

# Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach

Guixin Ye  
Northwest University, China  
gxye@stumail.nwu.edu.cn

Zhanyong Tang\*  
Northwest University, China  
zytang@nwu.edu.cn

Dingyi Fang  
Northwest University, China  
dyf@nwu.edu.cn

Zhanxing Zhu  
Peking University, China  
zhanxing.zhu@pku.edu.cn

Yansong Feng  
Peking University, China  
fengyansong@pku.edu.cn

Pengfei Xu  
Northwest University, China  
pfxu@nwu.edu.cn

Xiaojiang Chen  
Northwest University, China  
xjchen@nwu.edu.cn

Zheng Wang\*  
Lancaster University, United Kingdom  
z.wang@lancaster.ac.uk

## ABSTRACT

Despite several attacks have been proposed, text-based CAPTCHAs<sup>1</sup> are still being widely used as a security mechanism. One of the reasons for the pervasive use of text captchas is that many of the prior attacks are scheme-specific and require a labor-intensive and time-consuming process to construct. This means that a change in the captcha security features like a noisier background can simply invalid an earlier attack. This paper presents a generic, yet effective text captcha solver based on the generative adversarial network. Unlike prior machine-learning-based approaches that need a large volume of manually-labeled real captchas to learn an effective solver, our approach requires significantly fewer real captchas but yields much better performance. This is achieved by first learning a captcha synthesizer to automatically generate synthetic captchas to learn a base solver, and then fine-tuning the base solver on a small set of real captchas using transfer learning. We evaluate our approach by applying it to 33 captcha schemes, including 11 schemes that are currently being used by 32 of the top-50 popular websites including Microsoft, Wikipedia, eBay and Google. Our approach is the most capable attack on text captchas seen to date. It outperforms four state-of-the-art text-captcha solvers by not only delivering a significantly higher accuracy on all testing schemes, but also successfully attacking schemes where others have zero chance. We show that our approach is highly efficient as it can solve a captcha within 0.05 second using a desktop GPU. We demonstrate that our attack is generally applicable because it can bypass the advanced security features employed by most modern

text captcha schemes. We hope the results of our work can encourage the community to revisit the design and practical use of text captchas.

## CCS CONCEPTS

• **Security and privacy** → **Authentication; Graphical / visual passwords; Access control;**

## KEYWORDS

Text-based CAPTCHAs; deep learning; transfer learning; generative adversarial networks

## ACM Reference Format:

Guixin Ye, Zhanyong Tang\*, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. 2018. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*, October 15–19, 2018, Toronto, ON, Canada. ACM, New York, NY, USA, 17 pages. <https://doi.org/10.1145/3243734.3243754>

## 1 INTRODUCTION

Text-based captchas are extensively used to distinguish humans from automated computer programs [58–60]. While numerous alternatives to text-based captchas have been proposed [2, 12, 44, 50], many websites and applications still use text-based captchas as a security and authentication mechanism. These include the majority of the top-50 popular websites ranked by alexa.com as of April 2018, including Google, Microsoft, Baidu, and many others. Due to the wide deployment of text-based captchas, a compromise on the scheme can have significant implications and could result in serious consequences.

Breaking captchas<sup>2</sup> is certainly not a new research topic. Over the past decade, researchers have demonstrated different ways for automatically recognizing text-based captchas [15, 16, 43, 64]. However, many of the prior attacks are hard-coded for a few specific captcha schemes, and tuning the attacking heuristics or models requires heavy expert involvement and follows a labor-intensive and time-consuming process of data gathering and labeling. Since

<sup>2</sup>In this paper, the term *breaking captchas* means automatically solving the captcha challenge using a computer program, i.e., recognizing the characters within a text-based captcha image.

\*Corresponding faculty authors: Zhanyong Tang and Zheng Wang.  
<sup>1</sup>To aid readability, we will use the acronym in lowercase thereafter.

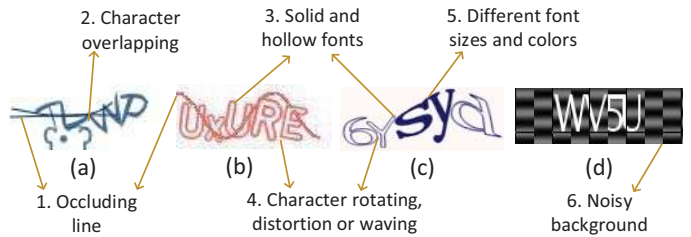
text captchas are keeping evolving and have become more robust, the newly introduced security features make many of the previous scheme-specific attacks no longer applicable [17]. Recently, some more generic attacks were proposed [8, 10, 17]. However, these methods only target text captchas with relatively simple security features such as simple noisy backgrounds and a single font style. The success of these generic attacks lies on the effectiveness of character segmentation [11], but the recent development of text captchas has made it more challenging by introducing e.g., more complex backgrounds as well as distorted and overlapping characters.

This paper presents a generic, low-effort yet effective approach to automatically solve text-based captchas based on deep learning [31, 48]. Unlike previous machine-learning-based attacks [55] that all require a large volume of captchas (which are increasingly difficult to gather) to train an effective solver, our approach significantly reduces the number of real captchas needed. We achieve this by first using automatically generated synthetic captchas to train a base solver and then fine-tune the basic model by applying transfer learning [48] to a small set of real captchas of the target scheme. Our approach is based on the recently proposed generative adversarial network (GAN) architecture that has demonstrated impressive performance on image translation tasks [22, 31]. Our method not only greatly reduces the human involvement and efforts needed in building a successful captcha solver, but also yields significantly better performance in solving a wide range of modern captcha schemes. Since our attack requires little human involvement, a captcha solver can be easily built to target a new or revised captcha scheme. This makes our attack a particular serious threat for text-based captchas.

We evaluate our approach by applying it to a total of 33 text-based captcha schemes, of which 11 are currently being used by 32 of the top-50 popular websites ranked by alexa.com as of April, 2018. These include schemes being used by Google, Microsoft, eBay, Wikipedia and Baidu, many of which employ advanced security features. We demonstrate that our generic attack needs as few as 500 real captchas instead of millions [21] to learn a text-based captcha solver, but the resulting solver can significantly outperform four state-of-the arts [8, 10, 17, 19]. Experimental results show that our approach can successfully crack all testing schemes, judged by the commonly used standard [10], and solve a captcha in less than 50 milliseconds using a desktop GPU.

This paper makes the following contributions:

- We present the first GAN-based approach for automatically generating training data and constructing solvers for text-based captchas (Section 4.1).
- We apply, for the first time, transfer learning to train text-based captcha solvers. Our approach reduces the number of real captchas needed for building an effective solver by several orders of magnitudes when compared with prior machine-learning-based attacks (Section 4.3).
- Our work provides new insights, showing that the security features employed by the current text-based captcha schemes are particularly vulnerable under deep learning methods (Section 6).



**Figure 1: Captcha security features targeted in this work. Examples in (a), (b), (c) and (d) are samples collected from Baidu, Sina, Microsoft and JD captcha schemes, respectively.**

## 2 BACKGROUND

In this section, we describe the threat model and introduce the GAN architecture.

### 2.1 Threat Model

In this work, we assume that the adversary can access and correctly label some text-based captchas of the target scheme. Since our approach can work effectively using no more than 500 captchas collected from the target scheme, we consider the overhead of collecting and labeling captchas to be low. We also assume the attacker has the computation power to generate synthetic captchas, and to train and deploy the solver. Later in the paper, we show that a modern GPU cloud server will provide sufficient computation power for launching the attack.

Without loss of generality, to make our experiments manageable, we restrict our scope to six widely used security features employed by the current text captcha schemes. These security features (as illustrated in Figure 1), including anti-segmentation and anti-recognition features. They are used by the top-50 popular websites ranked by alexa.com at the time this work was conducted. Specifically, an anti-segmentation feature makes it harder for a bot program to segment the characters. The features labeled as 1, 2 and 6 in Figure 1 give some of the anti-segmentation features targeted in this work. In a similar vein, an anti-recognition feature increases the difficulty of character recognition by using a variety of font styles. The features labeled as 3, 4 and 5 in Figure 1 illustrate some of the anti-recognition features investigated in the work. More details on how these features are used by each evaluated captcha scheme is given later in Table 1.

### 2.2 Generative Adversarial Networks

Our attack is based on the recently proposed GAN architecture [22]. A GAN consists of two models: a *generative* network for creating synthetic examples and a *discriminative* network to distinguish the synthesized examples from the real ones. We use backpropagation [28] to train both networks, so that over the training iterations, the generator produces better synthetic samples, while the discriminator becomes more skilled at flagging synthetic samples. GANs have shown impressive results in image [31, 67] and natural language [39, 66] processing tasks. However, due to the newness of the technique, no work to date has yet exploited GANs to develop a generic solver for text-based captchas.

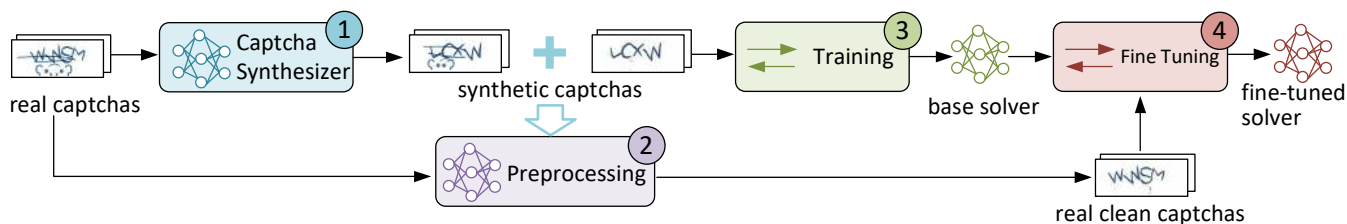


Figure 2: Overview of our approach. We first use a small set of real captchas of the target scheme to learn a captcha synthesizer ①. The captcha synthesizer is then used to automatically generate synthetic captchas (with and without background confusion) to learn a pre-processing model to remove security features, e.g., noisy backgrounds and occluding lines, from the input captcha image ②. At the same time, the synthetic captchas (with and without security features) are used to train a base solver ③. The base solver is then refined to build the final, fine-tuned solver using a few real clean captchas ④.

### 3 OVERVIEW OF OUR APPROACH

Figure 2 depicts the four steps of building a captcha solver using our approach. Each of the step is described as follows.

**Step 1. Captcha synthesis.** The first step is to generate captchas that are visually similar to the target captchas. Our GAN-based captcha generator consists of two parts: a captcha generator that tries to produce captchas which are as similar as possible to the target captchas, and a discriminator that tries to identify the synthetic captchas from the real ones. This generation-discrimination process terminates when the discriminator *fails* to identify a large portion of the synthetic captchas. Once training has terminated, we can then use the trained generator (referred as captcha synthesizer) to automatically generate an unbounded number of captchas (for which the characters of each synthetic captcha are known). This is detailed in Section 4.1.

**Step 2. Preprocessing.** Before presenting a captcha image to a solver, we use a pre-processing model to remove the captcha security features and standardize the font style (e.g., filling hollow characters and standardizing gaps between characters). The pre-processing model is based on a specific GAN called Pix2Pix [14]. It is trained from *synthetic captchas* for which we also have the corresponding clean captchas (i.e., captcha images without security features). The trained model can then be used for any *unseen* captchas of the target captcha scheme. This is detailed in Section 4.2.

**Step 3. Training the base solver.** With the captcha synthesizer and the pre-processing model in place, we then generate a large number of synthetic captchas together with their labels (i.e., corresponding characters) and use this dataset to learn a base solver for a target captcha scheme. Our captcha solver is a convolutional neural network (CNN). The trained solver takes in a pre-processed captcha image and outputs the corresponding characters. This process is described in more details at Section 4.3.

**Step 4. Fine-tuning the base solver.** In the last step, we apply transfer learning to refine the base solver by using a small set of manually labeled captchas that are collected from the target website. Transfer learning allows us to leverage knowledge learned from synthetic captchas to reduce the cost of collecting and labeling captchas, and to further improve performance of the base model. This is described in Section 4.3.

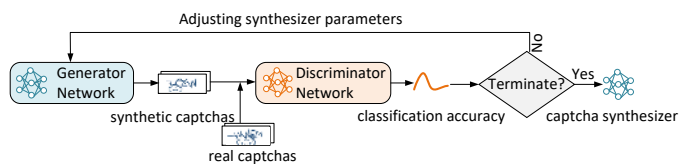


Figure 3: The training process of our GAN-based text captcha synthesizer.

## 4 IMPLEMENTATION DETAILS

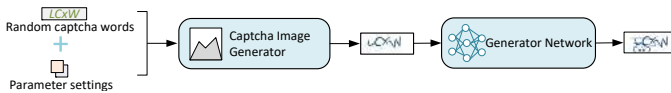
We now describe how to build the captcha synthesizer (Section 4.1), pre-processing model (Section 4.2) and solver (Section 4.3) in more details.

### 4.1 Captcha Synthesizer

Deep neural networks typically require a large volume of training examples to learn an effective model. Prior work shows that to build an effective CNN-based captcha solver would require over 2.3 million unique training images [19]. Collecting and manually labelling such number of real captchas would require intensive human involvement and incur significant costs. In this paper, we show that it is possible to minimize the human involvement and the associated costs via captcha synthesis. The idea is that using a captcha synthesizer, we can populate the training data with an unbounded number of synthetic captchas (that are similar to the real captchas). This allows the training data to cover the problem space far more finely than what could be achieved by exclusively using human-labelled training data.

Figure 3 illustrates the process of training a captcha synthesizer using GANs. The training process is largely automatic except that a user needs to provide a small set of real captchas (500 in this work) of the target captcha scheme, and to define the set of security features. The security feature definition is achieved by configuring a set of pre-defined parameters. Figure 5 lists the set of security parameters considered in this work and the specific settings for the Baidu captcha scheme. We stress that these parameters can be easily extended and adjusted to target other captcha schemes.

Our captcha synthesizer consists of two components, a generator and a discriminator. The generator,  $G$ , is trained to produce outputs that cannot be distinguished from real captchas by an adversarially



**Figure 4: Our captcha generator model includes a image generator and a generator network. The image generator produces a captcha image at the word level, and the generator network modifies the produced captcha image at the pixel level to add security features.**

trained discriminator,  $D$ , which is trained to do as well as possible at detecting the synthetic captchas.

**Captcha generator.** As depicted in Figure 4, our captcha generator model includes a captcha image generator which automatically generates captcha images according to a given parameter setting and captcha word, and a CNN model that modifies the generated synthetic captcha at the pixel level. We provide the image generator and the learning engine a large number of free fonts so that the learning engine can learn which font best suits the target scheme. The image generator takes in the security feature configuration setting provided by a user and tries to find a set of configurable parameter values so that the synthetic captchas are as similar as possible to the ones from the target captcha scheme. We use the grid search method presented in [4] to search for the optimal parameters for a given captcha scheme. Like the image generator, the CNN model learns how to modify the generated images at the pixel level so that the resulting captcha contains security features that are similar to the real ones of the target scheme. The similarity is measured by the ratio of synthetic captchas that cannot be distinguished from the real ones by the discriminator. In other words, the more synthetic captchas that can “fool” the discriminator, the higher quality the synthetic captchas are.

**Captcha discriminator.** We use the discriminator network defined in [52], which is a convolutional network whose last layer outputs the probability of an input captcha being a synthetic one. We use batched captchas to train the discriminator, where each mini-batch consists of randomly sampled synthetic captchas,  $x_i$  and real captchas,  $y_j$ , and the target labels are 0 for every  $y$  and 1 for every  $x$ .

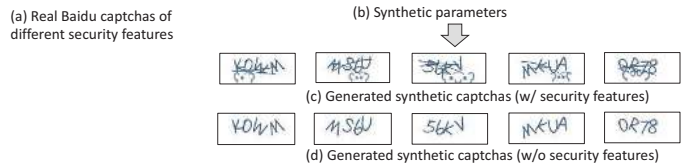
The discriminator network updates its parameters by minimizing the following loss function:

$$\mathcal{L}_D = - \sum_i \log D(x_i) - \sum_j \log(1 - D(y_j)) \quad (1)$$

which is equivalent to cross-entropy error for a two class classification problem where  $D(\cdot)$  is the probability of the input being a synthetic captcha, and  $1 - D(\cdot)$  that of a real one. *We note that the real captchas in training are different from the one used to test our approach.*

**Training.** We use the minibatch stochastic gradient descent (SGD) and the Adam solver [34] with a learning rate of 0.0002 to train our captcha synthesizer. The overall training objective follows the general GAN approach [52], using the  $L_1$  norm with the regularization term  $\lambda$  set to 0.0001. The training objective is defined as:

Security Feature	On/Off	#Options	Value Range
Noisy background(s)	On	5	[10, img.width]
Occluding lines	On	2	(Line, Sin, Quadratic, Bezier)
Char. Overlapping	On	-	[-3, 10]
Character set	On	4	[A-Z]
Font style(s)	On	1	Solid
Font color(s)	On	1	RGB (65, 103, 141)
Distortion	On	-	{{[0.1, 0.2], [0.2, 0.3]}}
Rotation	On	-	[-30, 30]
Waving	Off	-	-

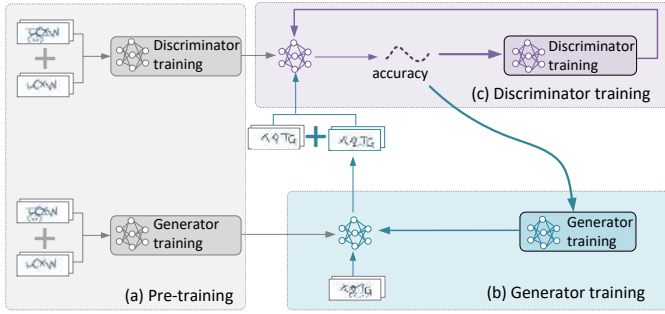


**Figure 5: Example synthetic captchas for the Baidu scheme. Our captcha synthesizer is trained using a set of real captchas (a). The parameter setting (b) defines the security feature space. The trained captcha synthesizer is used to produce synthetic captchas with (c) and without (d) the security features (i.e., noisy backgrounds and occluding lines in this example) included.**

$$G^* = \arg \min_G \max_D \mathcal{L}_{GAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \quad (2)$$

where the generator,  $G$ , tries to minimize the difference between the generated captchas and the real ones, while the discriminator,  $D$ , tries to maximize it. During training, when updating the parameters of the generator, we fix the parameters of the discriminator; and when updating the discriminator, we fix the parameters of the generator. Training terminates when the discriminator fails to identify more than 5% of the synthetic captchas. Training the synthesizer takes around 2 days for one captcha scheme on our platform. The trained generator network (together with the captcha image generator) can then be used to quickly generate synthetic captcha images. In our case, it takes less than one minute to generate one million captchas images.

**Example.** We use the Baidu captcha scheme to explain the process for training the captcha synthesizer. To initialize the training, we provide a set of real captchas for the GAN learning engine and initial parameter values for the image generator. The generator then produces a batch of synthetic captchas which are examined by the discriminator. If the discriminator can successfully distinguish a large number of synthetic captchas from the real ones, the grid search method is employed to adjust the parameter values for synthesizing another batch of captchas. This process continues until the discriminator can distinguish less than 5% of the synthetic captchas from the real ones (see Section 6.6). When the process is terminated, the learning engine will output the optimal parameter values that are used by the image generator and the generator network for synthesizing captcha images. As an example, Figure 5 (a) shows a real Baidu captcha while (b) and (c) in Figure 5 are the synthetic captchas with and without background and security features produced by our approach. As can be seen from the figure, the



**Figure 6: The training process of our GAN-based pre-processing model. The generator tries to remove as much noisy backgrounds and occluding lines from the input captchas, while the discriminator tries to identify which of the input clean captchas are produced by the generator. All the captchas used in the training are generated by our captcha synthesizer.**

security features of the synthetic captchas are visually similar to the real captchas.

## 4.2 Captcha Preprocessing

Previous successful attacks have led to the development of more robust text-based captchas that include advanced security features like occluding lines (e.g., Figure 1a) and distorted hollow fonts (e.g., Figure 1 b and c). These features make the previous pre-processing methods like [16, 64] inapplicable (see Section 6.3).

To remove these security features, we turn again to employ deep learning to build a pre-processing model. The goal of our pre-processing model is to remove noise and occluding lines from the background and to standardize the font style (such as filling hollow parts of characters and widening and standardizing the gap between two characters - see also Section 6.3). Specifically, we adapt the *Pix2Pix* image-to-image translation framework [14]. This algorithm was developed to transform an image from one style to another. In our case, the images to be translated are captcha images with background noises such as the Baidu captchas (Figure 1a) or different font styles such as the Microsoft captchas (Figure 1c). Our model is also able to remove multiple security features (e.g., Figure 5b) at once. It is to note that we train a pre-processing model for each captcha scheme using synthetic data.

Our pre-processing model is also a GAN consisting of a generator and discriminator. The training goal is to learn a generator to remove security features and standardize the font style. Figure 6 illustrates the training process of our pre-processing model. The generator works at the pixel level, which tries to amend some pixels of the input captcha image to e.g., remove noise from the background (Figure 6b). By contrast, the discriminator tries to distinguish the pre-processed captchas from the *clean captchas* that are produced by the captcha synthesizer described in Section 4.1.

To train the pre-preprocessing model, we first learn an initial discriminator and generator using some synthetic captchas (Figure 6a). The training captchas are organized as pairs where each pair contains a synthetic captcha with the target security features enabled

(e.g., noisy backgrounds and occluding lines) and a corresponding captcha with these security features excluded. Since the training captchas are generated by our captcha synthesizer, it is trivial to exclude the security features from the generation process. After having the initial discriminator and generator, we then train them under the generative adversarial framework. The process is similar to how we train our captcha synthesizer (Section 4.1). Over time, the generator would become better in removing security features, i.e., the resulting captchas are increasingly like the clean captchas; and the discriminator would become better in recognizing security features of the captcha (even the changes are small). Training terminates when the discriminator fails to identify more than 5% of the generated captchas from the clean counterparts (Figure 6c). After that, we use the trained generator to pre-process *unseen* captcha images of the target scheme.

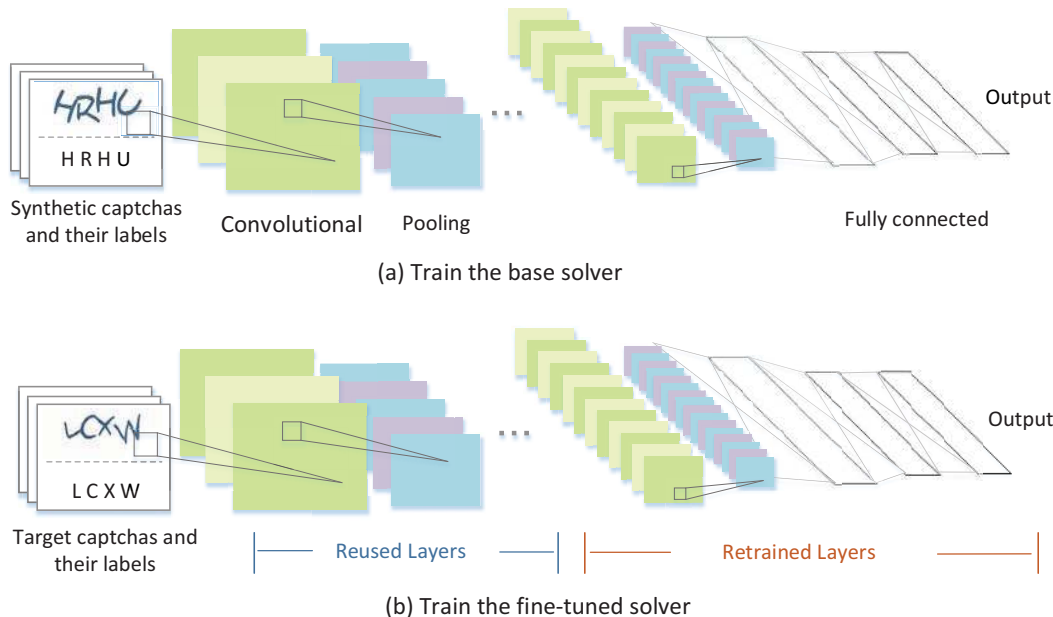
## 4.3 Captcha Solvers

To build a captcha solver, we follow a two-step approach. We first learn a base solver from *synthetic* captchas. We then fine-tune the base solver using the same set of *real* captchas used to build the captcha synthesizer.

**4.3.1 Solver model structure.** Our captcha solver tries to recognize the characters of a pre-processed captcha image. The solver is based on a classical CNN called *LeNet-5* [38]. We have also considered other influential CNN structures including ResNet [27], Inception [56] and VGG [53]. We found that there is little difference for solving text-based captchas among these models. We choose LeNet-5 due to the simplicity of the network, which gives the quickest inference (i.e., prediction) time and requires least training data for applying transfer learning. We use the same network structure for the base and the fine-tuned solvers, but we train a solver for each captcha scheme using synthetic data.

*LeNet-5* was originally proposed to recognize single characters but we introduce some additional layers (2x convolutional and 3x pooling layers) to extend its capability to recognize multiple characters. Figure 7a shows the structure of our solver which has five convolutional layers, five pooling layers followed by two fully-connected layers. Each of the convolutional layer is followed by a pooling layer. We use a  $3 \times 3$  filter for the convolutional layer and a max-pooling filter for the pooling layer. We use the default parameters of *LeNet-5* for the rest of the network structures.

The output layer of our solver consists of a number of neurons, one neuron for a character of the target scheme. For example, if a captcha scheme uses  $n$  characters, the output layer will consist of  $n$  neurons where each neuron corresponds to a candidate character. Each neuron applies an activation function  $f(x)$  over its inputs. The activation of each neuron represents the model's confidence that the corresponding character is the correct one. To obtain the predicted characters, we find the neurons with the largest activations for a given captcha scheme and map the chosen neurons to the corresponding characters. For example, for a captcha scheme of four characters, we will choose the four neurons with the largest activation values and then translate the chosen neurons to the corresponding characters.



**Figure 7: Overview of our CNN based captcha solver. The base solver is trained using synthetic captchas (a), which is then refined using a small number (500 in this work) of real captchas (b).**

**4.3.2 Training the base solver.** We train a base solver for each target captcha scheme. If the number of characters in a captcha from a scheme is not fixed, we also train a base solver for each possible number of characters. We use 200,000 synthetic captchas generated by our scheme-specific captcha synthesizer to train a base solver. Each training sample consists of a captcha image (without security features) and an integer vector that stores the character IDs of the captcha. Note that we assign a unique ID to each candidate character of the target captcha scheme. We use a Bayesian based parameter tuner [20] to automatically choose the hyperparameters for training the base solver. Training a base solver takes around five hours using 4x NVIDIA P40 GPUs on a cloud server (see Section 5.2). The trained base solver can then be applied to any *unseen* captcha image of the target scheme. Note that before passing a raw captcha image to the solver, we first use the pre-processing model to remove the security features of the captcha image.

**4.3.3 Building the fine-tuned solver.** In the final step, we apply *transfer learning* [65] to update later layers (i.e., those that are closer to the output layer) of the base solver using a small set of manually-labeled real captchas. The idea of transfer learning is that in neural network classification, information learned at the early layers of neural networks (i.e. closer to the input layer) will be useful for multiple classification tasks. The later the network layers are, the more specialized the layers become [48]. Our work exploits this property to calibrate the base solver to avoid any bias and over-fitting that may arise from the synthetic training data.

Figure 7b illustrates the process of applying transfer learning to refine the base solver. Transfer learning in our context is as simple as keeping the weights of the early layers and then update the parameters of the later layers by applying the standard training

process using the real captchas. The fine-tuning process is quick, taking then less than 5 minutes on our training platform.

## 5 EXPERIMENTAL SETUP

In this section we describe our experimental parameters and evaluation platforms.

### 5.1 Data Preparation

We use two sets of captchas in this work: one for training and the other for testing. Most of our training data are synthetic captchas generated by our captcha synthesizer. To train and test our GAN-based synthesizer and the fine-tuned solver, we use in total 1,500 labeled, real captchas collected from the target website. From the 1,500 real captchas of a captcha scheme, we use 500 captchas for training and the remaining 1,000 captchas for testing. We make sure that the testing captchas are different from the ones used to train our models.

**Captcha schemes.** Our main evaluation targets 11 current text-based captcha schemes used by 32 of the top-50 popular websites ranked by Alexa<sup>3</sup>. We note that some of the websites use the same captcha scheme, e.g., Youtube uses the Google scheme, and Live, Office and Bing use the Microsoft scheme. The websites we examined cover a wide range of domains including e-commerce, social networks, search, and information portals. Table 1 lists the captcha schemes tested in this work and the target websites. We note that many captcha schemes exclude characters that are likely to cause confusion after performing the character distortion. Examples of such characters include ‘o’ and ‘0’, ‘1’ and ‘l’, etc. These excluded

<sup>3</sup>Data were collected between May, 2017 and April, 2018.

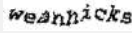










Scheme	Website(s)	Example	Security Features		Excluded Characters
			Anti-segmentation	Anti-recognition	
Wikipedia	wikipedia.org		Overlapping characters, English letters	Rotation, distortion, waving	-
Microsoft	{live, bing, micosoft}.com {office, linkedin}.com		Overlapping characters, solid background	Different font styles, varied font sizes, rotation, waving	0, 1, 5, D, G, I, Q, U
eBay	ebay.com		Overlapping characters, Only Arabic numerals	Character rotating, distortion and waving	-
Baidu	{baidu, qq}.com		Occluding lines, overlapping, only English letters	Varied font size, color, rotation, disortion and waving	Z
Google	google.{com,co.in,co.jp,co.uk,ru,com.br,fr,com.hk,it,ca,es,com.mx}youtube.com		Overlapping characters, English letters	Varied font sizes & color, rotation, disortion, waving	-
Alipay	{alipay, tmall}.com {taobao, login.tmall}.com alipayexpress.com		English letters and Arabic numerals, overlapping characters	Rotation and distortion	0, 1, I, L, O
JD	jd.com		English letters and Arabic numerals, overlapping characters	Rotation and distortion	0, 1, 2, 7, 9, D, G, I, J, L, O, P, Q, Z
Qihu360	360.cn		English letters and Arabic numerals, overlapping characters	Varied font sizes, rotation and distortion	0, I, L, O, T, i, l, o, t, q
Sina	sina.cn		English letters and Arabic numerals, overlapping characters	Rotation, distortion, waving	1, 9, 0, D, I, J, L, O, T, i, j, l, o, t, g, r
Weibo	weibo.cn		English letters and Arabic numerals, overlapping characters, occluding lines	Rotation and distortion	0, 1, 5, D, G, I, Q, U
Sohu	sohu.com		Complex background, occluding lines, and overlapping	Varied font size, color and rotation	0, 1, i, l, o, z

Table 1: Text-based captcha schemes tested in our experiments.

characters are also given in Table 1. When compared to prior attacks, we extend our evaluation to 22 other captcha schemes used in prior studies. These captcha schemes are listed in Table 4. It is worth mentioning that while we collected the captchas from the official websites, many of the captcha schemes we tested are also used by third-party websites and applications as a security mechanism.

**Synthesizing training captchas.** To generate synthetic captchas for a target scheme, we first initialize the security feature parameters as described in Section 4.1. We then use the initial parameters to generate the first batch of synthetic captchas which are used together with 500 real captchas to automatically train our synthesizer. Once we have trained the synthesizer, we then use it to generate synthetic samples to learn the preprocessing model and the base solver. Specifically, we use 20,000 and 200,000 synthetic captchas to train the pre-processing model and the base solver respectively.

**Collecting testing captchas.** The real captchas are automatically collected using a web crawler written in Python. Each collected captcha is manually labeled by three paid participants (nine participants in total) recruited from our institution. We use only captchas

where a consensus has been reached by all the three annotators. In total, we have used 1,500 real captchas for each target scheme. We randomly divided the collected captchas to two sets, one set of 500 captchas for training our synthesizer and final solver, and the other set of 1,000 captchas for testing our solver. It takes up to 2 hours (less than 30 minutes for most of the scheme) to collect 500 captchas and less than 2 hours to label them by one user. This means that the effort and cost for launching our attack on a particular captcha scheme is low.

## 5.2 Implementation and Evaluation Platforms

Our prototype system<sup>4</sup> is implemented using Python. Specifically, the pre-processing model is built upon the *Pix2Pix* framework [14], implemented using Tensorflow v. 1.2.1, and the captcha solver is coded using Keras v. 2.1.2.

We use two different hardware platforms. For training, we use a cloud server with a 2.4GHz Intel Xeon CPU, four NVIDIA Tesla P40 GPUs and 256GB of RAM, running Centos 7 operating system with Linux kernel 3.10. The trained solver is then run and tested

<sup>4</sup>Code and data will be released at: <https://goo.gl/92VxXC>

Scheme	Success rate		Running Time per Captcha (ms)
	Base Solver	Fine-tuned Solver	
Sohu	83%	92%	43.78
eBay	52%	86.6%	4.22
JD	60%	86%	43.18
Wikipedia	7%	78%	4.71
Microsoft	36.6%	69.6%	46.06
Alipay	23%	61%	3.75
Qihu360	48.6%	56%	3.10
Sina	40.6%	52.6%	42.81
Weibo	4.7%	44%	3.41
Baidu	6%	34%	41.57
Google	0%	3%	4.02

Table 2: The overall success rate and solver running time for each captcha scheme.

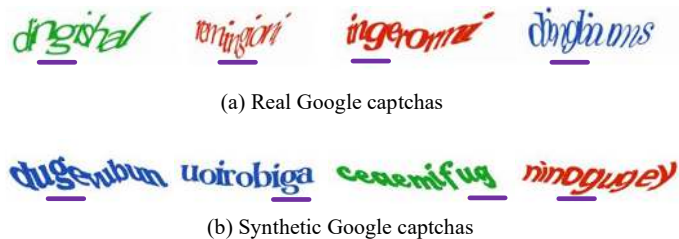


Figure 8: Real Google captchas and the synthetic versions generated by our captcha synthesizer.

on a workstation with a 3.2GHz Intel Xeon CPU, a NVIDIA Titan GPU and 64GB of RAM, running the Ubuntu 16.04 operating system with Linux kernel 4.10. All trained models run on the NVIDIA Titan GPU for inference.

## 6 EXPERIMENTAL RESULTS

In this section, we first present the overall success rate of our approach on 11 current captcha schemes, showing that it can break all the testing schemes – judged by the criterion commonly used by the captcha community [10]. We then compare our approach against prior attacks on another 22 text captcha schemes, demonstrating that our approach significantly outperforms all prior attacks. Finally, we analyze the working mechanism our approach before discussing the implications of the attack.

### 6.1 Evaluation on Current Captcha Schemes

Table 2 shows the success rate given by our base and fine-tuned solvers, as well as the average time in solving a captcha. All the tests were performed on real captchas which are not used for training the solvers. Since the base and the fine-tuned solvers use the same network structure, there is no difference in their running time. For each captcha scheme, we report the average running time across 1,000 tested captchas. We observe little variation in the running time, less than 0.5% across test runs.

Scheme	Captcha Image	Ground Truth	Solver Output
Sohu		d4sk	d4sh
eBay		934912	994912
JD		BHER	BFER
Wikipedia		mewsboxes	mewsbores
Microsoft		XK6NK	XK6VK
Alipay		B7JK	B7YK
Qihu360		s34Ea	s3VFa
Sina		nG3uu	nG3uv
Weibo		4TXB	4TX8
Baidu		WFIH	WFEH
Google		irgandoca	igiruloca

Table 3: Example incorrectly labeled captchas.

Our base solver, built from synthetic data, is able to solve most of the captcha schemes with a success rate of over 20%. This demonstrates the capability of CNN models in performing image recognition. However, it gives a low success rate for some of the schemes such as Weibo (4.7%) and Google (0%). The fine-tuned solver, refined using transfer learning, significantly boosts the performance of the base solver. In particular, it improves the success rate for Wikipedia from 7% to 78%, Weibo from 4.7% to 44%, Alipay from 23% to 61% and Microsoft from 36.6% to 69.6%. This result shows that transfer learning in combination of captcha synthesis can reduce the data collection efforts for building an effective text-based captcha solver.

Furthermore, the fine-tuned solver also improves the success rate for Google from 0% to 3%. While this success rate is lower than other schemes because of the strong security features like distorted characters and dynamic font styles employed by the Google scheme. The strong security features make it difficult our synthesizer to generate high-quality synthetic data. This is depicted in Figure 8 where our synthetic captchas are not similar enough to the real captchas (especially for the font styles). However, 3% is still above the 1% threshold for which a captcha is considered to be ineffective [10]. We also note that there is no prior attack can successfully crack the current Google captcha scheme under this criterion.

Table 3 gives some example captchas that are incorrectly labeled by our fine-tuned solver. For most of these captchas, our solver only incorrectly labels one character and the mis-identified character is similar to the ground-truth character. For example, for the eBay captcha shown in Table 3, our solver incorrectly label character “3” to “9” due to character overlapping. For the Google scheme, our solver often fails to label several characters in the middle due to excessive character distortion and overlapping. Note that for



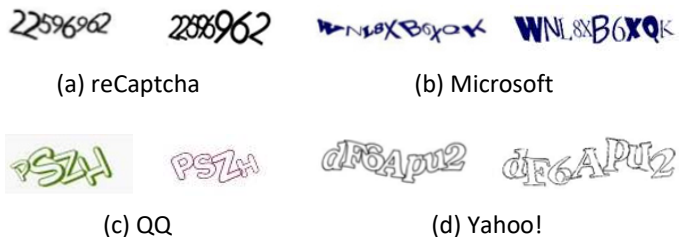


Figure 9: Examples of the captcha schemes (left) tested in prior work, and the synthetic versions (right) generated by our captcha synthesizer. Our captcha synthesizer is highly effectively in synthesizing captcha images.

many of these incorrectly labeled captchas, our annotators were also struggling to recognize the characters in the first attempt.

Finally, the running time for solving a captcha using our solvers is negligible, taking less than 50 milliseconds on a NVIDIA Titan Desktop GPU. Overall, our approach can solve all the testing schemes under the commonly used criterion [10] with a quick running time.

## 6.2 Compare to Prior Attacks

In this experiment, we compare our approach against four state-of-the-art attacks [8, 10, 17, 19] on 24 distinct captcha schemes, including the eBay and Wikipedia schemes from Table 1 and other 22 schemes. To provide a fair comparison, we try to use captchas in total that these methods were tested on. Whenever possible, we use the same dataset or captchas from the original scheme where the prior work was tested on. For those obsolete schemes (21 out of 24 schemes), we collected the test data from public datasets, or using captcha generation tools developed by independent researchers. Specifically, we use (1) public datasets of previous captcha schemes, (2) online captcha generators, such as captchas.net which was used by some of the previous captcha schemes, and (3) open-sourced captcha generators used in the prior work.

For each captcha scheme, we collected 1,500 samples for which we use 500 for training and 1,000 for testing. Figure 9 gives some examples of the real captchas and the one produced by our synthesizer. The figure suggests that our synthesizer can produce captchas that are visually similar to real examples from the target scheme.

Table 4 compares our fine-tuned solver to previous attacks. In this experiment, our approach outperforms all comparative schemes by delivering a significantly higher success rate. For many of the testing schemes, our approach boosts the success rate by 40%. It can successfully solve all the captchas of Blizzard, Megaupload and Authorize used in [10]. Our approach achieves a success rate of 87.4% and 90% for reCAPTCHA 2011 and 2013 respectively. This scheme was previously deemed to be strong where the human accuracy is 87.4% [19]. As a result, our solver matches the capability of humans in solving reCAPTCHA. To achieve a comparable accuracy for reCAPTCHA, a CNN-based captcha solver [21] would require 2.3 million unique real captcha images [19].

We want to stress that unlike all the competitive approaches which require manually tuning a character segmentation method, our approach bypasses this process by learning an end-to-end solver.

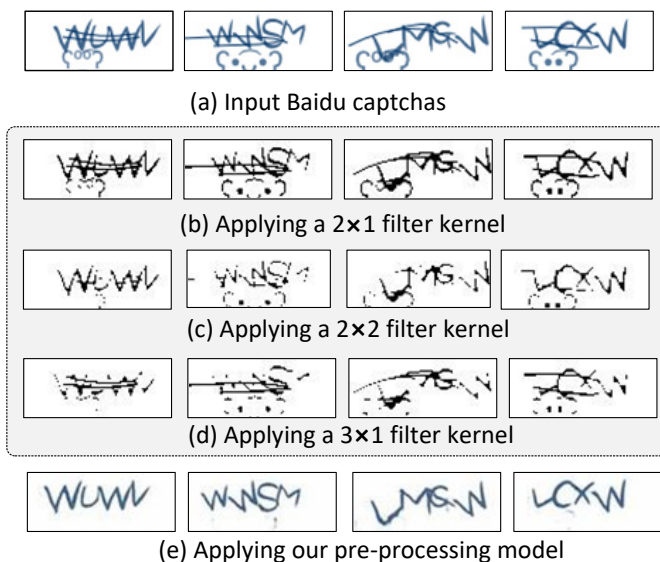


Figure 10: Comparing a filtering-based method with our approach for removing noisy backgrounds and occluding lines. The filtering-based method fails to remove security features from the latest captcha schemes while our approach can.

As a result, our approach requires less expert involvement, yet it delivers better performance.

## 6.3 Pre-processing Security Features

One of the key steps in solving text-based captchas is to remove the security features and standardize the font style of an input captcha image. In this evaluation, we compare our pre-processing model against prior pre-processing methods on removing noisy backgrounds [8, 10, 32] and standardize font styles [11, 16].

**Remove security features.** Filtering is often used in prior attacks for pre-processing text-based captchas [8, 10, 32]. The idea is to apply a fix-sized window, or filter kernel, throughout the image to remove the occluding lines and noise while keeping edges of the characters. Figure 10 compares a previously used filtering method [8, 10, 32] against our automatically learned pre-processing model. Finding the right filter kernel size for the input captchas shown in Figure 10a is non-trivial, because the filter either fails to eliminate the background and occluding lines (b and c in Figure 10) or it over does it by eroding edges of the characters (which makes it harder to recognize the characters). While filtering was effective for prior text-based captchas, the latest captcha schemes have introduced more sophisticated security features which make filtering no longer feasible. In contrast to filtering, our pre-processing model can successfully eliminate nearly all the background noise and occluding lines from the input image, leading to a much cleaner captcha image while keeping the character edges, as depicted in Figure 10a. This experiment shows that our pre-processing model is highly effective in processing and removing security features from the latest text captcha schemes.

Captcha Scheme	Captcha Example	Success rate		Captcha Scheme	Captcha Example	Success rate	
		Ref. [10]	Our approach			Ref. [17]	Our approach
Megaupload		93%	100%	Baidu (2016)		46.6%	97.5%
Blizzard		70%	100%	QQ		56%	94%
Authorize		66%	100%	Taobao		23.4%	90.7%
Captcha.net		73%	99.6%	Sina		9.4%	90%
NIH		72%	99%	reCAPTCHA (2011)		77.2%	87.4%
Reddit		42%	98%	eBay		58.8%	86.6%
Digg		20%	95%	Amazon		25.8%	79%
eBay		43%	86.6%	Wikipedia		23.8%	78%
Slashdot		35%	86.4%	Microsoft		16.2%	72.1%
Wikipedia		25%	78%	Yahoo! (2016)		5.2%	63%

Captcha Scheme	Captcha Example	Success rate		Captcha Scheme	Captcha Example	Success rate	
		Ref. [8]	Our approach			Ref. [19]	Our approach
reCAPTCHA (2013)		22.3%	90%	PayPal		57.1%	92.4%
Baidu (2013)		55.2%	89%	reCAPTCHA (2011)		66.6%	87.4%
reCAPTCHA (2011)		22.7%	87.4%	Yahoo! (2016)		57.4%	63%
eBay		51.4%	86.6%				
Baidu (2011)		38.7%	83.1%				
Wikipedia		28.3%	78%				
Yahoo! (2014)		5.3%	75.1%				
CNN		51.1%	51.6%				

Table 4: Comparing our approach against four prior attacks [8, 10, 17, 19] on 24 captcha schemes where the prior methods were tested on. These captcha schemes include eBay and Wikipedia evaluated in Section 6.1 and other 22 schemes.

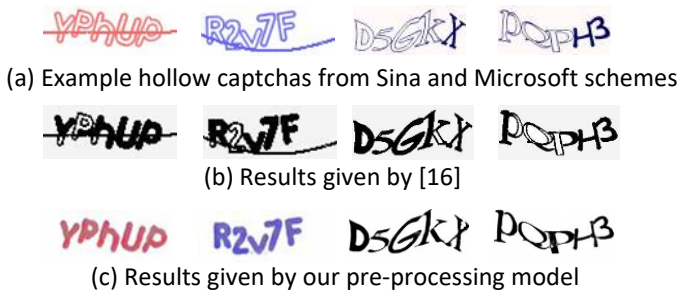


Figure 11: Comparing font style standardization between a state-of-the-art hollow captcha solver [16] and our pre-processing model. Our pre-processing model is able to fill the hollow parts more effectively.



Figure 12: Character segmentation produced by our pre-processing model. For each scheme, the left image is the input captcha, and the right image is the output of our pre-processing model.

**Filling hollow characters.** Figure 11 compares our pre-processing model against a state-of-the-art hollow captcha solver [16]. The task in this experiment is to fill the hollow parts of the characters. Here, we apply both schemes to the testing hollow captchas from Sina and Microsoft schemes. Figure 11a gives some of the examples from these two schemes, while Figure 11b and Figure 11c present the corresponding results given by the hollow filling method in [16] and our approach respectively. As can be seen from the diagrams, our pre-processing model is able to fill most of the hollow strokes, while the state-of-the-art method leaves some hollow strokes unfilled. Therefore, our approach is more effective in standardizing the font style. We also note that unlike prior attacks which require manually designing and tuning an individual method to process each security feature, our approach automatically learns how to process all features at one go. As a result, our approach requires less effort for implementing a holistic pre-processing model.

**Standardizing character gaps.** Prior research suggests that the robustness of text captchas largely relies on the difficulty of finding where the character is (i.e., segmentation) rather than what character it is (i.e., recognition) [11]. This segmentation-resistance principle has become a crucial part for designing text captcha schemes. The examples given in Figure 12 suggest that our pre-processing model is effectively not only in removing security features (like noisy backgrounds and occluding lines) and standardizing font styles, but also in segmenting characters by widening and standardizing the gap between collapsed characters. The high-quality

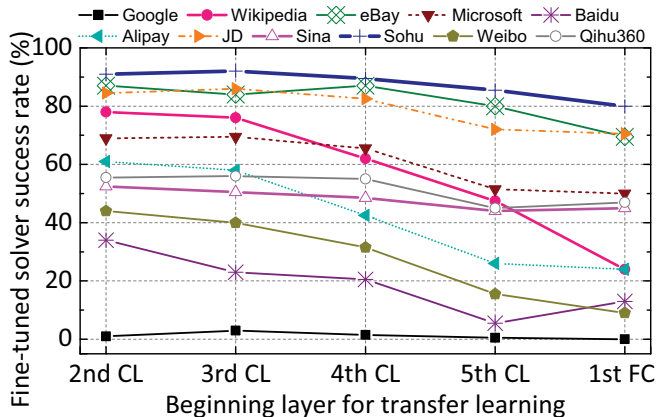


Figure 13: How the beginning layer for transfer learning affects the resulting performance of the fine-tuned solver.

character segmentation produced during pre-processing has a positive contribution to the success rate of our solver, helping it to achieve a higher accuracy compared to existing attacks.

## 6.4 Transfer Learning

Recall that we only use 500 real captchas to refine the base solver by employing transfer learning (Section 4.3). Our strategy for transfer learning is to only retrain some of the latter neural network layers of the base solver (see Figure 7). In this experiment, we investigate how the choice of transfer learning layers affects the performance of the fine-tuned solver. To that end, we apply transfer learning to different levels of the base solver, by changing the starting point of transfer learning from the 2nd convolutional layer (CL) all the way down to the first fully-connected layer (FC).

Figure 13 reports performance of the resulting fine-tuned solvers trained under different transfer learning configurations for the 11 captcha schemes given in Table 1. Overall, applying transfer learning to the second or third CL onward leads to the best performance. To determine the best starting layer for transfer learning, we apply cross-validation to the real captcha training dataset. Specifically, we divide the 500 real captchas into two parts, the first part of 450 captchas is used to refine the base solver, and the rest 50 captchas are used to validate the refined solver. We vary the beginning layer for transfer learning, and then test the refined base solver on the validation set to find out which beginning layer leads to the best performance. Since we only train and validate on 500 captchas, this process for finding the optimal beginning layer only takes several minutes on our training platform.

## 6.5 Impact of Fine-tuning Training Data Sizes

In this experiment, we evaluate how the number of real captchas used in transfer learning affects the success rate of the fine-tuned solver. Figure 14 shows the success rates of the fine-tuned solver when using different numbers of real captchas in transfer learning. When the number of training examples is 500, our approach reaches a high success rate. For most captcha schemes, the success rate drops significantly when the number of training examples less than 400. Nonetheless, our approach can achieve a high success rate when

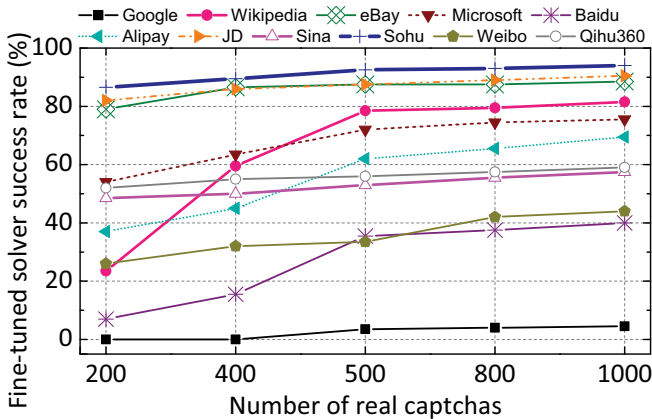


Figure 14: The achieved success rates when the fine-tuned solver is trained using different number of real captchas.

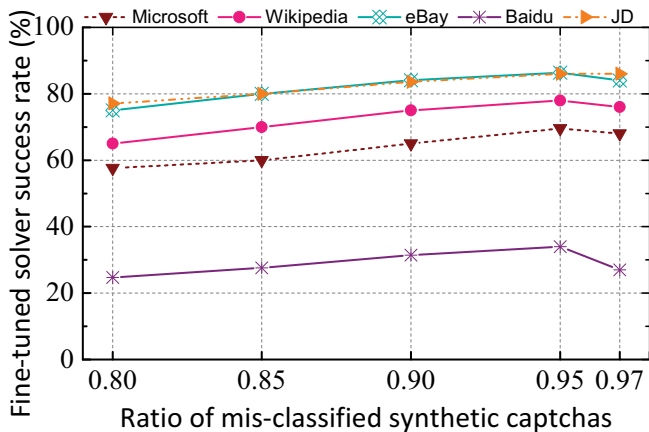


Figure 15: How the synthesizer training termination criterion affects the solver performance. Training terminates when the discriminator fails to classify a certain ratio of synthetic captchas.

the number of training examples is 500. Such a number allows an attacker to easily collect from the target website.

## 6.6 Synthesizer Training Termination Criteria

Our captcha synthesizer is trained under the GAN framework, and training terminates when the discriminator fails to classify a certain ratio of synthetic captchas (Section 4.2). Figure 15 reports how the termination criterion affects the quality of the synthetic captchas. The x-axis shows the ratio (from 0.8 to 0.97) of synthetic captchas that are misclassified as a real captcha by the discriminator when training terminates. The y-axis shows the success rate achieved by the fine-tuned solver for five current captcha schemes, where the base solver is trained on the resulting synthetic captchas using different termination criteria but the fine-tuned solver is trained on the same set of real captchas.

In general, the more synthetic captchas that the discriminator fails on, the higher the quality the generated synthetic captchas

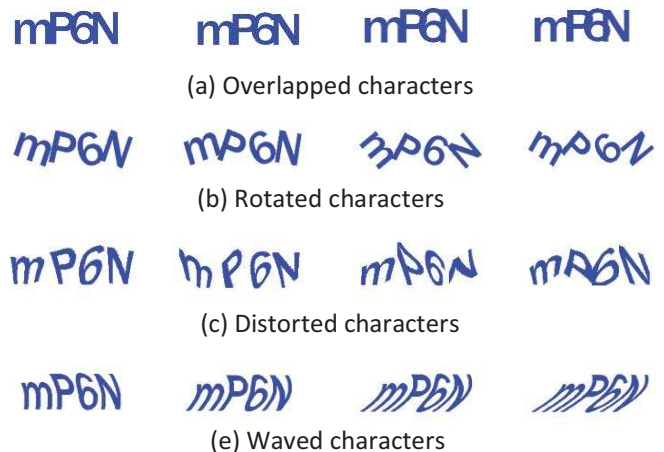


Figure 16: Example captchas with single security features.

will be, which in turns leads to a more effective captcha solver. However, the increase in the success rate reaches a plateau at 0.95. Further increasing the similarity of the synthetic captchas to real ones does not improve the success rate due to overfitting. Based on this observation, we choose to terminate synthesizer training when the GAN discriminator can successfully distinguish less than 5% (i.e., fail on 95% or more) of the synthetic captchas. We found that this threshold works well for all the captcha schemes tested in this work.

## 6.7 Impact of Captcha Security Features

In this experiment, we evaluate how security features affect the effectiveness of our solver. Having this knowledge is crucial for designing a more robust captcha scheme. This experiment considers four common security features for text captchas: overlapping, rotation, distortion, and waving. We exclude noisy backgrounds and occluding lines when evaluating individual features, as the two features have been shown to be vulnerable under our GAN-based pre-processing model in Section 6.3. We use a *third-party* captcha generator [36] to generate captchas of different security feature settings. For each setting, we generate 220,000 synthetic captchas. We then train our CNN-based solver on 200,000 captchas and test it on the remaining 2,000 captchas. Note that we do not fine-tune the solver in this experiment because the test data are also synthetic captchas.

**Overlapping.** By decreasing the space between adjacent characters, overlapping is a widely for anti-segmentation [10]. For captcha images of  $150 \times 70$  pixels, when the overlapping area of adjacent characters are 4, 6, 8 and 10 pixels (as depicted in Figure 16a), the success rate of our solver is 65%, 50.1%, 42.6% and 25.1%, respectively. The success rate is still significantly higher than the 1% threshold at which captchas are considered to be ineffective. It is worth mentioning that prior study has shown that if the resulting overlapping area is greater than 6 pixels, the resulting captcha will significantly affect user experience because it becomes difficult for humans to recognize characters from the image [8].

No.	Sample	Overlapping	Rotation	Distortion	Waving	Success Rate
1		✓	✓			74.85%
2		✓		✓		65.05%
3		✓			✓	58.8%
4			✓	✓		64.95%
5			✓		✓	82.35%
6				✓	✓	62.45%
7		✓	✓	✓		57.50%
8		✓	✓	✓	✓	52.50%
9		All security features				46.30%

**Table 5: Impacts of multiple combined security features.**

**Rotation.** In this experiment, we apply our solver to captchas where the characters are rotated clockwise/anti-clockwise with an angle of 15, 30, 45 and 60 degrees. Figure 16b illustrates some of the rotated captchas generated by our synthesizer. Our solver correctly recognizes all (100%) the captchas when characters are rotated at a 15- or 30-degree angle. It only fails to recognize 3 (99.85%) and 9 (99.55%) out of 2,000 captchas when characters are rotated at a 45- and a 60-degree angles respectively. Our solver fails to recognize some captchas because some of the characters of these captchas are largely overlapping with each due to rotation. We note that the rotation angle used by most of the current captcha schemes is under 30 degrees, because a greater rotation angle may have a negative impact on user experience. The results indicate that rotation alone does not enhance the security of text captchas under our attack.

**Distortion.** Character distortion can confuse bot programs as two different characters could look similar when they are distorted. For example, “O” and “0” are visually similar when they are distorted. Figure 16c gives some of the synthetic, distorted captchas that are used to test our solver. For this set of testing captchas, our solver correctly labels 92.9% of the captchas. This experiment suggests that distortion alone is not strong enough to defeat our attack.

**Waving.** Figure 16d shows some of the testing captchas with various waving degrees. Our solver is able to successfully label 98.85% of the captchas, outperforming the 93.6% success rate presented in [36]. Our solver only fails on 23 captchas which contain characters that are similar after waving, such as “O” and “0”, and “1” and “l”. For some of those failed captchas, our annotators also did not reach a consensus.

**Combining security features.** Table 5 shows how the combination of security features affect the accuracy of our solver. Combining multiple security features does improve the robustness of a captcha scheme. This can be seen from the drop in the solver’s success rate

when using two or more security features together. Specifically, character overlapping and distortion are more effective compared to rotation and waving, because overlapping and distortion can result in significant alterations to the shape of a character. This observation is also confirmed by the relatively lower success rate presented in rows No. 1 to No. 4 (where overlapping or distortion is used) compared with the 82.35% success rate at row No. 5 when rotation and waving (but not overlapping nor distortion) are used. Moreover, while using more security features results in a stronger captcha scheme, it reduces the usability of captchas. For example, our annotators struggle to recognize the captcha presented in row No. 8 in the first attempt. If we now consider Row No. 9, an example with all the security features considered in this work (including background noise and occluding lines), all our annotators consider this captcha to be user unfriendly and fail to correctly recognize it in the first attempt. For all the cases presented in Table 5, our solver success rate is above 46%, which is still greater than the 1% threshold when a captcha scheme is considered to be ineffective. Therefore, the results of this experiment suggest that balancing the security strength and usability of a text captchas under our attack is non-trivial.

## 6.8 Captcha Usability Study

Captcha is designed to be easy for humans to recognize while hard for bots. However, balancing the security strength and user experience is becoming increasingly difficult. In this experiment, we perform user study to quantify the impact of security features on user experience (i.e., captcha usability) and the success rate of our solver. To do so, we have conducted an online survey by recruiting 20 participants to fill in an anonymous questionnaire. Our participants are at the age group of under 30s and are familiar with text captchas. In the questionnaire, we present 100 synthetic captchas with different security strengthes. We give each participant one minute to label a captcha. We divide the captchas into six categories based on the number of characters and the security parameters used for generating the captcha. In the user study, we ask each participant to rate the usability of five captchas from each category on a 5-point Likert-scale, where 1 = very poor and 5 = excellent usability.

Table 6 gives the criteria used to determine the captcha difficulties and an example captcha image for each category. For each captcha category, we also give the averaged success rates achieved by our participants and our solver, as well as the averaged rating given by the participants.

While using more security features increases the difficulty for a computer program to solve a captcha challenge, doing so also makes it harder for a user to recognize the content of the captcha. For example, the averaged human success rate for the captchas in category 6 of Table 6 is below 70%, meaning that nearly one-third of the time a user will enter a wrong answer for captchas in this category. Therefore, captchas in this category were given the lowest usability score of 2.1 is not surprising. Also, as we expected, humans in general are better than computers at solving captchas, and the success rate of a computer solver drops as the difficulty of the captcha increases.

No.	Example	Security Features		Success Rate		Usability
		Anti-segmentation	Anti-recognition	Humans	Our approach	
1		English letters and arabic numerals	Rotation, varied font sizes	95.25%	100%	4
2		English letters	Rotation, varied font sizes	90.25%	88%	2.75
3		English letters, complex background	Rotation, distortion	91%	96%	2.8
4		English letters, overlapping characters, complex background	Varied font sizes, rotation, distortion	89.25%	86%	2.7
5		English letters	Varied font sizes, rotation, distortion	79.75%	77%	2.8
6		English letters, overlapping characters	Varied font sizes, rotation, distortion, waving	68.75%	40%	2.1

Table 6: Example captchas used in our user study, the success rates of humans and our approach, and the usability rating.

If we now consider categories 3 and 4 in Table 6 where background confusion is used, we find that noisy backgrounds have a negative impact on the user experience because our participants gave an averaged usability score of less than 3 for captchas in these categories. On the other hand, background confusion has little contribution to the security strength of captchas under our attack. This can be confirmed from the similar, or even better solving performance given by our solver when compared to human participants for captchas in the two categories. This finding suggests that complex background confusion perhaps should be abandoned in future text captcha schemes.

Overall, this user study shows that a deep-learning-based captcha solver can achieve comparable performance for solving text captchas when compared to humans, but balancing the security and usability of a text captcha scheme is non-trivial.

## 7 DISCUSSIONS

In this section, we discuss the limitations of our approach and the potential countermeasures for our attack.

### 7.1 Limitations

Naturally there is room for further work and possible improvements. We discuss a few points here.

**Captchas with variable numbers of characters.** Our current implementation targets text-based captchas with a fixed number of characters, but it can be extended to captchas with a variable numbers of characters. This can be achieved by first predicting how many characters are in the captcha and then selecting a model specifically trained for that number of characters. Our preliminary results show that we can learn a CNN model to predict the number of characters of an input captcha image with an accuracy of 89% and 70% for Wikipedia and Google schemes respectively.

**Multi-word captchas.** Our approach can be easily extended to multi-word captchas too. This can be done by either treating all words as a sequence of characters, or first segmenting the words and then recognizing individual words.

**Extend to other captcha schemes.** Our approach is generally applicable and can be naturally extended for video and image captchas by adapting the network architecture to recognize objects from the inputs; and favorably, the process of synthetic data generation, model training and tuning still is unchanged. This flexibility allows one to attack various types of captchas, not just text-based ones. For example, to target NuCAPTCHA [2], a motion-based captcha scheme, we need to replace our CNN solver with a model similar to the Mask R-CNN [26]. The idea is to first segment the video frames into images and then recognize characters from individual images. After replacing the solver structure, we also need to extend our GAN-based captcha synthesizer to generate a sequence of synthetic images (as recognition is performed at the image level). For motion-based captchas, the key is to maintain the temporal relationships among images, for which a temporal CNN can be useful [37].

### 7.2 Countermeasures

Recent studies have shown that adversarial examples generated by GANs can confuse machine learning classifiers [57]. By inserting some imperceptible perturbation on captcha images, one can mislead a machine learning model [47, 57] and at the same time the small perturbation does not interfere with a successful recognition of the image contents by humans. However, the perturbation to be put on the captcha image is tightly coupled to not only the captcha image itself, but also the captcha solver and its parameters. To generate effective adversarial examples requires having a way to observe the solver behavior. Doing so is difficult in practice because an adversary is unlikely to release the solver, while a small change in the solver structure (e.g., by changing the number and types of some neural network layers) is often sufficiently enough to invalidate the adversarial mechanism. This work shows that it is possible to quickly learn a highly accurate captcha solver using a small set of real captchas. This means the structure of the solver can be quickly changed to invalidate an adversarial mechanism used by a captcha scheme. While our work does not necessarily pronounce a death sentence to text-based captchas – as they are keeping evolving, we hope the high success rate achieved by our deep-learning-based

attack can encourage the community to carefully think about the implications of this widely used security mechanism.

Numerous alternative schemes have been proposed to replace text-based captchas. These include video-based captchas such as NuCAPTCHA [2] and game-based CAPTCHAs [43]. The former was shown to be vulnerable [7, 62]. The latter seemingly offers some promises but the recent breakthrough of deep reinforcement learning in game playing may pose a threat to such schemes [42]. To develop a robust countermeasure for deep-learning-based attacks, one probably needs to combine multiple mechanisms similar to the multi-factor authentication protocol [33, 51]. Nonetheless, how to balance the security strength and usability of a scheme is an outstanding problem.

## 8 RELATED WORK

Text-based captchas are a dominant captcha scheme used by many websites. There is an extensive body of prior work investigating ways to improve the security of text-based captchas, building upon attacks on existing schemes. However, text captchas are going through an iterative development process, just like cryptography and digital watermarking, where the previously successful attacks have led to the development of more secure schemes. While there are alternative captcha schemes available, text captchas are still preferred by many users due to familiarity and a sense of security and control [35].

Mori *et al.* were among the first attempts to break text captchas [25]. Their attack employs a set of analytical models and heuristics to break Gimpy and EZ-Gimpy, two early simple text-based captcha schemes. Yan *et al.* show a simple character segmentation method [63], which counts the number of pixels of individual characters, can break most of the captchas from Captchaservices.org. Later, they show an improved segmentation method can be used to attack the early captcha schemes used by Yahoo!, Microsoft and Google [64]. The work presented by Gao *et al.* targets captchas of hollow characters [15]. Their approach first fills the hollow character strokes, and then searches for the possible combinations of adjacent character strokes to recognize individual characters. While effective on hollow characters, this approach is ineffective on captcha images with overlapping and distorted characters. Unlike our approach, all the aforementioned attacks are tightly coupled to the captcha scheme and hard to generalize. This means to target a new captcha scheme, they would require human involvement to revise the existing heuristics and possibly to design new heuristics.

Decaptcha [8] employs machine-learning-based classifiers to develop a generic attack for text-based captchas. It is able to break 13 captcha schemes but achieves zero success on more difficult schemes including reCAPTCHA and Google's own scheme. By contrast, our approach not only gives a higher accuracy on the schemes where Decaptcha succeeds, but also delivers a success rate of 87.4% on reCAPTCHA for which Decaptcha has a success rate of zero (see Table 4). A more recent work presented by Gao *et al.* [17] uses the Log-Gabor filter, a classical signal processing algorithm, to first extract character components from the captcha image; it then uses the k-Nearest Neighbor algorithm to recognize individual characters using the extracted information. Due to the limitation of the Log-Gabor filter, their approach is ineffective for captcha

images with noisy backgrounds. For example, their approach fails to recognize the Baidu captcha shown in Figure 1a. Recently, George *et al.* presents a hierarchical model called the Recursive Cortical Network (RCN) for image recognition [19]. The RCN is effective in recognizing individual characters but are less effective for solving text-based captchas when compared to our approach. Our approach outperforms all the three captcha schemes showcased in the RCN work (on the same testing dataset from the RCN paper) [19]. In particular, on the PayPal dataset, our approach boosts the success rate from 57.1% to 92.4%. Stark *et al.* [55] show that active learning can be used to reduce the number of captchas required to learn a solver. However, this approach requires having access to a captcha generator of the target scheme, which is often not available to the adversary. On the other hand, active learning is complementary to our approach as it allows the learning engine to use a fewer number of training samples to speed up the training process [45, 46]. Compared to these prior generic attacks, our approach is by far the most effective generic attack – it delivers a higher success rate and can successfully attack some current captcha schemes where others failed.

It is worth mentioning that there are also other captcha schemes built around images [1, 3, 13, 24, 44] or audio data [6, 50]. Many of these were proposed to replace text-based captchas. Unfortunately, these alternative schemes are less popular than text captchas and were shown to be vulnerable too [9, 18, 40, 43, 54, 58]. In particular, a significant weakness of an image-based scheme is that the number of images used by the scheme is typically limited. As a result, an adversary may exploit side channels to obtain and label a large portion of the images used by a scheme [29].

As a final remark, we would like to point out that our work builds upon the foundations of adversarial machine learning [22, 30]. This technique is shown to be useful in constructing adversarial applications to bypass malware detection [49, 61], escape from spam mail filtering [5], or confuse machine learning classifiers [23, 41]. However, no work to date has employed the technique to construct a generic solver for text captchas and our work is the first to do so.

## 9 CONCLUSION

This paper has presented the first generative-adversarial-network based solver for text-based captchas. Our solver is automatically learned from training examples and hence can target a wide range of schemes. As a departure from prior machine-learning-based attacks, our approach requires significantly fewer real captchas to construct the solver. We achieve this by first learning a captcha synthesizer to automatically generate synthetic training examples to build a base solver, and then refining the base solver by applying transfer learning to a small set of real captchas. The key advantage of our attack is that it needs less human involvement when targeting a new captcha scheme. This means that our attack can be easily adjusted to catch up with the ever-changing captcha scheme.

Our approach was evaluated on 33 text captcha schemes, including 11 schemes that were being used by 32 of the top-50 popular websites at the time the work was conducted. Our approach outperforms four prior state-of-the-arts by successfully solving more captchas. We show that our approach is robust and generally applicable, which can break many advanced security features used

by modern text captchas. Our results suggest that these advanced features only make it difficult for a legitimate user but would fail to stop automated programs. Given that deep learning and generative adversarial based approaches are making great progress in solving image-related tasks, the insights provided in this work can help security experts to revisit the design and usability of text captchas.

## ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under grant agreements 61672427, 61672428, 61772422 and 61872294; the China Scholarship Council (201806970007); the Science and Technology Innovation Team Support Program of Shaanxi Province, China (2018TD-O26); the UK Engineering and Physical Sciences Research Council (EPSRC) through grant agreements EP/M01567X/1 (SANDeRs) and EP/M015793/1 (DIVIDEND); and the Royal Society International Collaboration Grant (IE161012). We thank to JD Cloud for the use of their computing servers.

## REFERENCES

- [1] Are you a human. <https://www.oneyouarehuman.com/>.
- [2] Nucaptcha. [www.nucaptcha.com/](http://www.nucaptcha.com/).
- [3] ATHANASOPOULOS, E., AND ANTONATOS, S. Enhanced captchas: using animation to tell humans and computers apart. In *IFIP International Conference on Communications and Multimedia Security* (2006), pp. 97–108.
- [4] AUDET, C., AND JR, J. E. D. Mesh adaptive direct search algorithms for constrained optimization. *Siam Journal on Optimization* 17, 1 (2006), 188–217.
- [5] BARRENO, M., NELSON, B., SEARS, R., JOSEPH, A. D., AND TYGAR, J. D. Can machine learning be secure? In *ACM Symposium on Information, Computer and Communications Security* (2006), pp. 16–25.
- [6] BIGHAM, J. P., AND CAVENDER, A. C. Evaluating existing audio captchas and an interface optimized for non-visual use. In *Sigchi Conference on Human Factors in Computing Systems* (2009), pp. 1829–1838.
- [7] BURSZTEIN, E. How we broke the nucaptcha video scheme and what we proposed to fix it. <https://elie.net/blog/security/how-we-broke-the-nucaptcha-video-scheme-and-what-we-propose-to-fix-it>.
- [8] BURSZTEIN, E., AIGRAIN, J., MOSCICKI, A., AND MITCHELL, J. C. The end is nigh: generic solving of text-based captchas. In *USENIX WOOT* (2014).
- [9] BURSZTEIN, E., AND BETHARD, S. Decaptcha: breaking 75% of ebay audio captchas. In *Usenix Conference on Offensive Technologies* (2009), pp. 8–8.
- [10] BURSZTEIN, E., MARTIN, M., AND MITCHELL, J. Text-based captcha strengths and weaknesses. In *CCS* (2011), pp. 125–138.
- [11] CHELLAPILLA, K., LARSON, K., SIMARD, P. Y., AND CZERWINSKI, M. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *Conference on Email & Anti-Spam* (2005).
- [12] CHOW, R., GOLLE, P., JAKOBSSON, M., WANG, L., AND WANG, X. Making captchas clickable. In *Proceedings of the 9th workshop on Mobile computing systems and applications* (2008), ACM, pp. 91–94.
- [13] ELSON, J., DOUCEUR, J. R., HOWELL, J., AND SAUL, J. 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.
- [14] ET AL., P. I. Pix2Pix: Image-to-image translation with conditional adversarial networks. <https://github.com/phillipi/pix2pix>.
- [15] GAO, H., TANG, M., LIU, Y., ZHANG, P., AND LIU, X. Research on the security of microsoft’s two-layer captcha. *IEEE Transactions on Information Forensics & Security* 12, 7 (2017), 1671–1685.
- [16] GAO, H., WEI, W., WANG, X., LIU, X., AND YAN, J. The robustness of hollow captchas. In *ACM Sigsac Conference on Computer & Communications Security* (2013), pp. 1075–1086.
- [17] GAO, H., YAN, J., CAO, F., ZHANG, Z., LEI, L., TANG, M., ZHANG, P., ZHOU, X., WANG, X., AND LI, J. A simple generic attack on text captchas. In *NDSS* (2016).
- [18] GAO, S. An evolutionary study of dynamic cognitive game captchas: Automated attacks and defenses. *Dissertations & Theses - Gradworks* (2014).
- [19] GEORGE, D., LEHRACH, W., KANSKY, K., LIĆZARO-GREDILLA, M., LAAN, C., MARTHI, B., LOU, X., MENG, Z., LIU, Y., AND WANG, H. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science* (2017), eaag2612.
- [20] GOLD, C., HOLUB, A., AND SOLLICH, P. Bayesian approach to feature selection and parameter tuning for support vector machine classifiers. *Neural Networks* 18, 5 (2005), 693–701.
- [21] GOODFELLOW, I. J., BULATOV, Y., IBARZ, J., ARNOUD, S., AND SHET, V. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *International Conference on Learning Representations (ICLR)* (2014).
- [22] GOODFELLOW, I. J., POUGETABADIE, J., MIRZA, M., XU, B., WARDEFARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial networks. *Advances in Neural Information Processing Systems* 3 (2014), 2672–2680.
- [23] GOODFELLOW, I. J., SHLENS, J., SZEGEDY, C., GOODFELLOW, I. J., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. In *ICML* (2015), pp. 1–10.
- [24] GOSSWEILER, R., KAMVAR, M., AND BALUJA, S. What’s up captcha?: a captcha based on image orientation. In *International Conference on World Wide Web, WWW 2009, Madrid, Spain, April* (2009), pp. 841–850.
- [25] GREG, M., AND MALIK, J. Recognizing objects in adversarial cultter: Breaking a visual captcha. In *IEEE Computer Society Conferene on Computer Vision and Pattern Recognition* (2003).
- [26] HE, K., GKIOXARI, G., DOLLÁR, P., AND GIRSHICK, R. Mask R-CNN. In *IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.
- [27] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. 770–778.
- [28] HECHT-NIELSEN, R. *Theory of the backpropagation neural network*. Harcourt Brace & Co., 1989.
- [29] HERNANDEZCASTRO, C. J., RIBAGORDA, A., AND SAEZ, Y. Side-channel attack on labeling captchas. *Computer Science* (2009).
- [30] HUANG, L., JOSEPH, A. D., NELSON, B., RUBINSTEIN, B. I. P., AND TYGAR, J. D. Adversarial machine learning. *IEEE Internet Computing* 15, 5 (2011), 4–6.
- [31] ISOLA, P., ZHU, J.-Y., ZHOU, T., AND EFROS, A. A. Image-to-image translation with conditional adversarial networks. *arxiv* (2016).
- [32] J, W. Strong captcha guidelines v1. 2.
- [33] JIANG, Z., ZHAO, J., LI, X.-Y., HAN, J., AND XI, W. Rejecting the attack: Source authentication for wi-fi management frames using csi information. In *IEEE INFOCOM* (2013), pp. 2544–2552.
- [34] KINGMA, D. P., AND BA, J. Adam: A method for stochastic optimization. *Computer Science* (2014).
- [35] KROL, K., PARKIN, S., AND SASSE, M. A. Better the devil you know: A user study of two captchas and a possible replacement technology. In *NDSS Workshop on Usable Security* (2016).
- [36] LE, T. A., BAYDIN, A. G., ZINKOV, R., AND WOOD, F. Using synthetic data to train neural networks is model-based reasoning. In *International Joint Conference on Neural Networks* (2017), pp. 3514–3521.
- [37] LEA, C., VIDAL, R., REITER, A., AND HAGER, G. D. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision* (2016), pp. 47–54.
- [38] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324.
- [39] LI, J., MONROE, W., SHI, T., JEAN, S., RITTER, A., AND JURAFSKY, D. Adversarial learning for neural dialogue generation.
- [40] MEUTZNER, H., AND KOLOSSA, D. Reducing the cost of breaking audio captchas by active and semi-supervised learning. In *International Conference on Machine Learning and Applications* (2014), pp. 67–73.
- [41] MIYATO, T., MAEDA, S., KOYAMA, M., NAKAE, K., AND ISHII, S. Distributional smoothing by virtual adversarial examples. *arXiv* (2015).
- [42] MNH, V., KAVUKCUOGLU, K., SILVER, D., GRAVES, A., ANTONOGLU, I., WIERSTRA, D., AND RIEDMILLER, M. Playing atari with deep reinforcement learning. *arXiv* (2013).
- [43] MOHAMED, M., SACHDEVA, N., GEORGESCU, M., GAO, S., SAXENA, N., ZHANG, C., KUMARAGURU, P., OORSCHOT, P. C. V., AND CHEN, W. B. 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.
- [44] MOHAMED, M., GAO, S., SACHDEVAC, N., SAXENA, N., ZHANG, C., KUMARAGURU, P., AND OORSCHOTE, P. C. V. On the security and usability of dynamic cognitive game captchas. *Journal of Computer Security* (2017), 1–26.
- [45] OGLIVIE, W. F., PETOUMENOS, P., WANG, Z., AND LEATHER, H. Fast automatic heuristic construction using active learning. In *International Workshop on Languages and Compilers for Parallel Computing* (2014), pp. 146–160.
- [46] OGLIVIE, W. F., PETOUMENOS, P., WANG, Z., AND LEATHER, H. Minimizing the cost of iterative compilation with active learning. In *Proceedings of the 2017 International Symposium on Code Generation and Optimization* (2017), CGO ’17, pp. 245–256.
- [47] OSADCHY, M., HERNANDEZ-CASTRO, J., GIBSON, S., DUNKELMAN, O., AND PŁĘREZ-CABO, D. No bot expects the deepcaptcha! introducing immutable adversarial examples, with applications to captcha generation. *IEEE Transactions on Information Forensics & Security* 14, 99 (2017), 1–1.
- [48] PAN, S. J., AND YANG, Q. A survey on transfer learning. *IEEE Transactions on Knowledge & Data Engineering* 22, 10 (2010), 1345–1359.
- [49] ROSENBERG, I., SHABTAI, A., ROKACH, L., AND ELOVICI, Y. Generic black-box end-to-end attack against rnns and other api calls based malware classifiers. *arXiv* (2017).



- [50] SCHLAIKJER, A. A dual-use speech captcha: Aiding visually impaired web users while providing transcriptions of audio streams. *LTI* (2010).
- [51] SHAHZAD, M., LIU, A. X., AND SAMUEL, A. Behavior based human authentication on touch screen devices using gestures and signatures. *IEEE Transactions on Mobile Computing* 16, 10 (2017), 2726–2741.
- [52] SHRIVASTAVA, A., PFISTER, T., TUZEL, O., SUSSKIND, J., WANG, W., AND WEBB, R. Learning from simulated and unsupervised images through adversarial training. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).
- [53] SIMONYAN, K., AND ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *Computer Science* (2014).
- [54] SIVAKORN, S., POLAKIS, I., AND KEROMYTIS, A. D. I am robot: (deep) learning to break semantic image captchas. In *IEEE European Symposium on Security and Privacy* (2016), pp. 388–403.
- [55] STARK, F., HAZIRBAS, C., TRIEBEL, R., AND CREMERS, D. Captcha recognition with active deep learning. In *German Conference on Pattern Recognition Workshop* (2015).
- [56] SZEGEDY, C., VANHOUCHE, V., IOFFE, S., SHLENS, J., AND WOJNA, Z. Rethinking the inception architecture for computer vision. *Computer Science* (2015), 2818–2826.
- [57] SZEGEDY, C., ZAREMBA, W., SUTSKEVER, I., BRUNA, J., ERHAN, D., GOODFELLOW, I., AND FERGUS, R. Intriguing properties of neural networks. *Computer Science* (2013).
- [58] TAM, J., SIMSA, J., HYDE, S., AND AHN, L. V. Breaking audio captchas. In *Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December* (2008), pp. 1625–1632.
- [59] VON AHN, L., BLUM, M., HOPPER, N. J., AND LANGFORD, J. *CAPTCHA: Using Hard AI Problems for Security*. Springer Berlin Heidelberg, 2003.
- [60] VON AHN, L., BLUM, M., AND LANGFORD, J. Telling humans and computers apart automatically. *Communications of the Acm* 47, 2 (2004), 56–60.
- [61] XU, W., QI, Y., AND EVANS, D. Automatically evading classifiers: A case study on pdf malware classifiers. In *Network and Distributed System Security Symposium* (2016).
- [62] XU, Y., REYNAGA, G., CHIASSON, S., FRAHM, J.-M., MONROSE, F., AND VAN OORSCHOT, P. C. Security analysis and related usability of motion-based captchas: Decoding codewords in motion. *IEEE transactions on dependable and secure computing* 11, 5 (2014), 480–493.
- [63] YAN, J., AND AHMAD, A. S. E. Breaking visual captchas with naive pattern recognition algorithms. In *Computer Security Applications Conference, 2007. ACSAC 2007. Twenty-Third Annual* (2007), pp. 279–291.
- [64] YAN, J., AND AHMAD, A. S. E. 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.
- [65] YOSINSKI, J., CLUNE, J., BENGIO, Y., AND LIPSON, H. How transferable are features in deep neural networks? In *Advances in neural information processing systems* (2014), pp. 3320–3328.
- [66] YU, L., ZHANG, W., WANG, J., AND YU, Y. Seqgan: Sequence generative adversarial nets with policy gradient.
- [67] ZHU, J.-Y., PARK, T., ISOLA, P., AND EFROS, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593* (2017).