters and translating the distorted Captcha to a regular one. The sequenced models share the same translation model. The key part of the translation model is a variant of *Pix2Pix*, which consists of an image generator and discriminator. They are competing with each other until reach Nash equilibrium when training processing (Section II-B).

In our case of removing background, the inputting data are the image pairs including the Captcha images with complex background and the images with white background. The goal is to train a generative model that can translate the Captcha image with complex background to the images with white background. During training, the image generator produces the fake image that similar to the image with white background by randomly add the noisy points to it. The fake image and the image with complex background compose an image pair. For the composed image pair, the discriminator learns to classify between the real and fake pairs. This competing process will be terminated until the discriminator cannot classify the image pairs. Unlike the *Pix2Pix*, we use the L2 distance to figure out the loss of the generator as L2 loss can capturing the overall structure of the Captcha image, which contribute to removing the background.

$$\mathcal{L}_{L2}(Gen) = \mathbb{E}_{x,y \in C_O, z \in N_O} \|y - Gen(x, z)\|_2 \quad (2)$$

Where  $x$  is the Captcha image with white background,  $y$  is the image with complex background and  $z$  is the fake image with the noisy points.

### C. Identify Captcha

In this step, we use a rudimentary CNN framework, *LeNet-5* [12], as our recognition engine, to identify the text of the translated Captcha image. The initial *LeNet-5* was comprised of three convolutional layers, two pooling layers and followed by two fully connected layers. The convolutional layer extracts the features using a number of filters that are trained during training process. The pooling layer aggregates the features extracted from the convolutional layer for extracting more representative features meanwhile reducing the amount of calculation. The fully connected layer classify the extracted features into target categories. The appropriate number of network layers determines the quality of the extracted features as proper number of layers will extract more representative features.

Unlike the *LeNet-5*, the goal of our recognition engine is to recognize the text of Captchas on the whole, which is more difficult than recognition a single character done by *LeNet-5*.

This is because recognizing more characters need to extract more complex and abstract features. To do so, we redesign the *LeNet-5* and adding another two convolution layers and another three pooling layers. Figure 4 depicts the framework of our recognition engine. Generally, it consists of five convolution layers, five pooling layers and followed by two fully connected layers. Each convolution layer is followed by a pooling layer.

To extract more representative features, each convolutional layer uses a convolution filter of  $3 \times 3$  and each pooling layer employs the max-pooling value. Other parameters are the same as the *LeNet-5*.

## VI. EXPERIMENTAL SETUP

### A. Data Preparation and Collection

The Captchas used in our evaluation are made up of both training data and testing data, and they come from two different sources. The training Captchas are synthesized by our Captcha generator (Section III) and the testing Captchas are collected from corresponding websites using a traditional mining method written through a python script.

**Target Websites** We select the target websites based on the following reasons: (1) The Captchas of the websites combines at least two anti-segmentation and three anti-recognition features; (2) the websites must have a large number of active users. According to the two factors, we target ? famous websites for preparing our data: Baidu, XX, XX.

**Synthesize Training Captchas.** To ensure the synthesized Captcha as likely as the true one, we first manually analyze the traits of the true Captchas and aggregates these traits to a feature tuple as described in Section III. According to the feature tuple, a large number of training Captchas can be synthesized by our generator. In our experiment, we totally synthesized ? kinds of Captchas come from the target websites. For each Captcha scheme, we mined 20000 Captchas to used for training process.

**Mine Testing Captchas.** The testing Captchas used in our evaluation are automatically mined from the target websites using a python script. For each target website, we apply 2000 Captchas to test the efficiency of our attacking system. We recruited nine participators from our institution to manually mark the labels of the mined Captchas. To decrease the marking error rate, the nine participator are divided into three groups and each group has three people. We choose these Captcha which two participators mark right at the same time as the testing data.

**Implementation.** Our prototyped attacking system built upon a variant of *Pix2Pix* framework [6] in Tensorflow. The developed software ran on an ordinary server with a 3.2-GHz Inter Xeon CPU with 100-GB RAM and a TITAN Xp GPU. The operating system is Ubuntu 16.04.






### B. Captchas Show

## VII. EXPERIMENTAL RESULTS

In this section, we first present the overall success rate for breaking the Captchas collected from the target websites. Our results shows that the overall success rate range from XX to XX. We then analyze how the success rate is affected by



Table II. TARGET TEXT-BASED CAPTCHA SCHEMES USED IN OUR EXPERIMENT

Scheme	Website	Sample	Security Features	
			Anti-segmentation Features	Anti-recognition Features
Baidu, Tecent	baidu.com		Connecting Lines, Overlapping, Only Enligh letters used	Both hollow and solid characters, varied font size, color, rotating, disortion and Waving used
Alipay	alipay.com		Overlapping characters used	Both English letters and Arabic numerals, rotating and distortion used
NetEase	163.com		Complex background, connecting lines	Both hallow and solid characters, varied rotating angles used
VIPSHOP	vip.com		Complex background, Overlapping characters	Both English letters and Arabic numerals, varied font colors used
TOUP	tuniu.com		Overlapping characters used	Both English letters and Arabic numerals, waving characters used

the number of the training Captchas. Finally, we demonstrate the risks of our attacking approach by simulating a real world attacking process before evaluating the robustness of the secure features using variant types of text-based Captchas.

#### A. Overall Success Rate

### VIII. RELATED WORK

The Automated Turing Tests were first proposed by Naor [18], but they did not provide a formal definition. Lillibridge *et al.* [14] developed the first practical Automated Turing Test to prevent bots from automatically registering web pages. This system was effective for a while and then was defeated by common Optical Character Recognition (OCR) technology.

Text-based Captchas based on English letters and Arabic numerals, is still the most widely deployed scheme. Many research communities focus on developing attack approaches for existing text-based Captchas, and then explore guidelines for better designs.

### ACKNOWLEDGEMENTS

We would like to thank all participants who help for completing the experiments. Thank all volunteers for their time and insights as well as the anonymous reviewer for their critical and constructive comments. This work was supported by NSFC (Grant No. 61672427) and the UK Engineering and Physical Sciences Research Council (Grants No. EP/M01567X/1(SANDeRs) and EP/M015793/1(DIVIDEND)).

### REFERENCES

- [1] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: generic solving of text-based captchas," in *USENIX conference on Offensive Technologies*, 2014.
- [2] E. Bursztein, M. Martin, and J. Mitchell, "Text-based captcha strengths and weaknesses," in *ACM Conference on Computer and Communications Security*, 2011, pp. 125–138.
- [3] E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, J. C. Mitchell, and J. Dan, "Easy does it: more usable captchas," in *ACM Conference on Human Factors in Computing Systems*, 2014, pp. 2637–2646.
- [4] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Building segmentation based human-friendly human interaction proofs (hips)," vol. 3517, pp. 1–26, 2005.
- [5] —, "Computers beat humans at single character recognition in reading based human interaction proofs (hips)," in *Conference on Email & Anti-Spam*, 2005.
- [6] P. I. *et al.*, "Pix2Pix: Image-to-image translation with conditional adversarial networks," <https://github.com/phillipi/pix2pix>.
- [7] H. Gao, M. Tang, Y. Liu, P. Zhang, and X. Liu, "Research on the security of microsofts two-layer captcha," *IEEE Transactions on Information Forensics & Security*, vol. 12, no. 7, pp. 1671–1685, 2017.
- [8] H. Gao, W. Wei, X. Wang, X. Liu, and J. Yan, "The robustness of hollow captchas," in *ACM Sigsac Conference on Computer & Communications Security*, 2013, pp. 1075–1086.
- [9] 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 *Network and Distributed System Security Symposium*, 2016.
- [10] I. J. Goodfellow, J. Pougetabadie, M. Mirza, B. Xu, D. Wardefarley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.
- [11] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arxiv*, 2016.
- [12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [13] J. Li, W. Monroe, T. Shi, S. Jean, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," 2017.
- [14] M. D. Lillibridge, M. Abadi, K. Bharat, and A. Z. Broder, "Method for selectively restricting access to computer systems," 2001.

- [15] Y. Liu, X. Chen, C. Liu, and D. Song, “Delving into transferable adversarial examples and black-box attacks,” 2016.
- [16] M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *Computer Science*, pp. 2672–2680, 2014.
- [17] M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. V. Oorschot, and W. B. Chen, “A three-way investigation of a game-captcha: automated attacks, relay attacks and usability,” in *ACM Symposium on Information, Computer and Communications Security*, 2014, pp. 195–206.
- [18] M. Naor, “Verification of a human in the loop or identification via the turing test,” 1996.
- [19] N. Ostu, O. Nobuyuki, and N. Otsu, “A threshold selection method from gray- level histogram iee transactions on systems,” *IEEE Transactions on Systems Man & Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.
- [20] J. Tam, J. Simsa, S. Hyde, and L. V. Ahn, “Breaking audio captchas,” in *Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December, 2008*, pp. 1625–1632.
- [21] K. Thomas, D. McCoy, A. Kolcz, A. Kolcz, and V. Paxson, “Trafficking fraudulent accounts: the role of the underground market in twitter spam and abuse,” in *Usenix Conference on Security*, 2013, pp. 195–210.
- [22] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, *CAPTCHA: Using Hard AI Problems for Security*. Springer Berlin Heidelberg, 2003.
- [23] 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.
- [24] J. Walker, K. Marino, A. Gupta, and M. Hebert, “The pose knows: Video forecasting by generating pose futures,” 2017.
- [25] W. Xu, Y. Qi, and D. Evans, “Automatically evading classifiers: A case study on pdf malware classifiers,” in *Network and Distributed System Security Symposium*, 2016.
- [26] 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.
- [27] —, “A low-cost attack on a microsoft captcha,” in *ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, Usa, October, 2008*, pp. 543–554.
- [28] —, “Usability of captchas or usability issues in captcha design,” in *Symposium on Usable Privacy and Security, SOUPS 2008, Pittsburgh, Pennsylvania, Usa, July, 2008*, pp. 44–52.
- [29] L. Yu, W. Zhang, J. Wang, and Y. Yu, “Seqgan: Sequence generative adversarial nets with policy gradient,” 2016.
- [30] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks,” *arXiv preprint arXiv:1703.10593*, 2017.