

Received March 3, 2020, accepted March 18, 2020, date of publication March 24, 2020, date of current version April 7, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2982945

Simple and Easy: Transfer Learning-Based Attacks to Text CAPTCHA

PING WANG¹, HAICHANG GAO¹, (Member, IEEE), ZIYU SHI¹,
ZHONGNI YUAN¹, AND JIANGPING HU²

¹School of Computer Science and Technology, Xidian University, Xi'an 710071, China

²School of Cyber Engineering, Xidian University, Xi'an 710071, China

Corresponding author: Haichang Gao (hchgao@xidian.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61972306.

ABSTRACT CAPTCHA, or Completely Automated Public Turing Tests to Tell Computers and Humans Apart, is a common mechanism used to protect commercial accounts from malicious computer bots, and the most widely used scheme is text-based CAPTCHA. In recent years, newly emerged deep learning techniques have achieved high accuracy and speed in attacking text-based CAPTCHAs. However, most of the existing attacks have various disadvantages, the attack process made high complexity or manually collecting and labeling a large number of samples to train a deep learning recognition model is time-consuming and expensive. In this paper, we propose a transfer learning-based approach that greatly reduces the attack complexity and the cost of labeling samples, specifically, by pre-training the model with randomly generated samples and fine-tuning the pre-trained model with a small number of real-world samples. To evaluate our attack, we tested 25 online CAPTCHAs achieving success rates ranging from 36.3% to 96.9%. To further explore the effect of the training sample characteristics on the attack accuracy, we elaborately imitate some samples and apply a generative adversarial network to refine the samples, sequentially we use these two kinds of generated samples to pre-train the models, respectively. The experimental results demonstrate that the similarity between randomly generated samples and elaborately imitated samples has a negligible impact on the attack accuracy. Instead, transfer learning is the key factor; it reduces the cost of data preparation while preserving the model's attack accuracy.

INDEX TERMS CAPTCHA, security, deep learning, transfer learning.

I. INTRODUCTION

Since the text-based CAPTCHA scheme was first introduced, it has been widely used to distinguish malicious bots from humans [1]. A text-based CAPTCHA requires a user to decipher letters or Arabic numerals embedded in an image and then re-enter them to pass the test. This simple structure makes CAPTCHAs intuitive for users worldwide. As one of the most important tools for preventing computer attacks, the security aspect of text-based CAPTCHAs has attracted considerable attention from researchers and industry practitioners. To improve security, researchers have introduced numerous resistance mechanisms that enhance CAPTCHA complexity, including background interference, noise lines, and geometric alterations [8]. However, with the advent of

deep learning techniques, all these defense mechanisms have been overcome [2], [3], [6], [7], [9].

To attack text CAPTCHAs, most of the previous attacks have adopted a three-step strategy: preprocessing, segmentation and recognition. The preprocessing operation involves extra calculation cost and effort, and the segmentation-based methods do not perform well in terms of accuracy and efficiency. Moreover, deep learning techniques require a large number of training samples, but manually collecting and labeling large numbers of samples is labor-intensive and inefficient. Therefore, some researchers have attempted to train models using synthetic data [20], [29]. However, mimicking real-world CAPTCHA schemes is also complex, and time-consuming because of the requirement of background recovery, font matching, features adjustment, etc. Furthermore, the models trained on synthetic data are usually unable to achieve considerable accuracy on the real-world attacks. In 2018, Ye *et al.* [9] combined a generative adversarial

The associate editor coordinating the review of this manuscript and approving it for publication was Amit Singh¹.

network (GAN) [19] and transfer learning to address the limitations of current deep learning attacks. Their method was the first to adopt a GAN to break text-based CAPTCHAs. However, that work contains three main limitations. First, using SimGAN [10] to fine-tune the samples greatly increases the computational complexity. Second, it uses a complicated method to remove image noise, which is also time-consuming. Third (and most important), the impact of using a GAN on the attack success rate was not discussed. Briefly, the existing methods either have high complexity in the image-processing stage or requires a large effort on training the recognition engine.

In this paper, we propose a simple, generic and efficient transfer learning-based method that can greatly decrease the complexity in breaking text CAPTCHAs. Unlike previous works that require manually labeling a large number of real samples or elaborately imitating the real CAPTCHAs, our attack requires only randomly generated samples and a small set of real samples. We achieve this by using randomly generated synthetic images to pre-train the model and then using a few real labeled CAPTCHAs to fine-tune the model. The recognition engine is a combination of a residual network (ResNet) [12], a recurrent neural network (RNN), and an attention mechanism.

To evaluate the effectiveness of the attack, 20 Roman-character-based CAPTCHAs and 5 Chinese CAPTCHAs found on popular websites, such as Google, Apple, Wikipedia, Baidu, Alipay, and Sina, were tested. With transfer learning, only 500 (1,000 for the Chinese scheme) real samples from each scheme are used to fine-tune the base-solver model. The proposed attack achieves high success rates—ranging from 36.3% to 96.9%. The experimental result shows that our approach reduces the crack complexity while remaining high accuracy, and it also demonstrates that transfer learning is a generic and efficient strategy for breaking text-based CAPTCHAs.

To further study the effect of the similarity between synthetic data and real-world samples on the attack accuracy, we used carefully simulated samples and GAN-based refined samples to pre-train the model. Seven targeted schemes were chosen as representatives. The results show that the level of similarity between the synthetic and real samples has only a small impact on attack accuracy and that using a GAN to imitate real samples has little influence on improving the attack success rate.

The remainder of this paper is organized as follows. In Section II, we survey prior attacks and introduce background information about transfer learning. Section III includes not only our overall cracking idea and the details of our attack scheme but also reports the attack results using randomly generated samples. In Section IV, we instead use elaborately imitated samples and GAN-based generated samples to evaluate our attack. We also conducted a further study on the similarity between the synthetic data and the real-world samples. In section V, the results of a comprehensive comparison between our attack and prior works and the analysis

are provided. Then, we experimentally discuss the effects of some CNN models and how to optimize the training speed; also, we further analyze the impact of using different training strategies. Finally, we summarize our work in Section VI.

II. BACKGROUND

A. PRIOR ATTACKS

In large numbers of commercial websites, a CAPTCHA scheme is the most prevalent method used for protection against malicious bots. However, many of these text schemes have been successfully broken by the attacks of prior works, which has spurred the development of CAPTCHA designs. Although many other CAPTCHA schemes have been proposed, such as image-based schemes, audio-based schemes, and game-based schemes, text-based CAPTCHAs are still the most widely used type because of their intuitiveness and low deployment costs.

The early text-based CAPTCHAs were simple and differed only in character size and rotation angle; consequently, most of them were easily recognized using optical character recognition (OCR) engines [13]. Later, designers added more security features to text-based CAPTCHAs; however, the advances in traditional machine learning techniques quickly broke these schemes as well. In 2003, Mori *et al.* applied the shape context matching-based method to break two early simple schemes named Gimpy and EZ-Gimpy, achieving success rates of 33% and 92%, respectively [14]. In 2006, using pattern recognition algorithms, Yan *et al.* successfully broke most of the CAPTCHAs at CAPTCHA-service.org [16]. In 2008, Yan's work applied a machine learning mechanism to recognize segmented CAPTCHA characters [4]. Using a K-Nearest Neighbors (KNN) model, Gao's team proposed a generic attack in 2016 that achieved success rates of between 5% and 77% on Google reCAPTCHA, Yahoo! and Microsoft designs [18]. Undeniably, traditional machine learning algorithms are able to successfully attack some simple schemes. However, most of the traditional attack methods are very slow. The numerous image processing details they required also made them difficult to implement. In addition, they are unsuitable for attacking difficult schemes that feature complex resistance mechanisms.

As deep learning techniques developed and achieved great results in image classification tasks, neural networks (NNs) were also adopted in the CAPTCHA field; however, training an efficient neural network requires a certain number of labeled samples. Thus, some works used manually labeled collected CAPTCHAs to train their models. Using LeNet for recognition, Chellapilla attempted two-step attacks (segmentation and recognition) to study the security of some typical CAPTCHAs [15]. For each scheme, they labeled 1,800 CAPTCHAs for training. In 2013, using their novel segmentation algorithms and a convolutional neural network (CNN), a model by Gao *et al.* was able to attack a family of hollow CAPTCHA schemes [7]. However, their segmentation process was highly complex, and they used a manually labeled sample set to train the CNN. In 2017,

George *et al.* [32] proposed a recursive cortical network (RCN)-based approach to attack four CAPTCHA schemes. However, their attack required clean individual characters, which also increased the difficulty. In 2018, [3] proposed a deep-learning-based three-step attack. To obtain an efficient model, they used 2,400 manually labeled samples from each scheme for training. Most recently in 2019, Zi *et al.* [24] proposed an end-to-end attack to solve text CAPTCHAs with a deep learning network; but they trained the recognition models with 10,000 manually labeled CAPTCHAs for each scheme. In particular, to obtain higher accuracy in Google reCAPTCHA, they even used 200,000 manually labeled samples for training the model, which took a lot of time and labor costs.

Using real-world labeled samples to train the model is the most effective way, but training deep-learning networks requires a large number of labeled samples that are sometimes difficult to collect, and manual labeling is expensive and time-consuming. Moreover, when the CAPTCHAs expand their character classes, for example, by adopting large-alphabet languages such as Korean and Chinese, even more samples are required to train the model, which increases the labeling cost. Due to the difficulty of collecting and manually labeling real CAPTCHAs, it has become a trend to use synthetic samples to break text-based CAPTCHA. In 2015, Stark *et al.* [33] introduced a simple method to recognize CAPTCHAs using synthetic training data. However, their training process required that new samples be continuously added to the training dataset, which is complicated and inefficient. Besides, their approach applied to fixed-length CAPTCHAs only. In 2017, Gao's team proposed another deep-learning-based method that used an imitator to generate CAPTCHAs for attacking Microsoft's two-layer hollow scheme [2]. Later, Le *et al.* [20] used synthetic data to train the recognition model and achieved good success rates on their synthetic data, ranging from 91.05% to 99.8%, but their model performed poorly on the real-world CAPTCHAs. While using synthetic data indeed reduces the cost of manually labeling samples, the traditional image generation algorithms used to imitate real-world samples also introduce extra time and labor costs. Therefore, attackers must create diverse samples with different character fonts, sizes, rotation angles, backgrounds, and so on to make their synthetic samples as similar to real-world ones as possible.

To solve the sample simulation problem, in 2018, [9] first utilized SimGAN [10] to implement a CAPTCHA synthesizer that could generate synthetic CAPTCHA automatically. They also used a specific GAN named Pix2Pix [28] to train a preprocessing model to remove security features and standardize the font style. In conjunction with transfer learning, their attack achieved good success rates, ranging from 3% to 100% on 33 text-based CAPTCHAs. However, their work involves training two individual GANs, which greatly increases the time and computational cost. Adjusting the parameters for the two GANs to closely imitate real samples

also requires extra labor and time. Last but not least, they did not report the effects of using the GANs in their work. Thus, it is still unclear whether using a GAN to generate CAPTCHA samples can indeed improve the attack results.

In summary, regardless of which types of resistance mechanisms are used in text CAPTCHAs, some approaches exist that can attack them successfully. However, the prior attacks have various drawbacks with regard to attack efficiency and cost.

- Image preprocessing and segmentation are complex operations that have a great impact on the subsequent recognition step.
- Manually labeling samples is expensive and time-consuming. (especially for large-alphabet schemes)
- Generating synthetic samples requires careful adjustments that involve additional time and labor costs. Moreover, training GANs induces yet more costs, and the effects of using a GAN are still unclear.

Thus, an approach is expected to solve these defects while achieving low-cost and efficient attacks, and this is the novelty and motivation of our work.

B. TRANSFER LEARNING

In many machine learning and deep learning tasks, it is assumed that the training and test data should exist in the same feature space and have a similar distribution. However, in many real-world cases, that assumption cannot be satisfied because collecting sufficient training data for machine learning or deep learning models may be difficult (labeling may be expensive). Under this circumstance, researchers seek to find substitute samples that are similar to the source data to train their models and complete their tasks. In this way, the knowledge learned from the substitute data can be applied to the source data, which underpins the concept of transfer learning.

The fundamental motivation for transfer learning was first discussed in 1995 with a focus on "learning to learn",¹ and it subsequently attracted more attention in different tasks such as classification, object localization, and clustering problems. It is a promising strategy for the tasks that lack sufficient samples. For example, suppose we want to classify cats of different breeds, but we only have sufficient training data from ImageNet, which contains some cats. We could train a base model on the ImageNet dataset and later retrain the model using a few detailed cat data to obtain a more accurate model. In transfer learning applications, we define the source data that are sufficient and related to the target as the *source domain* and the task when training the source domain as the *source task*. Naturally, the other important definition in such applications is the *target domain* and *target task*, which respectively indicate the domain-specific data used for training the model to achieve the final goal. In this definition, transfer learning uses a dual learning approach that

¹http://socrates.acadiau.ca/courses/comp/dsilver/NIPS95_LTL/transfer.workshop.1995.html.

TABLE 1. A survey of text-based scheme targeted by our work.

Scheme	Sample image	String length	Security features	Scheme	Sample image	String length	Security features
Google		8~10	Distortion, overlapping, varied fonts	Microsoft		4~6	Hollow character, diagonal distribution
Wiki		8~10	Distortion	Apple		4~5	Background, overlapping
Baidu_1		4	Noise lines, rotation	Baidu_2		4	Rotation
Baidu_3		4	Hollow character, varied fonts	Alipay		4	Overlapping, distortion
QQ		4	Varied fonts, rotation	Bilibili		5	Distortion, noise lines, rotation
Weibo		4	Distortion	Sina		5	Noise lines, varied fonts, rotation
Csdn		5	Color background	JD_1		4	Color Background
JD_2		4	Color Background	JD_3		4	Distortion
Sohu		4	Noise lines, rotation	Douban		5~8	Color background, distortion
360_1		4~5	Noise lines, varied fonts	360_2		4~5	Color background, rotation
Baidu		2	Overlapping, varied fonts, rotation	Renmin		2	Rotation, color background
Dajie		4	Overlapping, rotation, noise lines, color background	Douban		3~5	Complex background, rotation, distortion
It168		4	Rotation, noise lines				

first attempts to learn the knowledge from the source domain and then to retrain the model for detailed parameters in the target domain to finally complete the target task [11].

III. PROPOSED ATTACK

In this section, we introduce our attack based on simple synthetic data and transfer learning. To evaluate the effectiveness of our attack, we tested our approach on 25 real-world text CAPTCHAs with randomly generated samples, including both Roman-character-based schemes and Chinese schemes.

A. DATA PREPARATION

To test the feasibility of the proposed attack, we select 25 targeted CAPTCHAs, all of which are deployed by highly popular websites as ranked by Alexa. As shown in Table 1, each scheme has distinct security features, and our targeted schemes cover almost all the resistance mechanisms employed in text CAPTCHAs.

1) ROMAN-CHARACTER-BASED SCHEMES

Roman-character-based schemes are the most widely used type of CAPTCHA because they are universal for worldwide use. In this study, we selected 20 Roman-character-based schemes deployed by famous websites worldwide, including Google, Microsoft, Wikipedia, Apple, Baidu and so on. For each scheme, we collected 1,500 real CAPTCHAs from the websites. Note that only 500 of them are used for fine-tuning, and another 1,000 are applied to calculate the test accuracy, which is not required in the real attack. Each scheme uses a different character dictionary; we found a total of 62 characters, including 52 English letters and 10 digits.

2) CHINESE SCHEMES

To expand the solution space of text CAPTCHAs, many schemes have utilized large-alphabet languages for CAPTCHA designs. For example, Chinese has over 3,000 commonly used characters, many more than the Roman alphabet’s 62 characters. Moreover, most Chinese characters consist of integrated Chinese radicals, which also makes them more difficult to recognize than Roman characters [5]. In fact, some prior works have analyzed the significance of and worked partly with Chinese schemes [3], [5], [22]. To investigate the potential of our attack, we also evaluated our transfer-learning-based attack on five Chinese CAPTCHA schemes deployed by five well-known Chinese commercial websites. For each scheme, we collected 2,000 samples in total, half of which were used for fine-tuning, and the other half for testing. Each CAPTCHA image was labeled manually. In the Chinese schemes, the total number of character classes was 3,626.

B. ATTACK DETAILS

As Fig. 1 shows, our attack consists of three parts:

Step 1. CAPTCHA generation. This step uses image-processing algorithms to generate CAPTCHAs unrelated to the targeted scheme for training our recognition network. In our attack, all the pre-training samples are generated completely randomly without any special design, which is easy to implement and greatly reduces the effort taken in gathering training samples.

Step 2. Pre-training. After generation, the synthetic CAPTCHAs are input directly into the recognition engine without any preprocessing to train a base model. After the pre-training, we adopted the trained model as the base model of all the subsequent schemes.

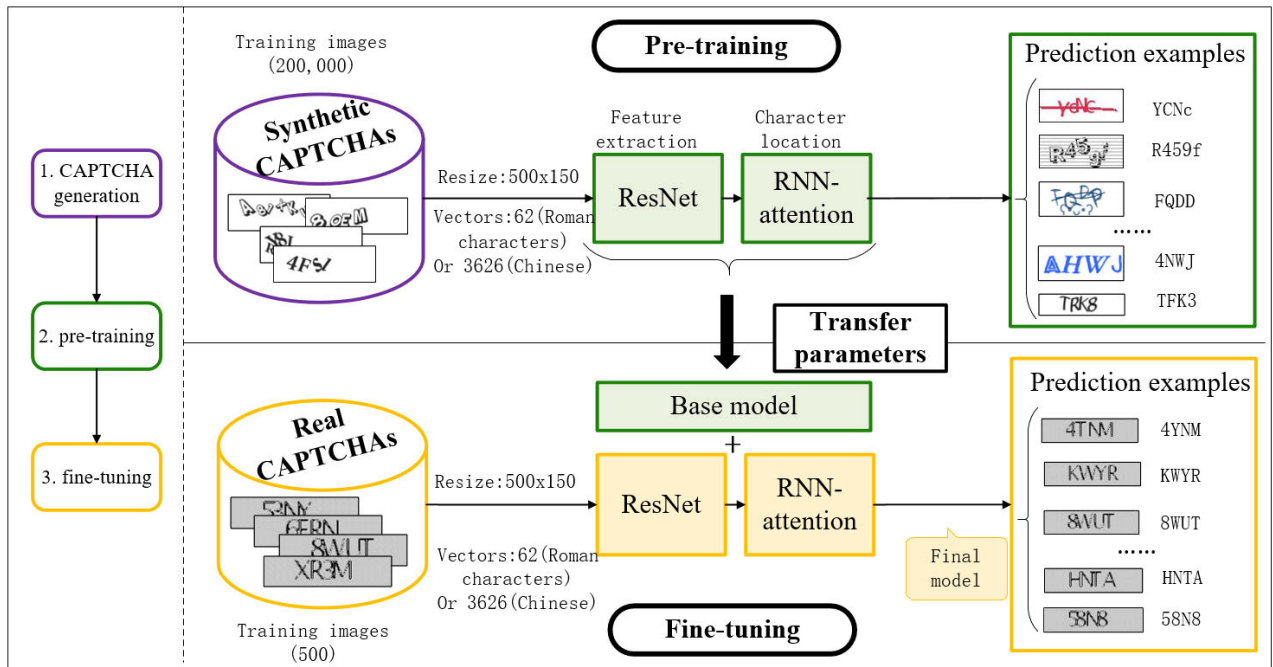


FIGURE 1. The overview of our attack approach.

Step 3. Fine-tuning. Finally, for each scheme, we used 500 real samples to fine-tune the base model. This stage was achieved by retraining the base model using transfer learning for the purpose of updating the parameters corresponding to the real features. Note that we only used domain adaptation of transfer learning, and the model retained consistency in the pre-training and retraining stages.

In the following, we explain the details of our attack.

1) CAPTCHA GENERATION

To reduce the costs associated with manual labeling, in this study, we also generated synthetic CAPTCHAs as pre-training data. All of the training data for the base model is generated by simple image processing algorithms.

As Fig. 2 shows, all the pre-training samples are generated with black characters on a pure white background. Unlike the original CAPTCHAs, we did not add any security features to the generated CAPTCHAs, e.g., no noise lines, distortion, overlapping, and so on. Instead, the samples were generated in the simplest manner to reduce the generation cost since this type of CAPTCHA is easy to implement and requires no special effort. Note that our generated CAPTCHAs are completely unrelated to targeted CAPTCHAs, and they are not similar to any of the targeted schemes.

For the Roman-character-based schemes, the length of the text string is set to between 4 and 10; the fonts are randomly selected from the font library, including both regular and hollow styles; all the images are the same size and the text rotation angle is set from minus 45 to 45 degrees. We generate 200,000 images for base model pre-training. For the Chinese schemes, we set the string length between 2 and 5;

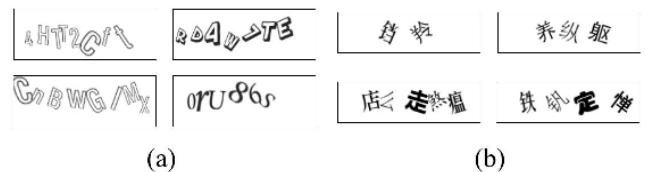


FIGURE 2. Some examples of our randomly generated CAPTCHAs for training the base model: (a) Randomly generated Roman-character-based CAPTCHAs; (b) Randomly generated Chinese characters.

selected the most commonly used fonts in the font library; and randomly generated 500,000 images. All the samples were resized to 500×150 and can be generated within an hour in Python using an image processing framework named Pillow on a desktop with Intel Core i3 CPU.

2) PRE-TRAINING

To recognize the entire character string in one step, a combined model [23] consisting of a CNN and a long short-term memory (LSTM) model [27] was utilized as the recognition engine. The CNN is responsible for extracting the feature vector of the CAPTCHA image. For this study, we chose ResNet v2-101 [12] as our CNN, which is designed to solve the degradation problem that occurs as network depth increases. The performance of other CNN structures was also evaluated, such as Inception v3 [25] and SeNet [26]; however, ResNet performs better than those two candidates. The details are described in Section V-B.

The LSTM converts the feature vectors extracted by CNN into a single text string; it can be considered as a character-level language model. The main operations are

applying input and output gate units and memory cells to learn to open and close access to the constant error flow. Decisions are made using the last states in the memory cells. In this experiment, the number of LSTM cells depends on the maximum string length of the targeted CAPTCHA.

To make the model “location-aware”, we also utilized a spatial attention mechanism following [23]. The attention mechanism allows our model to tackle text strings of different lengths without segmentation. Spatial attention models are typically used for OCR prediction based on the RNN state, as in [30]. They combine image content and spatial information to determine where the network should look. In this work, we utilized the same attention model as in [23] to enable the network to locate the characters automatically.

We trained the model by maximizing $\sum_{t=1}^T \log P(y_t | y_{1:t-1}, x)$, where x is the input image, y_t is the predicted label for location t , and T ranges from 2 to 10 (indicating the length of the string). The model is optimized by a stochastic gradient descent (SGD) strategy with an initial learning rate of 0.004, weight decay of 0.00004 and momentum of 0.9. We finally obtained one base model for Roman-character-based schemes with 20 epochs and one for Chinese schemes with 30 epochs.

3) FINE-TUNING

In the last step, we use transfer learning to fine-tune the pre-trained model parameters with few real CAPTCHAs. To further illustrate the fine-tuning process, we first provide mathematical justifications for how the mechanism of transfer learning works. In transfer learning [11], domain D is denoted as $D = \{X, P(X)\}$, where X is the feature space and $P(X)$ is a marginal probability distribution. For a specific domain, a task can be defined as $T = \{Y, f\}$, where Y denotes the label space and f denotes the objective predictive function. In general, a complete transfer learning process involves one source domain (D_S) and one target domain (D_T), which correspond to one source task (T_S) and one target task (T_T), respectively. From the knowledge in D_S and T_S , transfer learning aims to help improve the learning of the target predictive function f in D_T . In our CAPTCHA solver, f denotes the predictive function in ResNet, and D_S and D_T are as follows:

$$D_S = \left\{ (x_{S_1}, y_{S_1}), \dots, (x_{S_{n_S}}, y_{S_{n_S}}) \right\} \quad (1)$$

$$D_T = \left\{ (x_{T_1}, y_{T_1}), \dots, (x_{T_{n_T}}, y_{T_{n_T}}) \right\} \quad (2)$$

Regarding the training data, $x_{S_i} \in \mathcal{X}_S$ is the synthetic CAPTCHA and $y_{T_i} \in \mathcal{Y}_T$ is the corresponding CAPTCHA label, a character string. Here, x_{T_i} and y_{T_i} have the same meanings as in the real CAPTCHAs. Note that all the labels remain the same in D_S and D_T (62 or 3,626 characters), but the feature spaces are different because the features of the synthetic and real CAPTCHAs have different details.

To fine-tune the model, for each scheme, we restored the pre-trained model and all the layers were fixed unchanged before updating the prediction model. The architecture of the

TABLE 2. Attack results.

Scheme	Success rate		Speed(s)
	Pre-trained model	Fine-tuned model	
Alipay	3.6%	96.9%	0.025
JD_2	0.0%	96.8%	0.020
Csdn	0.0%	96.4%	0.020
JD_3	1.2%	96.2%	0.004
Baidu_2	1.2%	96.0%	0.025
Sohu	0.9%	95.5%	0.020
Douban	0.0%	94.8%	0.020
JD_1	0.0%	94.6%	0.021
Baidu_1	0.0%	93.5%	0.020
Bilibili	0.0%	93.2%	0.020
Baidu_3	0.0%	92.9%	0.026
Sina	0.2%	91.6%	0.026
Weibo	0.0%	91.4%	0.020
Apple	0.0%	81.1%	0.026
QQ	0.0%	80.5%	0.026
360_1	0.0%	79.9%	0.021
Wiki	0.0%	79.5%	0.020
Microsoft	0.7%	67.3%	0.021
360_2	0.0%	66.6%	0.021
Google	0.8%	51.9%	0.020
Renmin (Chinese)	0.0%	95.7%	0.018
It168 (Chinese)	0.0%	95.1%	0.015
Douban (Chinese)	0.0%	80.1%	0.014
Baidu (Chinese)	0.4%	76.4%	0.014
Dajie (Chinese)	0.0%	36.3%	0.013

network was exactly the same as the pre-training stage. All of the parameters also remained the same and only the epoch for each scheme was turned to 1. For each Roman-character-based scheme, 500 manually labeled real samples were used. Considering that the Chinese CAPTCHAs feature a larger character set than do the Roman CAPTCHAs, we adopted 1,000 real-world manually labeled Chinese CAPTCHAs per Chinese scheme.

Both pre-training and fine-tuning were implemented on the called TensorFlow deep learning framework. The models were trained on a computer equipped with an Intel Core i7 CPU (1.80GHz), an NVIDIA GeForce GTX 1080 GPU, and 16 GB of RAM. Training one epoch takes nearly 15 minutes.

C. ATTACK RESULTS

We implemented our attack and tested it on all the targeted schemes. For each scheme, 1,000 real-world CAPTCHAs were tested using the pre-trained and fine-tuned models, respectively. We summarize the resulting success rates and test speeds in Table 2.

1) SUCCESS RATE

As shown in Table 2, before fine-tuning the base model with real CAPTCHAs, the success rates of the targeted schemes are extremely low on both Roman-character-based and Chinese schemes. Most of the tested schemes achieved success rates of 0%—only four were above 1%. This demonstrates that training using only the randomly generated data is not feasible. However, after using a small number of real CAPTCHAs to fine-tune the base models, our approach was able to attack all the targeted schemes with considerable

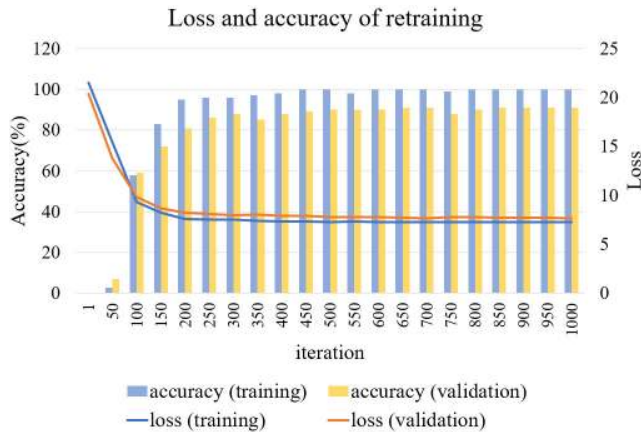


FIGURE 3. The loss (training VS validation) and accuracy (training VS validation) of retraining stage.

success rates. For more than half of the schemes, the attack achieved a success rate of over 90%. On reCAPTCHA, which has historically proven to be a difficult scheme [24], the fine-tuned model also achieved good results—51.9%. The lowest accuracy was achieved on the Dajie scheme of 36.3% (a Chinese scheme): this scheme is the most difficult because it has many security features and over 3,000 classes. However, the success rate on Dajie also satisfies the criteria commonly used in the CAPTCHA community—higher than 1% [31], which means that the attack was successful.

2) ATTACK SPEED

Finally, regarding runtime speed, our model requires approximately 0.02 seconds to test one CAPTCHA, which is completely acceptable because it satisfies the commonly used criterion in [31]. The results suggest that our approach executes in real time.

From the retraining process, we can argue that the transfer learning can help improve attack accuracy and accelerate the training stage. Taking Baidu_1 as an example, we recorded the loss and accuracy of training and validation, respectively, and showed them in Fig. 3. It is clear that when the number of training iterations is increased to 150, the loss is basically stable (1 epoch includes 1000 iterations).

To sum up, first of all, our attack is clearly extremely easy. The proposed approach requires neither preprocessing nor CAPTCHA segmentation. It achieves one-step recognition and is an end-to-end attack because when a new CAPTCHA comes, it can recognize all the characters directly. It is also suitable for solving variable-length strings. Furthermore, our method is very low cost in that we can attack any text-based CAPTCHA by using only a small number of real labeled samples, and mimicking those samples is not necessary; instead, we simply generate images with random fonts in a simple style. Moreover, our attack is undoubtedly efficient. The evaluation of the targeted 25 schemes all achieved considerable success rates at a very fast speed. In addition, this is the first work to use transfer learning to attack Chinese CAPTCHAs, and the results showed that our attack is also applicable to

large-alphabet CAPTCHAs. The results also demonstrate that transfer learning is a promising way of to enhance attack efficiency. In brief, this result not only helps make our attack extremely low-cost but also greatly reduces the complexity and effort for attacking text CAPTCHAs—and once again demonstrates the insecurity of text CAPTCHAs.

IV. THE EFFECT OF THE SIMILARITY OF SYNTHETIC SAMPLES

Based on the results in the last section, it is clear that even when pre-trained with randomly generated data, our attack achieved high success rates after transfer learning. However, most prior works that used synthetic data spent considerable effort to simulate real CAPTCHAs: they even applied deep-learning-based generation mechanisms to assist with sample generation. Therefore, in this section, we further analyze the impact of the similarity of pre-training data. To explore whether the similarity of the training samples affects the attack accuracy, we first use traditional (non-deep-learning) image generation algorithms to carefully simulate each scheme and generate corresponding samples. Then, we utilize a GAN mechanism to refine our imitated CAPTCHAs to make them more similar to real CAPTCHAs. Finally, we use the simulated data and the GAN-based refined data to train the base model.

A. MOTIVATION

To train an effective model, the ideal solution is to train the network with the original dataset. However, in reality, many factors affect the difficulty of obtaining sufficient quantities of labeled data. In such cases, we can simulate the original data to generate training samples. As is known, highly similar training samples will lead to good performance. However, generating synthetic samples that have high similarity requires substantial human effort and time. Therefore, Ye proposed a GAN-based attack in [9] for text CAPTCHAs, claiming that they could use SimGAN to refine synthetic CAPTCHAs, making them highly similar to real CAPTCHAs. However, they did not verify this claim through experiments.

In this paper, to verify whether training data similarity substantially influences the results of transfer-learning-based attacks, we also generated samples similar to real-world CAPTCHAs and employed SimGAN [10] to make the visual characteristics of our simulated samples more similar to real data. Then, we test the targeted schemes using base models trained with the simulated samples and fine-tuned with a few real CAPTCHAs.

Our experiments were intended to answer two main questions:

- Does the similarity between pre-trained data and original data have a large impact on transfer-learning-based attacks (including ours and Ye's)?
- Is it truly practical to use a GAN to assist with sample generation? Moreover, is it truly necessary to expend the effort to closely simulate real CAPTCHAs?

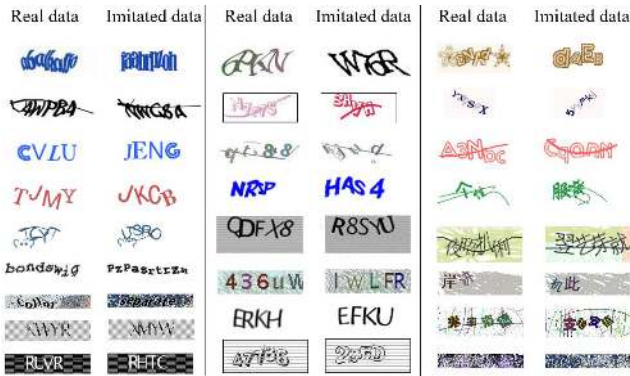


FIGURE 4. Examples of real and imitated targeted CAPTCHAs.

B. ATTACK WITH SIMULATED DATA

1) SIMULATING REAL CAPTCHAS

We first generated 25 types of CAPTCHAs by imitating real samples. For each targeted scheme, we observed the security features and tried our best to generate CAPTCHAs that were visually similar to the real CAPTCHAs. Our generation process is the same as in [24]. Fig. 4 shows some examples of the real data and our generated data for all the schemes.

For each Roman-character-based CAPTCHA scheme, we generated 10,000 imitation samples for training and another 1,000 for validation. Then, we mixed all the training and validation samples together to train one base model on 20 targeted schemes. For each Chinese scheme, we generated 100,000 imitated CAPTCHAs and then mixed them all to train a base model.

2) ATTACK RESULTS WITH SIMULATED DATA

In this experiment, the same 1,000 real-world CAPTCHAs were tested for each scheme as in Section III. Table 3 shows the results achieved by the pre-trained and fine-tuned models, respectively.

As Table 3 shows, when using our elaborately imitated samples without fine-tuning, we were able to successfully break most of the targeted schemes according to the criteria proposed by [31]. However, for some complex schemes such as Douban, QQ, and Wiki, the success rates were extremely low. For example, the success rate of our scheme on Google was still zero. However, after using a small number of real CAPTCHAs to refine the base model for each scheme, the success rates increased rapidly. Using the fine-tuned models, our attack achieved success rates of over 90% on more than half the targeted schemes.

C. ATTACK WITH GAN-BASED REFINED SAMPLES

1) USING SimGAN TO REFINE THE SIMULATED SAMPLES

In this experiment, we also used SimGAN to refine the synthetic samples. Fig. 5 shows an overview of the SimGAN-based generation method. As proposed in [10], SimGAN learns from simulated and unsupervised images through adversarial training, where the goal is to improve the realism of synthetic images. It uses an adversarial loss

TABLE 3. Attack success rates with simulation data.

Scheme	Pre-trained model	Fine-tuned model
Alipay	54.0%	97.5%
Sohu	91.5%	97.3%
JD_2	96.9%	97.1%
Baidu_2	71.2%	96.9%
Csdn	67.2%	96.7%
JD_3	92.6%	96.3%
JD_1	95.0%	95.1%
Douban	5.8%	94.5%
Bilibili	65.0%	93.6%
Weibo	70.6%	93.4%
Baidu_1	15.3%	93.4%
Baidu_3	83.5%	93.3%
Sina	66.9%	91.8%
360_1	32.3%	89.2%
Apple	43.8%	88.1%
Wiki	4.2%	86.0%
QQ	2.6%	82.7%
Microsoft	26.4%	77.3%
360_2	10.5%	76.2%
Google	0.0%	46.0%
It168 (Chinese)	60.6%	97.4%
Renmin (Chinese)	90.3%	96.7%
Douban (Chinese)	64.6%	92.5%
Baidu (Chinese)	29.0%	87.1%
Dajie (Chinese)	60.8%	67.7%

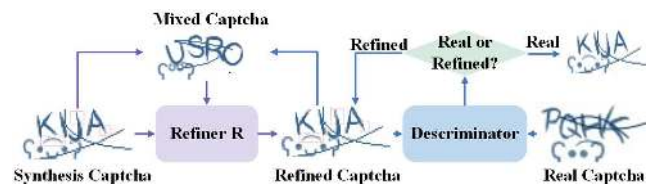


FIGURE 5. The training process of our GAN-based refining model.

to make the refined images indistinguishable from real ones with a neural network as discriminator. To preserve the annotations of synthetic images, the adversarial loss was complemented with a self-regularization loss to penalize large changes. Moreover, a fully convolutional neural network was used. It operates on a pixel level and preserves the global structure, thus the image content was not modified holistically. To obtain the most accurate synthetic data, we chose to terminate synthesizer training when the discriminator failure rate for distinguishing real from synthetic CAPTCHAs exceeds 95%—the same as in [9].

Five Roman-character-based schemes and two Chinese schemes from Table 1 were selected as representatives because they include different security features. For each scheme, we utilized the GAN to refine the synthetic CAPTCHAs generated in the last section. To further reveal the difference between samples refined by the GAN model with different numbers of iterations, the samples refined by SimGAN [10], which was trained in 2,000, 5,000, and 8,000 steps, respectively, were used to train the base model. Note that during this period, the base model of each scheme was trained separately rather than all together.



FIGURE 6. Changes in the sample refined by SimGAN.

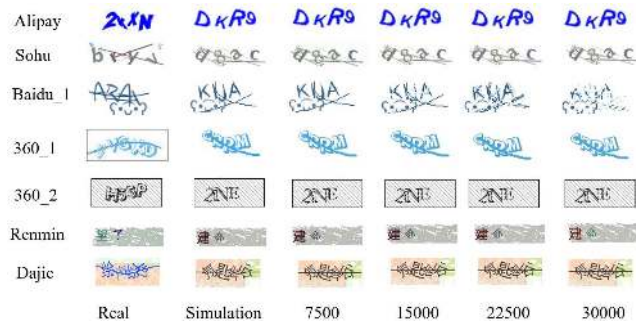


FIGURE 7. Fine-tuned CAPTCHAs by the refiner under different numbers of training iterations.

2) ATTACK RESULTS WITH REFINED DATA

Fig. 6 detailedly shows the changes in the images before and after refined. The GAN model changed the color of the generated CAPTCHA or refined the pixel details. This result demonstrates that using a GAN can indeed refine the tiny details of the pixels of generated CAPTCHAs to make them more accurate and increase their similarity to real samples.

Fig. 7 displays the refined results by the SimGAN under different numbers of training iterations. Obviously, for most schemes, the images vary little in visual characteristics. Before 10,000 training iterations, the generated CAPTCHAs change at the pixel level to become increasingly visually similar to real CAPTCHAs. However, after the number of training iterations increases beyond 10,000, the CAPTCHA samples become blurry. This result occurs because when the training iterations increase considerably, the model changes more pixels from the original image. For the Baidu scheme, the CAPTCHAs obviously change visually. Therefore, in our evaluation, we used only the data refined by GAN models trained with no more than 10,000 steps.

The attack results with the refined data are shown in Table 4. There are two groups of success rates. The first group was achieved by pre-training the base model with refined samples for different numbers of training steps. The other group of results were achieved by the fine-tuned models. The numbers in the second sub-row denote the training steps used in SimGAN to refine the training data. In summary, these success rates were achieved by a pre-trained model with refined CAPTCHAs and a fine-tuned model with refined CAPTCHAs. The fine-tuned model was trained on the same set of real CAPTCHAs as in the last section.

TABLE 4. Attack results achieved by GAN-based generated samples.

Scheme	Pre-trained model			Fine-tuned model		
	2,000	5,000	8,000	2,000	5,000	8,000
Alipay	24.1%	27.6%	27.1%	94.5%	96.3%	96.7%
Sohu	28.5%	27.2%	28.2%	92.5%	90.6%	93.8%
Baidu_1	8.2%	10.6%	8.0%	92.1%	91.7%	93.0%
360_1	15.2%	6.2%	6.8%	88.1%	87.3%	87.9%
360_2	25.7%	24.5%	26.2%	75.1%	74.8%	76.0%
Renmin (Chinese)	74.3%	84.6%	80.3%	90.1%	92.0%	89.9%
Dajie (Chinese)	45.1%	45.7%	32.5%	44.5%	62.3%	61.4%

As shown in Table 4, for all the schemes, when the training data are refined by SimGAN, the success rates do not obviously increase over those of the pre-trained models: the values are all similar to the success rates achieved by the pre-trained model evaluation with simulated data in Section IV-B. However, all the fine-tuned models achieved great increased success rates. We infer that the GAN did not help change the essential features of the generated samples, suggesting that even using GAN for refinement, there is also a gap between the synthetic data and the real data when they are used to train the models.

D. ANALYSIS

Based on the above experimental results, we can formulate exact answers to the questions raised in Section IV-A. We summarize and analyze our conclusions in the following section.

- First, the similarity between pre-trained data and original data has a very small impact on the transfer-learning-based attack. Based on our attack results, it is clear that whether using randomly generated samples or carefully simulated samples, after fine-tuning by a small set of real CAPTCHAs, the final model can achieve very high success rates on all the targeted schemes, including Chinese CAPTCHAs. We can definitively conclude that between data similarity and transfer learning, transfer learning is the true reason why the attack performance is enhanced rather than the use of a generation mechanism to optimize the generated samples.
- Second, it is unnecessary to use a GAN or, indeed, pay much effort to refine the synthetic samples. The reason can be explained in two aspects. From the time cost perspective, simulating the real CAPTCHAs is complex, and matching fonts or recovering backgrounds takes time to implement or requires labor-intensive processes to adjust the details. For example, in our experiments in Section IV-B, it takes approximately two days to imitate one scheme CAPTCHA. For the Google scheme, it almost takes five days to adjust the details of the image to make it more visually similar to the real sample. For the GAN-based mechanism, apart from the effort of rebuilding the model, training a GAN model takes 9.6 hours on average for one scheme, which is also extremely time-consuming. From the attack accuracy

perspective, either using the simulated samples or the GAN-based mechanism, regardless of the number of iterations the training model is allowed to run, the attack accuracy was not greatly enhanced. For both experiments, the attack accuracy was notably increased after the models refined by the real data, which again demonstrates the significance of transfer learning rather than generated samples' optimization.

In summary, these experimental results yield two significant conclusions. On one hand, the experimental results further demonstrate that using complex operations and processes to create highly similar samples makes no sense from an improvement viewpoint; thus, it is superfluous. In contrast, using transfer learning provides substantial benefits for training an efficient model. On the other hand, our results also demonstrate that text CAPTCHAs are no longer secure because no matter what kind of samples are used, the model, refined by a small set of real samples through transfer learning, can achieve high success rates for different text CAPTCHAs.

V. DISCUSSION

In this section, we first present a comprehensive comparison with prior works and further explain the superiority of our attack. Then, to further address the implementation details of our attack, we analyze the effects of different CNNs and the impacts of each training step to optimize the training stage. Finally, we summarize all the strategies that can be implemented using our approach and discuss the differences among them.

A. COMPARISON WITH PRIOR WORKS

Many prior works have studied CAPTCHA breaking, and some of our targeted schemes have also been studied by other excellent works. In this section, we choose five typical latest works that were accepted by the top conference or by journals in the security community and conduct a comprehensive comparison between our attack and the proposed attacks in those works. The selected works were published between 2014 and 2019. All of the attacks are compared in multiple dimensions, such as success rate, process complexity, and the required number of training data. All the success rates are shown in Table 5 and the attack processes are shown in Table 6.

From the attack accuracy perspective, for each scheme, our success rates are all higher than those in [9]. It is worth mentioning that our attack achieved a 51.9% success rate on Google reCAPTCHA, which is widely considered to be a difficult scheme, but the attack in [9] achieved only 3%. For the attack process, both their work and ours use transfer learning for attacking text CAPTCHAs, but their approach differs from ours in two respects. First, they used two GANs in their attack to pay effort to sample generation: SimGAN to refine the synthetic CAPTCHAs and Pix2Pix to pre-process the synthetics. In contrast, our approach uses only irrelevant

TABLE 5. Comparison between our approach and five prior works [3], [9], [17], [18], [24]. RI = Results by imitated data; RR = Results by randomly generated data.

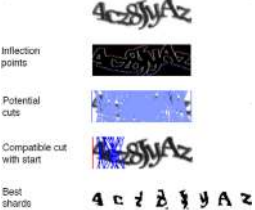
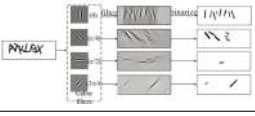
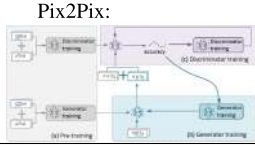
Scheme	Ours(RI/RR)	Bursztein [17]	Gao [18]	Tang [3]	Ye [9]	Zi [24]
Google	46.0%/51.9%	/	/	/	3.0%	42.7%
Microsoft	76.2%/67.3%	/	16.2%	50.9%	69.3%	74.8%
Wiki	86.0%/79.5%	28.3%	23.8%	90.0%	78.0%	90.6%
Apple	88.1%/81.1%	/	/	47.3%	/	82.4%
Baidu_1	93.4%/93.5%	/	/	/	34.0%	95.8%
Baidu_3	93.3%/92.5%	/	/	57.0%	/	96.0%
Alipay	97.5%/96.9%	/	/	/	61.0%	96.4%
Bilibili	93.6%/93.2%	51.1%	/	/	/	/
Weibo	93.4%/93.2%	/	/	51.2%	44.0%	86.2%
Sina	91.8%/91.6%	/	/	75.0%	52.6%	86.2%
JD_1	95.1%/94.6%	/	/	/	86.0%	96.4%
Sohu	97.3%/95.5%	/	/	/	92.0%	/
360_2	76.2%/66.6%	/	/	/	56.0%	84.3%
Baidu (Chinese)	87.1%/76.4%	/	/	28.6%	/	/

randomly generated CAPTCHAs. As discussed in section IV, an obvious flaw in using GANs is that it is laborious and time-consuming and does not pay significant role in accuracy enhancement. Second, their recognition network is based on LeNet-5 [21], which cannot recognize variable-length strings, but our attention-based network can solve variable-length strings. Overall, our approach achieves higher success rates, is much simpler, and requires a lower effort.

The success rates in [24] are similar to ours. Their approach also did not require pre-processing or segmentation and applied attention mechanism to tackle with variable-length strings. However, the labeled training-sample numbers in their method are much larger than ours. For each scheme, they used 10,000 manually labeled CAPTCHAs to train a proper model. For Google reCAPTCHA, they collected 200,000 real samples and labeled them to train a more efficient model. There is no doubt that using real samples is the most efficient strategy; but labeling such large number of CAPTCHA sample takes a lot of labor and time costs, which is inefficient from the cost perspective, and this approach does not work in the case where one does not have access to the real CAPTCHA system. In addition, they did not evaluate their attack on real-world large-alphabet text CAPTCHAs, e.g. Chinese schemes. They trained only synthetic Chinese samples and tested also synthetic CAPTCHAs, which showed only their deep-learning network capacity but could not illustrate their approach expandability on large-alphabet schemes.

The results of our approach compared to those of the other three works [3], [17], [18] are similar: most of their success rates are notably lower than ours when attacking the same scheme. The comparisons are acceptable because all these works use the image-processing based method to attack the targeted schemes. The essential traits of these methods are in three aspects. First, they are complex to implement. For example, in [18], five individual steps were used in the recognition, which makes much more parameters and details for the readers to perform. Second, the results of the first two steps

TABLE 6. The attack process comparison between our approach and five prior works [3], [9], [17], [18], [24].

Attack	Preprocessing	Segmentation	Recognition	Training data
[17]			Using KNN to score the segments; Using ensemble learning to arbitrate; Using Reinforcement learning to help;	Under 1,000 labeled CAPTCHAs
[18]			Component sorting: CFS; Graph building; Graph pruning; Recognizing component combinations: KNN; Graph search: dynamic programming (DP)	500 labeled CAPTCHAs for each scheme
[3]	Otsu's threshold method; Projection method; CFS;	CFS; CNN-based method;	LeNet-5	2000 labeled CAPTCHAs for each scheme (requiring preprocessing and segmentation)
[9]			LeNet-5	200,000 synthetic CAPTCHAs to train the base model (generated by image generator and refined by SimGAN); 500 labeled CAPTCHAs to fine-tune the base model
[24]			A combined architecture with CNN, LSTM, and attention model	10,000 labeled CAPTCHAs for each scheme
Ours			A combined architecture with CNN, LSTM, and attention model	100,000 synthetic CAPTCHAs to train the base model (randomly generated by image generator); 500 labeled CAPTCHAs to fine-tune the base model

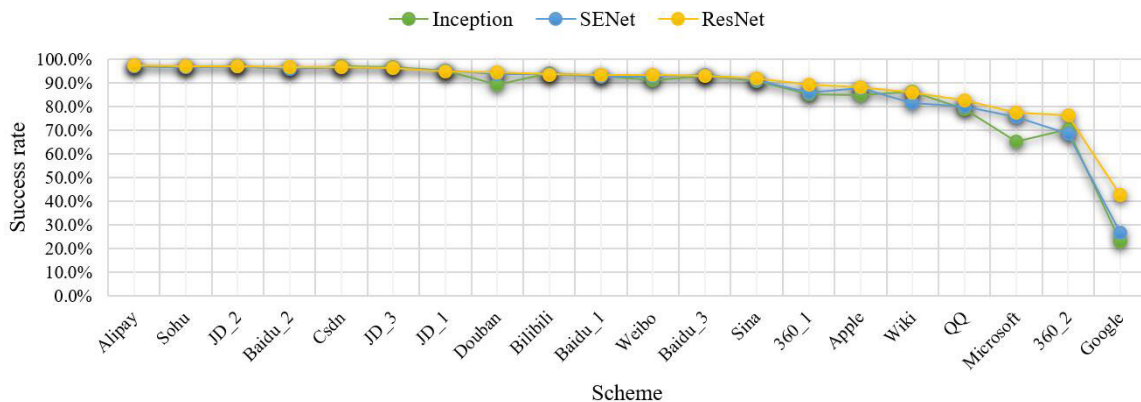


FIGURE 8. Attack results using different CNN models in our approach.

will have a significant impact on the last recognition step so that much effort is required to optimize the first two steps' results. In [17], to optimize the segmentation and recognition results, a large human effort is sacrificed to annotate segments that have been misclassified, which is a departure from the original intention of automated cracking. In contrast, our attack directly recognizes the entire CAPTCHA without any pre-processing or segmentation. Third, the image-processing based methods are always not generic for various CAPTCHAs. For instance, different schemes in [3] require

different algorithms and setting parameters to process the images. The success rate of Wiki in [3] is slightly higher than ours because they segment the characters individually, which makes them easier to recognize. Nevertheless, our approach is still both the simplest and most efficient.

In summary, our proposed approach is by far the simplest among the existing attacks. It further reduces the complexity of the attack procedures. From the training data perspective, it reduces the cost of labeling real samples or the efforts of imitating real samples. Furthermore, this approach is generic

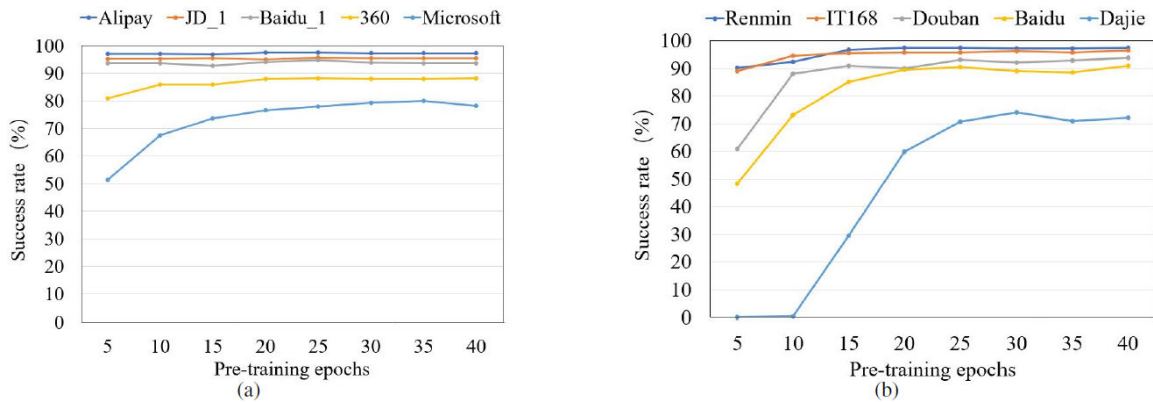


FIGURE 9. The success rates achieved with a base model trained with different numbers of epochs and fine-tuned with the same number of epochs: (a) Roman-character-based schemes; (b) Chinese schemes.

for different text CAPTCHAs, and it ensures comparable accuracy.

B. OTHER CNN MODEL ALTERNATIVES

In this section, we evaluate the performance of other typical CNN alternatives, including Inception v3 [25] and SENet [26], both of which perform excellently on object recognition and detection tasks. Note that we replace only the CNN component: the other modules in our recognition architecture remain the same. In addition, all the experimental configurations are the same as those reported in Section III.

We depict all the attack results of the three CNNs in Fig. 8. From the line chart, it is obvious that ResNet outperforms the other two CNNs on most targeted CAPTCHA schemes. For most of the schemes, the success rates are very close, except for Google. For the Google scheme, which is the most difficult of all the schemes, only ResNet achieves an accuracy greater than 50%, while the success rates of Inception and SENet are below 30%. We discuss why the ResNet performs better than other two models for Google reCAPTCHA. In this paper, we used ResNet-101 with 101 layers, the number of which is much larger than Inception v3 (47 layers) and SENet (47 layers (based on Inception v3)). It makes that ResNet can extract more advanced features. In addition, ResNet is constructed by residual mapping, as discussed in [12], which is easier to optimize than original, unreferenced mapping.

Therefore, we can make three conclusions. First, regardless of which of these three CNN models is used for extracting image features in our approach, the attention-based architecture can successfully break all the targeted schemes after fine-tuning the base model. Second, regardless of which model is used, the overall success rate trends between the different schemes are highly similar. Thus, different security features lead to different levels of difficulty for textual CAPTCHA. Finally, among the three networks, ResNet achieved the best performance.

C. TRAINING TIME OPTIMIZATION

To obtain a final appropriate model for CAPTCHA recognition, our CAPTCHA solver is trained in two stages:

pre-training and fine-tuning. This fact is significant when selecting the number of training epochs: more training epochs lead to more accurate performance but require more time. Therefore, in this experiment, we study how the training time affects model performance. Five representative Roman-character-based and Chinese schemes were selected for this experiment.

For the pre-training stage, the synthetic data were trained in 40 epochs. Fig. 9 shows the test success rates of the pre-trained models under different epochs. For the fine-tuning stage, we used the same pre-trained model and then retrained the model under a different number of epochs for different schemes. Because the fine-tuning stage, which learns detailed features based on the pre-trained model, is much faster, we trained the real CAPTCHAs using 6 epochs in total for each scheme. The test results are shown in Fig. 10.

In Fig. 9, for the Roman-character-based CAPTCHAs, the test accuracy increased substantially as the training epochs increased from 1 to 10, while the success rates became stable after the base-model training epochs exceeded 20. It is well known that, to some extent, more training epochs yield more accurate performance; however, more training epochs require more resources and time. Considering all 20 schemes, 20 epochs were determined for the base model because the time cost is acceptable, and the performance is comparable. Since the Chinese CAPTCHAs have a larger class space, training the base model for 30 epochs is more reasonable.

Fig. 10 shows the success rates when the same base model is fine-tuned for different epochs. The results intuitively show that, regardless of whether the target is a Roman or Chinese scheme, adding additional fine-tuning epochs does not significantly improve the success rate. The difference between the 1-epoch model and the 6-epoch model is small enough to ignore. Therefore, in our experiments, we chose 1 epoch for the fine-tuning stage. One epoch requires only approximately 15 minutes to complete on an NVIDIA GeForce GTX 1080 Desktop GPU, which saves time and is efficient.

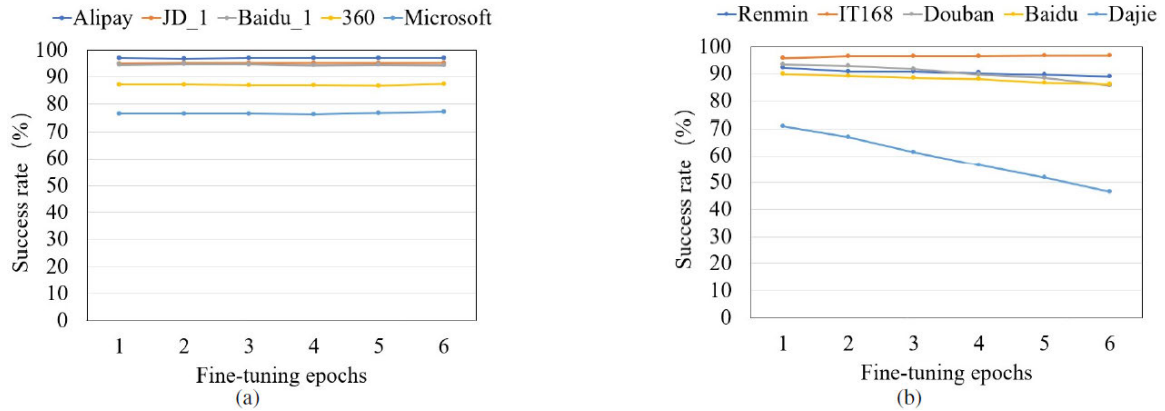


FIGURE 10. The achieved success rates with a fixed base model and a fine-tuned model using different numbers of epochs: Roman-character-based schemes; (b) Chinese schemes.

D. A DISCUSSION OF DIFFERENT TRAINING STRATEGIES

We have highlighted the performance of transfer learning and achieved considerable improvement by adopting it. However, what happens if transfer learning is not applied? In this section, we survey the results obtained by different training strategies and discuss the impact of the transfer learning mechanism.

To show the benefit of using transfer learning, we compare the results with and without transfer learning. There are seven different strategies in total. Without transfer learning, there are four groups of results obtained by training the synthetic CAPTCHAs alone, training the synthetic CAPTCHAs adjusted by GAN, training the randomly generated CAPTCHAs alone, and training the real CAPTCHAs individually. When using transfer learning, we used the first three groups of generated CAPTCHAs to train the base models and real CAPTCHAs to fine-tune the models. Note that the number of real CAPTCHAs for each scheme in the last four experiments remained the same (500 in the Roman-character-based schemes and 1,000 in the Chinese schemes).

For the Roman-character-based schemes, we selected Alipay, Sohu, Baidu_1, 360_1 and 360_2 as representative because they cover the range of results. For Chinese schemes, we selected Renmin and Dajie, for which the evaluation described in Section III achieved the best and worst results, respectively. All the experimental details were the same for all seven groups. We summarize all the success rates in Table 7, where $G1$, $G2$, $G3$ and $G4$ are the results achieved by models without transfer learning and represent the success rate achieved by the models trained with imitated samples, imitated samples refined by SimGAN, randomly generated samples, and a few real samples, respectively. The remaining three groups are the results achieved by fine-tuning the models with transfer learning. $G1/4$, $G2/4$, $G3/4$ denote the success rates achieved by using $G4$ to fine-tune models pre-trained by $G1$, $G2$, and $G3$, respectively.

Table 7 shows that among all the strategies, the results of the models without transfer learning are not as good as

TABLE 7. Attack results of different strategies.

Scheme	Without transfer learning				With transfer learning		
	$G1$	$G2$	$G3$	$G4$	$G1/4$	$G2/4$	$G3/4$
Alipay	54.0%	27.6%	3.6%	39.4%	97.5%	96.3%	96.9%
Sohu	91.5%	27.2%	0.9%	29.7%	97.3%	90.6%	95.5%
Baidu_1	10.3%	10.6%	0.0%	41.5%	93.4%	91.7%	93.5%
360_1	6.5%	6.2%	0.0%	0.0%	89.2%	87.3%	79.9%
360_2	10.5%	24.5%	0.0%	0.0%	76.2%	74.8%	66.6%
Renmin (Chinese)	78.8%	84.6%	0.0%	0.0%	96.3%	92.0%	95.7%
Dajie (Chinese)	55.7%	45.7%	0.0%	0.0%	67.7%	62.3%	36.3%

those with transfer learning. After applying transfer learning, the success rates on most of the schemes increased sharply due to the fine-tuning using real CAPTCHAs. This result illustrates that, without transfer learning, using either synthetic CAPTCHAs or a few real CAPTCHAs to train a recognition model is not sufficient.

For the three groups of experiments with transfer learning, although their results are very similar, the scenarios to which they were applied are not the same. Using imitation data to train the base model yields the highest success rates. However, simulating the CAPTCHAs is relatively complicated and time-consuming, and it is difficult to adjust the details. Therefore, this strategy is not suitable for tasks that require real-time cracking. Nevertheless, it could be used for tasks that require high precision because this method is the most accurate. Using GAN-based synthetics is also not suitable for real-time tasks: this strategy is more complicated and time-consuming because the GAN model must be trained. In comparison, using randomly generated samples combined with a small number of real CAPTCHAs for fine-tuning is the best choice. This is because the random generation is the simplest approach, has the lowest cost, and is generic for all schemes. Moreover, it is suitable for real-time tasks and achieves good results.

In summary, the results demonstrate that using transfer learning to combine the benefits of generating CAPTCHAs and labeling a small set of real CAPTCHAs is the

optimal choice. Moreover, applying transfer learning to CAPTCHA-breaking tasks enhances performance without increasing the time or labor costs.

VI. CONCLUSION

This paper systematically analyzed how to enhance the performance and reduce the complexity and costs of text-based CAPTCHAs attacks. We proposed a simple, low-cost and efficient method based on transfer-learning that uses randomly generated synthetic CAPTCHAs to train the base model and a very small set of real CAPTCHAs to fine-tune the base model. Our approach results in a high-performance CAPTCHA solver. Using deep learning techniques, we tested 20 Roman-character-based schemes and 5 Chinese schemes and achieved remarkably good success rates, ranging from 36.3% to 96.9% at an average attack speed 0.02 seconds per CAPTCHA. These results show that even on large-alphabet-based text CAPTCHAs, transfer learning can enhance the attack performance and reduce the collection and labeling costs.

To further understand how the similarity of training data affects the recognition results, we also carefully generated simulation samples and utilized SimGAN to refine the initial samples, making them more similar in detail to real CAPTCHAs. The experimental results demonstrate that under the transfer-learning attacks, the effort to similarity enhancement is not necessary. In addition, transfer learning plays a more crucial role in enhancing performance than refining samples to make them more similar. Finally, we compared our proposed attack with some typical existing attacks and further analyzed their respective characteristics. We also evaluated our method using some other classic CNNs on the targeted schemes. To study the efficiency of model training, we conducted further experiments to learn how to optimize the training time and to show the effects of different training strategies.

Our attack provides a more promising strategy that not only reduces the attack complexity and manual-labeling cost but also preserves comparable accuracy. We all know that text-based CAPTCHAs have security problems, and we hope and believe that our investigation will inspire other works. In addition, it also proves that using GAN-based sample adjustment is not the correct direction, conversely, attention should be paid to the fact that as one of the most popular techniques in generating pictures, a GAN will play a part in the CAPTCHA community in other ways. We plan to explore this idea in our ongoing future work.

REFERENCES

- [1] J. Yan and A. S. El Ahmad, "Usability of CAPTCHAs or usability issues in CAPTCHA design," in *Proc. ACM 4th Symp. Usable Privacy Secur.*, 2008, pp. 44–52.
- [2] H. Gao, M. Tang, Y. Liu, P. Zhang, and X. Liu, "Research on the security of microsoft's two-layer CAPTCHA," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 7, pp. 1671–1685, Mar. 2017.
- [3] M. Tang, H. Gao, Y. Zhang, Y. Liu, P. Zhang, and P. Wang, "Research on deep learning techniques in breaking text-based CAPTCHAs and designing image-based CAPTCHA," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 10, pp. 2522–2537, Oct. 2018.
- [4] J. Yan and A. S. El Ahmad, "A low-cost attack on a microsoft CAPTCHA," in *Proc. 15th ACM Conf. Comput. Commun. Secur. (CCS)*, 2008, pp. 543–554.
- [5] A. Algwil, D. Ciresan, B. Liu, and J. Yan, "A security analysis of automated Chinese turing tests," in *Proc. 32nd Annu. Conf. Comput. Secur. Appl. (ACSAC)*, 2016, pp. 520–532.
- [6] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. C. van Oorschot, "Security and usability challenges of moving-object CAPTCHAs: Decoding codewords in motion," in *Proc. USENIX Secur. Symp.*, 2012, pp. 49–64.
- [7] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow CAPTCHAs," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 1075–1086.
- [8] Y.-W. Chow, W. Susilo, and P. Thorncharoenri, "CAPTCHA design and security issues," *Advances in Cyber Security: Principles, Techniques, and Applications*. Singapore: Springer, 2019, pp. 69–92.
- [9] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, "Yet another text CAPTCHA solver: A generative adversarial network based approach," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2018, pp. 332–348.
- [10] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2107–2116.
- [11] S. Jialin Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2009.
- [12] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 630–645.
- [13] H. S. Baird and K. Popat, "Human interactive proofs and document image analysis," in *Proc. Int. Workshop Document Anal. Syst.* Berlin, Germany: Springer, 2002, pp. 507–518.
- [14] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA," in *Proc. Comput. Vis. Pattern Recognit.*, vol. 1, Jun. 2003, p. 1.
- [15] K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (hips)," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 265–272.
- [16] J. Yan and A. S. El Ahmad, "Breaking visual CAPTCHAs with Naive pattern recognition algorithms," in *Proc. 23rd Annu. Comput. Secur. Appl. Conf. (ACSAC)*, Dec. 2007, pp. 279–291.
- [17] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based CAPTCHAs," in *Proc. 8th USENIX Workshop Offensive Technol. (WOOT)*, 2014, pp. 1–15.
- [18] 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 *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, 2016, pp. 1–14.
- [19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
- [20] T. A. Le, A. G. Baydin, R. Zinkov, and F. Wood, "Using synthetic data to train neural networks is model-based reasoning," in *Proc. Int. Joint Conf. Neural New. (IJCNN)*, May 2017, pp. 3514–3521.
- [21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [22] B. Zhao, H. Weng, S. Ji, J. Chen, T. Wang, Q. He, and R. Beyah, "Towards evaluating the security of real-world deployed image CAPTCHAs," in *Proc. 11th ACM Workshop Artif. Intell. Secur. (AISec)*, 2018, pp. 85–96.
- [23] Z. Wojna, A. N. Gorban, D.-S. Lee, K. Murphy, Q. Yu, Y. Li, and J. Ibarz, "Attention-based extraction of structured information from street view imagery," in *Proc. 14th IAPR Int. Conf. Document Anal. Recognit. (ICDAR)*, vol. 1, Nov. 2017, pp. 844–850.
- [24] Y. Zi, H. Gao, Z. Cheng, and Y. Liu, "An end-to-end attack on text CAPTCHAs," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 753–766, Jul. 2019.
- [25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.
- [26] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
- [27] S. Hochreiter and J. Schmidhuber, *Long Short-Term Memory*, vol. 9, no. 8. Cambridge, MA, USA: MIT Press, 1997, pp. 1735–1780.

[28] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1125–1134.

[29] D. Lin, F. Lin, Y. Lv, F. Cai, and D. Cao, "Chinese character CAPTCHA recognition and performance estimation via deep neural network," *Neuro-computing*, vol. 288, pp. 11–19, May 2018.

[30] T. He, W. Huang, Y. Qiao, and J. Yao, "Text-attentional convolutional neural network for scene text detection," *IEEE Trans. Image Process.*, vol. 25, no. 6, pp. 2529–2541, Jun. 2016.

[31] E. Bursztein, M. Martin, and J. Mitchell, "Text-based CAPTCHA strengths and weaknesses," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, 2011, pp. 125–138.

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

[33] F. Stark, C. Hazrbas, R. Triebel, and D. Cremers, "CAPTCHA recognition with active deep learning," in *Proc. GPCR Workshop New Challenges Neural Comput.*, vol. 10, 2015, p. 94.



ZIYU SHI received the B.S. degree in software engineering from Xidian University, Xi'an, Shaanxi, China, in 2018, where he is currently pursuing the M.S. degree in computer technology. His current research interest is CAPTCHA.



ZHONGNI YUAN received the B.S. degree in software engineering from Xidian University, Xi'an, Shaanxi, China, in 2018, where she is currently pursuing the M.S. degree in computer science. Her current research interest is CAPTCHA.



PING WANG received the B.S. degree in software engineering from Xidian University, Xi'an, Shaanxi, China, in 2017, where she is currently pursuing the Ph.D. degree in software engineering. Her research interests include the security analysis of CAPTCHA and deep learning techniques.



HAICHANG GAO (Member, IEEE) is currently a Professor with Xidian University. He has published more than 30 articles. He is in charge of a project of the National Natural Science Foundation of China. His current research interests include CAPTCHA, computer security, and machine learning.



JIANGPING HU received the B.S. degree in information security from the China University of Mining and Technology, Xuzhou, Jiangsu, China, in 2018. He is currently pursuing the M.S. degree in cyberspace security with Xidian University, Xi'an, Shaanxi, China. His current research interest is Captcha.

...