SPC-GAN-Attack: Attacking Slide Puzzle CAPTCHAs by Human-Like Sliding Trajectories Based on Generative Adversarial Network

Rui Guo[1][0009−0005−1397−0983], Ke Yu[1][0009−0007−7074−9104], Enbo Yu[1][0009−0009−1686−6515], Qianqian Qiao[1][0009−0009−1200−5523], Zhenyu Mao[1][0009−0005−0127−5211], and Haizhou Wang[1][0000−0003−1197−5906] (✉)

School of Cyber Science and Engineering, Sichuan University, Chengdu 610065, China
{2021141530011,yuke039,enboy,2022141530055,2021141530031}@stu.scu.edu.cn, whzh.nc@scu.edu.cn

Abstract. CAPTCHAs are widely deployed as a Turing test to distinguish computers from humans. Slide puzzle CAPTCHAs are becoming more and more popular due to user-friendliness and diverse defense mechanisms. At present, there are few studies on the security of slide puzzle CAPTCHAs, and the defense mechanisms of slide puzzle CAPTCHAs studied in the past were relatively simple. The existing slide puzzle CAPTCHAs use rich notch shapes, add complex interferences to the background images, detect the characteristics of the sliding trajectories in the two-dimensional space. These advancements present challenges to existing attack methods. In order to test the security of the existing slide puzzle CAPTCHAs, six kinds of very popular slide puzzle CAPTCHAs including Tencent CAPTCHA, NetEase CAPTCHA, GeeTest CAPTCHA v3, GeeTest CAPTCHA v4, Shumei CAPTCHA and Dingxiang CAPTCHA are selected. Firstly, the Notch Detection Dataset and the Two-Dimensional Sliding Trajectory Dataset of each type of CAPTCHA are constructed for attacking slide puzzle CAPTCHAs. Secondly, **this paper is the first to apply generative adversarial network (GAN) to attack slide puzzle CAPTCHAs**. Finally, six kinds of CAPTCHAs are broken with the success rates close to 100%, which proves the effectiveness of this attack method. Experimental results show that the security of slide puzzle CAPTCHAs faces great challenges. The research work of this paper provides a novel idea for the security analysis of slide puzzle CAPTCHAs in the future.

Keywords: Slide puzzle CAPTCHA · Security · Notch detection · Trajectory generation · GAN.

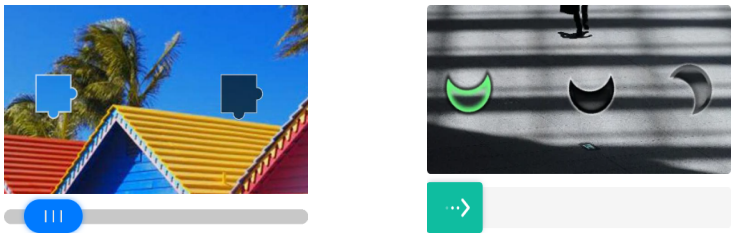
1 Introduction

CAPTCHA stands for Completely Automated Public Turing Test to Tell Computers and Humans Apart [1]. Humans can easily pass the CAPTCHAs, but computer programs are difficult to pass, so CAPTCHAs can be used to distinguish humans from computers. CAPTCHAs are widely used to protect websites

from attacks, such as malicious registrations, dictionary attacks, malicious comments and crawlers.

At present, the popular CAPTCHAs include text CAPTCHAs, image CAPTCHAs and Dynamic Cognitive Game (DCG) CAPTCHAs. Text CAPTCHAs used to be one of the most popular forms of CAPTCHAs, and require the user to identify a string of textual information. However, with the development of segmentation and character recognition technologies, most text CAPTCHAs have been successfully broken [2,3,14,12,13]. Image CAPTCHAs require the user to complete the image recognition or object recognition task. However, with the rapid development of computer vision technology, image CAPTCHAs have also been broken [15,16,17,4,7,24]. As CAPTCHA technology develops, DCG CAPTCHAs have become popular and been widely used because of their good user experience and security. DCG CAPTCHAs usually require user to complete the corresponding operations on the basis of target recognition. However, in recent years, studies have shown that the security of DCG CAPTCHAs also faces challenges [5,18,6].

As a typical type of DCG CAPTCHAs, slide puzzle CAPTCHAs have become more and more popular in recent years due to user-friendliness, diverse defense mechanisms and high security. Slide puzzle CAPTCHAs usually consist of three visual components: a background image containing a notch, a slider, and a button, as shown in Fig. 1. In the process of usage, the user needs to first click the button, then drags the slider to the notch position on the background image, and finally releases the mouse. In this process, the CAPTCHA security detection engine detects whether the slider reaches the notch position, whether the browser is controlled by script, whether the verification frequency exceeds the threshold, and whether the sliding trajectory is executed by human. Therefore, their security is higher than other CAPTCHAs [5,19,7,24].



(a) Without background interference. (b) Background interference.

Fig. 1. Samples of slide puzzle CAPTCHAs.

At present, there are few studies on the security of slide puzzle CAPTCHAs [5,19,7,24], and the defense mechanisms of slide puzzle CAPTCHAs analyzed in the past are relatively simple. The existing slide puzzle CAPTCHAs use rich notch shapes, add complex interferences to the background images, detect the characteristics of the sliding-trajectories in the two-dimensional space. Existing attack methods mainly face challenges in notch detection and trajectory generation. Firstly, the existing notch detection methods are difficult to

accurately locate the notch position, so that the slider cannot move to the correct notch position. Secondly, the existing methods of trajectory generation are limited to the horizontal single dimension, which cannot pass the trajectory detection. Finally, the existing sliding trajectories are generated based on fitted functions, which are relatively single and cannot well simulate the sliding characteristics of humans.

The main contributions of this paper are as follows:

(1).To the best of our knowledge, **we are the first to construct and publicly release Notch Detection Datasets (ND-Datasets) and Two-Dimensional Sliding Trajectory Datasets (2D-ST-Datasets)**[1] **for attacking slide puzzle CAPTCHAs**. For each type of slide puzzle CAPTCHA, we download 500 background images by a crawler, and manually label the notches in the background images to construct the ND-Dataset. For each type of slide puzzle CAPTCHA, we collect 500 verified human two-dimensional sliding trajectories to construct the 2D-ST-Dataset.

(2).To the best of our knowledge, **we are the first to apply generative adversarial network (GAN) to generate human-like sliding trajectories to attack sliding puzzle CAPTCHAs**. Although the 2D-ST-Datasets established in this paper is small in size, the sliding trajectories generated by GAN are already very close to the human sliding trajectories. Finally, six kinds of sliding puzzle CAPTCHAs are broken with the success rates close to 100%, which proves the effectiveness of this attack method.

2 Related Work

At present, the most popular CAPTCHAs include text CAPTCHAs, image CAPTCHAs and DCG CAPTCHAs. In this section, we review the security research on text CAPTCHAs, image CAPTCHAs, and DCG CAPTCHAs, and detail the security research on slide puzzle CAPTCHAs.

2.1 Text CAPTCHAs

As the earliest proposed CAPTCHA, text CAPTCHAs require the users to provide a correct sequence of characters in order according to the text image. Text CAPTCHAs are generally broken by segmentation and character recognition. Chellapilla et al. proved that computers are much better at character recognition than humans [25]. Therefore, designers focus on anti-segmentation schemes. The most commonly used anti-segmentation schemes include the hollow scheme, the two-layer scheme, crowding characters together (CCT), and background interference. But these resistance mechanisms have also been broken. Gao et al. have proven that the hollow scheme and two-layer scheme can be successfully broken [14,3]. Tang et al. proposed a pipeline method and broke a variety of CAPTCHAs with high success rates, thus proving that CCT scheme and background interference are also not secure [2]. Li et al. proposed an end-to-end attack

[1] https://github.com/CaptainTom2024/Slide-Puzzle-CAPTCHAs-Datasets.git

method based on Cycle-GAN, which significantly reduces the cost of data labeling [13]. Wang et al. also proved that text CAPTCHAs based on large character sets are also not secure [12].

2.2 Image CAPTCHAs

Image CAPTCHAs require the user to complete the image recognition or object recognition task. Compared with text CAPTCHAs, there are more types of image CAPTCHAs, and the images content are also richer, so the security is relatively higher. At present, there have been researchs on breaking image CAPTCHAs. For example, Asirra CAPTCHA, reCAPTCHA, Facebook CAPTCHA, and 12306 CAPTCHA based on image recognition were broken [16,17,7,24]. ARTiFACIAL CAPTCHA, FaceDCAPTCHA and FR-CAPTCHA based on face recognition were also broken [4,15].

2.3 DCG CAPTCHAs

In order to improve the security of CAPTCHAs and enhance interaction between humans and websites, researchers have invented DCG CAPTCHAs. DCG CAPTCHAs usually require the user to complete the corresponding operations on the basis of target recognition. For example, Drawing CAPTCHA, which asks the user to connect the three red diamonds with lines, was broken by an erosion-based algorithm [18]. Capy CAPTCHA requires the user to drag and drop the puzzle piece into the correct position, but it was broken by a side-channel attack [6]. Taobao CAPTCHA requires the user to drag a slider from the start to the end, but was broken by using Win32API (using an underlying API) to simulate mouse operations [5]. VAPTCHA CAPTCHA requires the user to draw a specific trajectory with the mouse. But the sliding trajectory can be determined by Faster R-CNN and LeNet-5, and the trajectory can be generated by fitted functions [5].

2.4 Slide puzzle CAPTCHAs

As the most popular type of DCG CAPTCHAs, slide puzzle CAPTCHAs are widely used due to user-friendliness and diverse defense mechanisms. In the process of usage, the user needs to first click the button, then drags the slider to the notch position on the background image, and finally releases the mouse. The existing methods of attacking the slide puzzle CAPTCHA mainly include two steps. Firstly, the notch position is recognized, and the sliding distance is determined. Secondly, the program generates sliding trajectory which simulates the characteristics of human sliding, and the slider is moved to the notch position according to the sliding trajectory [5,19,7,24].

Weng et al. recovered the original background images and detected the notch location by comparing between background image and its corresponding original background image [7,24]. Then, Sigmoid, Softmax, Relu and Tanh simulation functions were used to generate trajectories. Finally, they broke GeeTest

CAPTCHA, Tencent CAPTCHA and NetEase CAPTCHA with the success rate of 96%, 100% and 98%, respectively [7,24]. At present, the background images used in slide puzzle CAPTCHAs are diverse, making it difficult to recover all the background images. And when the background image is noisy, it is difficult to determine the correct notch position. At the same time, the sliding trajectories generated by these methods are limited to a single dimension and sliding speeds of displacements are random.

Wu et al. used Resnet-18 and Yolov3 for notch detection, achieving accuracies of 88% and 90%, respectively [19]. Then, they generated sliding trajectories by fitted curve. Finally, they broke Tencent CAPTCHA with the success rate of 75% [19]. However, to attack slide puzzle CAPTCHAs, higher notch detection accuracies are needed, otherwise the final attack results will be affected. At the same time, the sliding trajectories generated by this method are limited to a single dimension, and the training process is complex, and the success rate is low.

Chang et al. used Faster R-CNN to detect notches on GeeTest CAPTCHA, Tencent CAPTCHA and NetEase CAPTCHA, achieveing accuracy rates of all 100%. Then, they used acceleration formulas and log function to generate sliding trajectories. Finally, they broke GeeTest CAPTCHA, NetEase CAPTCHA, Tencent Security Center CAPTCHA and Tencent Online Demo CAPTCHA with the success rates of 87.5%, 91%, 88% and 100%, respectively [5]. However, there were deviations between the detected boxes and the actual notch edges, which led to the failure of some attacks. At the same time, the sliding trajectories generated by these methods are limited to a single dimension, and the generated sliding trajectories are relatively single.

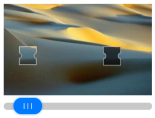
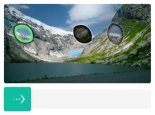
When attacking existing slide puzzle CAPTCHAs, existing attack methods mainly face challenges in notch detection and trajectory generation. Firstly, the existing notch detection methods are difficult to accurately locate the notch position, so that the slider cannot move to the correct notch position. Secondly, the existing researches on generating sliding trajectories are limit to the horizontal single dimension, which cannot pass the trajectory detection in the two-dimensional space. Finally, the existing sliding trajectories are generated by using fitted functions, which are relatively single and cannot well simulate the sliding characteristics of humans.

The object detection networks have achieved remarkable achievements and are widely used [8]. We try a variety of object detection networks to detect the notch position. According to the experimental results in Section 5.1, Yolov8 is more suitable for this task. GAN consists of a generator and a discriminator [23]. GANs are good at learning data distribution and have been applied to image processing, computer vision and sequential data [11]. We use human two-dimensional sliding trajectories to train GAN. After the adversarial training between the generator and the discriminator, the generator can well learn the distribution of human sliding trajectories and generate human-like sliding trajectories.

3 Analysis of the mainstream slide puzzle CAPTCHAs

In this paper, six kinds of most popular slide puzzle CAPTCHAs from five manufacturers including Tencent, NetEase, GeeTest, Shumei and Dingxiang are selected as experimental objects, as shown in Table 1.

Table 1. Characteristics of the mainstream slide puzzle CAPTCHAs (① Unfixed initial position of slider, ② Unfixed notch shape, ③ Noisy background image, ④ Limiting validation frequency, ⑤ Detecting controlled by script, ⑥ Detecting sliding trajectory).

CAPTCHA	Sample	①	②	③	④	⑤	⑥
Tencent		✓	✓	×	×	✓	✓
GeeTest v3		×	✓	✓	×	✓	✓
Shumei		✓	✓	✓	×	✓	✓
NetEase		×	×	×	✓	×	×
GeeTest v4		✓	✓	×	×	×	×
Dingxiang		✓	✓	✓	✓	×	×

Although in practice, the user only needs to drag the slider to the correct notch position. However, these six slide puzzle CAPTCHAs use different defense mechanisms. (1) The initial positions of the sliders for Tencent CAPTCHA, GeeTest CAPTCHA v4, Shumei CAPTCHA and Dingxiang CAPTCHA are not fixed. (2) Except for NetEase CAPTCHA, the other five CAPTCHAs use multiple notch shapes. (3) GeeTest CAPTCHA v3, Shumi CAPTCHA, and Dingxiang CAPTCHA add noises to the background images. (4) NetEase CAPTCHA and Dingxiang CAPTCHA limit the verification frequency, where high frequency verification will lead to the failure. (5) Tencent CAPTCHA, GeeTest CAPTCHA v3 and Shumei CAPTCHA would detect whether the browser is controlled by a script, which means that after using the script to control the browser to visit the

website, even if the human drags the slider to the notch position, the CAPTCHA will reject the verification. (6) Tencent CAPTCHA, GeeTest CAPTCHA v3 and Shumei CAPTCHA would detect sliding trajectories and non human-like sliding trajectories will lead to failure.

4 Attack Method

This paper proposes an attack method based on GAN to generate human-like two-dimensional sliding trajectories. The specific process is shown in Fig. 2.

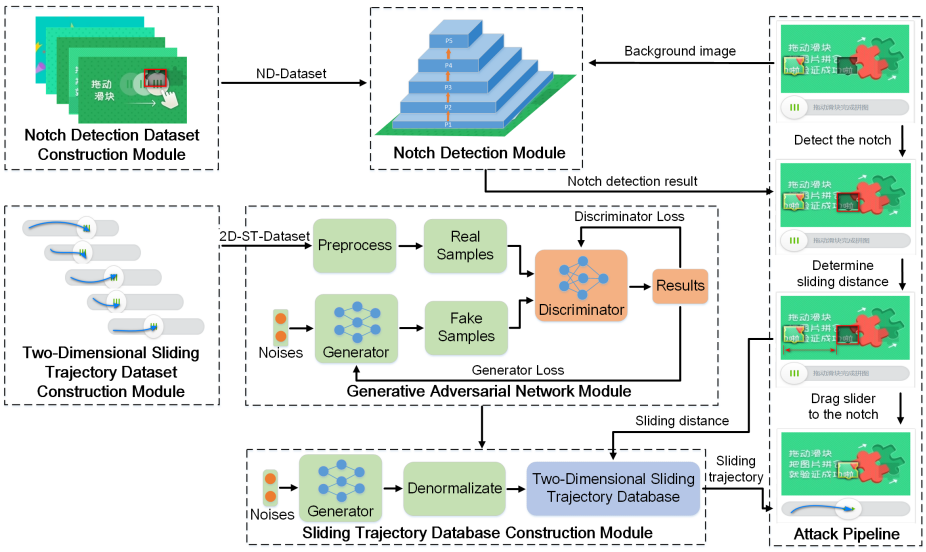


Fig. 2. Attacking slide puzzle CAPTCHA based on GAN.

4.1 Construction of Dataset

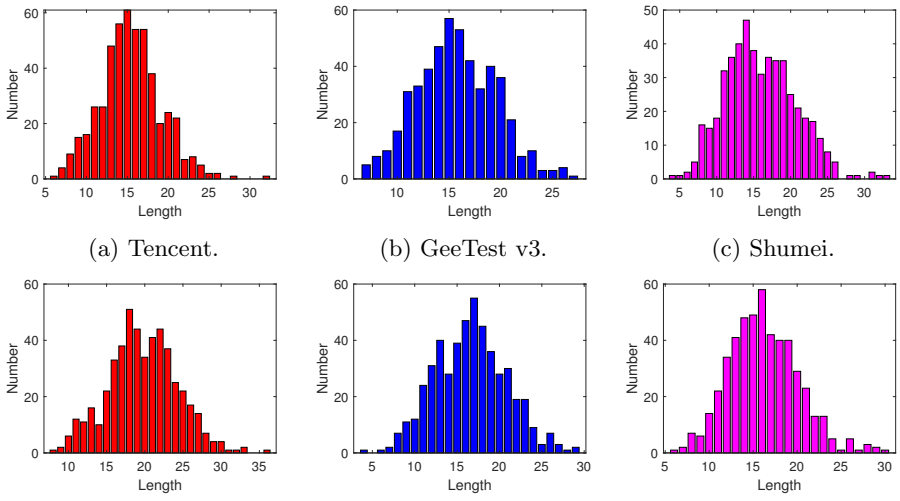
Construction of ND-Datasets In order to train detection model to accurately recognize notches in the background images, ND-Datasets need to be constructed. Firstly, we download 500 background images from each of the CAPTCHA website through a crawler, and divide them into a training set, a validation set, and a test set in the ratio of 6:2:2.

Secondly, we manually label the notches in the background images of the training and validation sets. The specific labeling processes are as follows: (1) There is no interference in the background images of Tencent CAPTCHA, NetEase CAPTCHA and GeeTest CAPTCHA v4, and only the notches need to be marked. (2) There are two notches in the background images of Shumei CAPTCHA, and it is impossible to distinguish which one is the correct notch only by observing the background image. Therefore, we label the both notches of Shumei CAPTCHA. (3) Although the GeeTest CAPTCHA v3 and Dingxiang

CAPTCHA also have interferences, the color of the interferences is significantly lighter than the color of the notches. So, only the notch positions need to be labeled.

Construction of 2D-ST-Datasets In this paper, the sliding trajectory refers to the movement path of the mouse when dragging button to move the slider to the notch position. GAN needs to learn the characteristics of human sliding trajectories to generate human-like attack trajectories. According to the experimental results in Section 5.2, the one-dimensional sliding trajectories only containing the horizontal direction are very difficult to pass the trajectory detection of CAPTCHAs, and the two-dimensional sliding trajectories can significantly improve the success rates. Therefore, we collect two-dimensional human sliding trajectories to construct 2D-ST-Datasets.

The specific method of collecting a human two-dimensional sliding trajectory is as follows: starting from the mouse pressed, record the horizontal and vertical displacements of the mouse in every time interval Δt, and a sub-trajectory $[\Delta x_i, \Delta y_i]$ is formed. To record the sliding distance of the trajectory completely, the horizontal and vertical displacements of the last period of time are recorded even if the time interval Δt is not reached when the mouse is released. Combining the sub-trajectories to form the trajectory $[[\Delta x_1, \Delta y_1], [\Delta x_2, \Delta y_2], \cdots, [\Delta x_k, \Delta y_k]]$ (k represents the length of the trajectory, which is the number of the last displacement, $\sum_1^k \Delta x_i$ denotes the sliding distance of the trajectory), we can obtain a two-dimensional sliding trajectory collected at time interval Δt.



(a) Tencent. (b) GeeTest v3. (c) Shumei.

(d) NetEase. (e) GeeTest v4. (f) Dingxiang.

Fig. 3. Distribution of human sliding trajectories' lengths.

According to the experimental results in Section 5.2, for CAPTCHAs with trajectory detection, the optimal collection time interval is $\Delta t = 0.05s$. When the collection time interval $\Delta t = 0.05s$, we collect 500 verified two-dimensional trajectories in each type of slide puzzle CAPTCHA. The distribution of trajectories' lengths is shown in Fig. 3.

The lengths of human sliding trajectories collected from the six kinds of CAPTCHAs generally meet the normal distribution. Except for NetEase CAPTCHA, the lengths of the other five CAPTCHAs mainly concentrate on 11-21, and the lengths of NetEase CAPTCHA concentrate on 15-25. Therefore, we construct 2D-ST-Datasets of five CAPTCHAs with the trajectories whose lengths are in the range of 11 to 21, and construct 2D-ST-Dataset of NetEase CAPTCHA with the trajectories whose lengths are in the range of 15 to 25.

4.2 Notch detection and trajectory generation

Notch detection In this paper, we use Yolov8 as the notch detection method, which provides a pre-trained model trained on the COCO dataset[2]. For each type of CAPTCHA, the corresponding notch detection model can be obtained after almost 50 rounds of training using the ND-Dataset. When attacking the slide puzzle CAPTCHA, the background image is input into the notch detection model, and the detection result for the notch position can be obtained.

After detecting the background image of the Shumei CAPTCHA, the model will output the position of the two notches. In conjunction with the height of the slider, the attack script can determine the correct notch position.

Trajectory generation After using the six 2D-ST-Datasets constructed in Section 4.1 to train the GAN respectively, we obtain six generator models, and construct sliding trajectory databases of six kinds of CAPTCHAs.

Preprocessing trajectories The trajectories in the 2D-ST-Dataset need to be filled and normalized before being input into the GAN. Due to the different sliding time and sliding distance. We fill $[0, 0]$ to unify the lengths of trajectories. Filling a trajectory with $[0, 0]$ means that when the slider reaches the notch position, the mouse stays for a short time and then releases, which is also in line with the characteristics of human sliding.

In the process of human sliding, the range of displacements in the horizontal direction is usually larger than that in the vertical direction. Therefore, Δx_i and Δy_i need to be normalized separately, and the specific calculation method is shown in Eq. 1 and Eq. 2:

$$\Delta x_i' = 2 \times \frac{\Delta x_i - \Delta x_{\min}}{\Delta x_{\max} - \Delta x_{\min}} - 1 \tag{1}$$

$$\Delta y_i' = 2 \times \frac{\Delta y_i - \Delta y_{\min}}{\Delta y_{\max} - \Delta y_{\min}} - 1 \tag{2}$$

[2] https://docs.ultralytics.com/tasks/detect

After filling the lengths of the trajectories, $\Delta x_{\min}$ and $\Delta x_{\max}$ respectively represent the minimum and maximum value of Δx in the 2D-ST-Dataset; $\Delta y_{\min}$ and $\Delta y_{\max}$ respectively represent the minimum and maximum values of Δy in the 2D-ST-Dataset.

Training GAN GAN consists of two parts: a generator and a discriminator. The input of generator is random noises, the output is fake samples. The training goal of generator is to improve the similarity between fake samples and real samples, so that they cannot be distinguished by the discriminator. The input of discriminator is real samples or fake samples, the output is the discrimination results. The training goal of discriminator is to distinguish real samples and fake samples.

The discrimination results are used to calculate the loss functions, and the loss of the discriminator is Eq. 3:

$$D_{loss} = -\frac{1}{m}\sum_{i=1}^{m}\left(\log\left(D\left(x^i\right)\right) + \log\left(1 - D\left(G\left(z^i\right)\right)\right)\right) \tag{3}$$

The loss of the generator is Eq. 4:

$$G_{loss} = -\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(G\left(z^i\right)\right)\right) \tag{4}$$

Where m represents the number of samples in each iteration, x represents the true sample, and z represents the noise.

The adversarial training process is performed alternately by two neural networks, and the discriminator is trained first in each step. The noises are generated randomly, and the noises are input into the generator to obtain fake samples. The samples and fake samples are input into the discriminator respectively, and the discriminator outputs the judgment results. Then, the weights of the discriminator are updated by the loss function 3. The generator is then trained. The fake samples are input into the updated discriminator, and the updated discriminator outputs the judgment results. Then, the weights of the generator are updated by the loss function 4. As the adversarial training progresses, the ability of the discriminator to discriminate between real and fake samples gradually improves. Meanwhile, in order to fool the discriminator, the data generated by the generator tends to the real samples.

Constructing two-dimensional sliding trajectory database By inputting a random noise to the trained generator, a list $[[\Delta x'_1, \Delta y'_1], [\Delta x'_2, \Delta y'_2], \cdots, [\Delta x'_k, \Delta y'_k]]$ can be generated. After denormalization of $\Delta x'_i$ and $\Delta y'_i$, a two-dimensional sliding trajectory can be obtained, and the specific calculation method is shown in Eq. 5 and Eq. 6:

$$\Delta x_i = (\Delta x'_i + 1) \times (\Delta x_{\max} - \Delta x_{\min}) \times 0.5 + \Delta x_{\min} \tag{5}$$

$$\Delta y_i = (\Delta y'_i + 1) \times (\Delta y_{\max} - \Delta y_{\min}) \times 0.5 + \Delta y_{\min} \tag{6}$$

Where $\Delta x_{\min}$, $\Delta x_{\max}$, $\Delta y_{\min}$ and $\Delta y_{\max}$ is as same as in Section 4.2.

We sequentially input 10,000 random noises into each generator. After denormalization, we select the trajectories whose sliding distances are different, the times of vertical displacement are greater than or equal to 5, and initial horizontal displacements are greater than or equal to 0 (Since the initial horizontal displacements of the trajectories in the dataset are small, the generator would generate the trajectories with negative initial displacements after training. This does not conform to the characteristics of human sliding), the two-dimensional sliding trajectory database can be constructed.

4.3 Attack algorithm

The attack algorithm is shown in Algorithm 1. Firstly, the attack script controls the browser to visit the target website that contains a slide puzzle CAPTCHA. Secondly, the attack script gets the URLs of the CAPTCHA background image and slider image, and downloads the CAPTCHA background image and slider image to the local disk. Thirdly, notch detection model is used to detect the notch position in the background image, and the position of the slider in the slider image is determined by pixel-by-pixel detection. Fourthly, the sliding distance is determined according to the notch position and the slider position. Then, the corresponding distance of the sliding trajectory is obtained from the two-dimensional sliding trajectory database. Finally, the script drags the slider to the notch position according to the sliding trajectory, and records the attack result and attack time.

Algorithm 1 Attack algorithm towards slide puzzle CAPTCHAs

1: $website \leftarrow$ `get("https://xxx.xxx.xxx")`
2: $(background\_image, slide\_image) \leftarrow$ `get`$(background\_url, slide\_url)$
3: $sliding\ distance \leftarrow$ `distance_predict`$(background\_image, slide\_image)$
4: $trajectory \leftarrow$ `get_trajectory`$(sliding\ distance)$
5: **for** `each` $sub-trajectory$ `in` $trajectory$ **do**
6: `Move`$(xoffset = sub-trajectory[0], yoffset = sub-trajectory[1], time = sliding\ time\ of\ sub-trajectory)$
7: **end for**
8: `save`$(attack\_result, attack\_time)$

Tencent CAPTCHA, GeeTest CAPTCHA v3, and Shumei CAPTCHA detect whether the browser is controlled by a script. Before the script opens the browser. Attack script needs to modify the property of brower to bypass the detection. Since NetEase CAPTCHA and Dingxiang CAPTCHA limit verification frequency, attack script will pause for 5 seconds and then carry out the next attack. Dingxiang CAPTCHA uses a stricter mechanism on the verification frequency, even if attack script stays for 5 seconds between attacks, it will fail to pass the verification after a large number of attacks. Therefore, when attacking Dingxiang CAPTCHA, we divide each 50 attacks into a group. After each group of attack, we wait for a few moments before performing the next group of attack.

5 Experiments and analysis

5.1 Notch detection

For notch detection, we try two traditional rule-based algorithms: edge detection [27] and template matching [26], as well as six object detection networks based on deep learning: Faster R-CNN [9], Mask R-CNN [20], SSD [21], Retina Net [10], DeTR [22] and YOLOv8[3]. In this paper, accurate detection means that the notch can be correctly detected without missing detection or false detection. The calculation formula of detection accuracy is shown in Eq. 7. The notch detection experiment is carried out on the test set, and the experimental results are shown in Table 2.

$$Accuracy = \frac{Samples\ of\ accurate\ detection}{All\ Samples} \tag{7}$$

Table 2. Notch detection results, where the metric is detection accuracy.

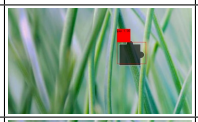
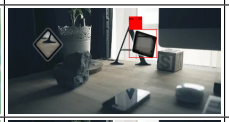
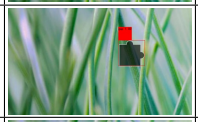
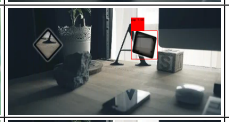
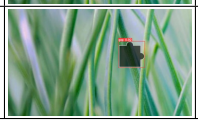
Method	CAPTCHA					
	Tencent	GeeTest v3	Shumei	NetEase	GeeTest v4	Dingxiang
Edge detection [27]	82%	24%	6%	8%	8%	68%
Template matching [26]	83%	98%	85%	85%	96%	88%
Faster R-CNN [9]	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
Mask R-CNN [20]	95%	72%	8%	65%	68%	54%
SSD [21]	**100%**	**100%**	94%	98%	**100%**	99%
Retina Net [10]	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
DeTR [22]	**100%**	**100%**	99%	98%	98%	92%
YOLOv8[3]	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**

The experimental results show that: (1) The edge detection algorithm and template matching algorithm achieve low accuracy of notch detection, which is consistent with the conclusion of Chang et al. [5]. It shows that sliding puzzle CAPTCHAs can effectively resist the notch detection by traditional algorithms. (2) The object detection networks based on deep learning show excellent performance on notch detection. Although the slide puzzle CAPTCHAs use rich and complex background images, and add interferences in the background images, they cannot resist the notch detection by object detection networks. The detection accuracies of SSD and DeTR for each CAPTCHA are more than 90%, and the detection accuracies of Faster R-CNN, RetinaNet and YOLOv8 for each CAPTCHA are 100%. However, the detection accuracies of Mask R-CNN are low, indicating that Mask R-CNN cannot be effectively used for notch detection in the case of training the model with a small scale of dataset. The notch detection accuracy of Faster R-CNN, Retina Net and YOLOv8 all achieves 100%. However, the prediction boxes drawn by Faster R-CNN sometimes fail to fit the notch edge completely, while the other two do not (see Table. 3). At the same

[3] https://github.com/ultralytics/ultralytics

time, the interface of Retina Net is more complex. Therefore, YOLOv8 is finally selected as the notch detection method in this paper.

Table 3. Notch detection results.

Detection network	Sample-a	Sample-b	Accurate fit
Faster R-CNN			×
Retina Net			✓
Yolov8			✓

5.2 Trajectory Generation

Training GAN. In Section 4.1, we construct a 2D-ST-Dataset for each type of slide puzzle CAPTCHA. After the trajectories in the dataset are filled and normalized, we start to train GAN. After adversarial training of the discriminator and the generator, the losses of the discriminator and the generator reach stability, as shown in Fig. 4, 5, 6. Then, we can get the trained generator model and build a human-like two-dimensional sliding trajectory database for each type of CAPTCHA.

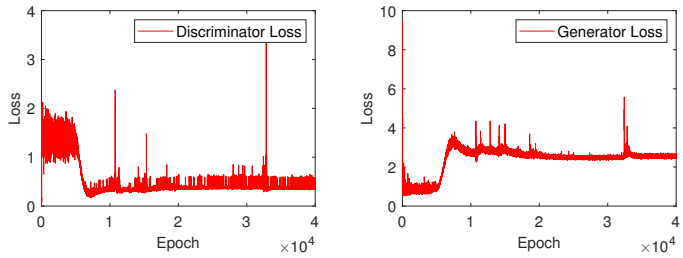


Fig. 4. Training loss of Tencent.

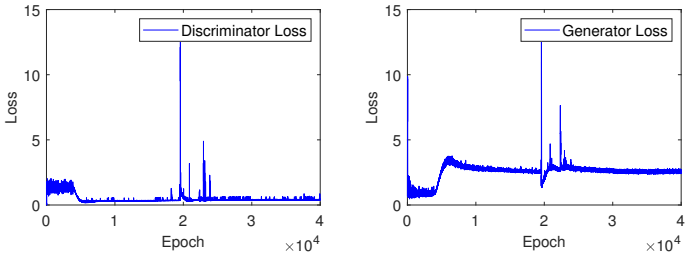


Fig. 5. Training loss of GeeTest v3.

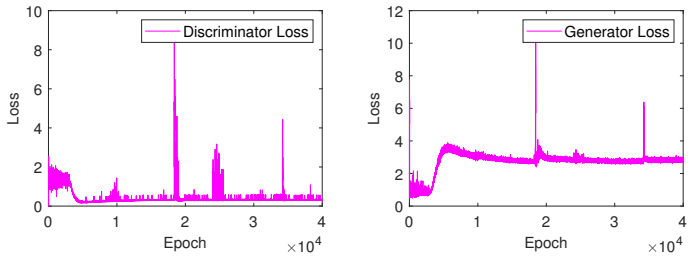


Fig. 6. Training loss of Shumei.

Using different kinds of sliding trajectories to attack slide puzzle CAPTCHAs In this section, we use eight kinds of methods to generate sliding trajectories to attack six kinds of slide puzzle CAPTCHAs. After determining the sliding distance, we use eight different kinds of trajectories to drag the slider to the notch. The first method directly drags the slider to the notch position by a single horizontal displacement in 0.1s (DD) [7,24]. The second method generates one-dimensional sliding trajectories in the horizontal direction by sigmoid function (1D-SF) [7,24]. The third method generates one-dimensional sliding trajectories in the horizontal direction by acceleration formulas (1D-AF) [5]. The fourth method generates one-dimensional sliding trajectories in the horizontal direction by log function (1D-LF) [5]. The fifth method uses sigmoid function to generate the sliding trajectories in the horizontal direction, and adds a small random displacement in vertical direction in each horizontal displacement (2D-SF). The sixth method uses acceleration formulas to generate the sliding trajectories in the horizontal direction, and adds a small random displacement in vertical direction in each horizontal displacement (2D-AF). The seventh method uses log function to generate the sliding trajectories in the horizontal direction, and adds a small random displacement in vertical direction in each horizontal displacement (2D-LF). The eighth method uses the trajectories in the database which are built based on GAN to attack the slide puzzle CAPTCHA, and the sliding time of each sub-trajectory is 0.05s (2D-GAN).

It is often necessary to click a button before a website can pop up a slide puzzle CAPTCHA. We start the timer by clicking the button and end it when the script drags the slider to the notch position and releases the mouse. Each method is tested 200 times, and the experimental results are shown in Table 4:

The trajectories generated by DD method are able to pass the NetEase CAPTCHA, GeeTest CAPTCHA v4 and Dingxiang CAPTCHA with the success rates of 100%, 100% and 97.5%. This indicates that they lack trajectory detection, because it is difficult for a human to drag the slider directly to the notch position within 0.1s. The failed Dingxiang CAPTCHAs are due to an inaccurate threshold to determine the starting pixel of the slider, which leads to deviations in the sliding distances. Using the sliding trajectories generated by other methods to attack NetEase CAPTCHA and Dingxiang CAPTCHA, the success rates are also nearly 100%. Using the sliding trajectories generated by

Table 4. Attack results on slide puzzle CAPTCHAs, where the metrics are success rate (SR) and average attack speed (AS, expressed in seconds).

Method		CAPTCHA					
		Tencent	GeeTest v3	Shumei	NetEase	GeeTest v4	Dingxiang
DD [7,24]	SR	6.5%	0%	0%	**100%**	**100%**	97.5%
	AS	**3.75s**	**2.58s**	**1.74s**	**2.02s**	**2.71s**	**1.64s**
1D-SF [7,24]	SR	6.5%	32.0%	0%	**100%**	**100%**	**100%**
	AS	8.66s	7.50s	7.56s	7.69s	8.75s	7.44s
1D-AF [5]	SR	7.5%	0%	0%	**100%**	88.5%	**100%**
	AS	7.37s	5.37s	5.77s	6.47s	7.16s	6.33s
1D-LF [5]	SR	7.0%	4.0%	0%	**100%**	96.5%	**100%**
	AS	6.60s	5.42s	5.48s	5.75s	6.60s	7.01s
2D-SF (Ours)	SR	**100%**	83.0%	99.5%	**100%**	**100%**	**100%**
	AS	8.67s	7.51s	7.59s	7.82s	8.77s	7.54s
2D-AF (Ours)	SR	**100%**	11.0%	99.0%	**100%**	93.5%	**100%**
	AS	7.39s	5.33s	5.80s	6.51s	7.30s	6.42s
2D-LF (Ours)	SR	99.5%	96.5%	99.0%	**100%**	97.5%	99.0%
	AS	6.68s	5.39s	5.38s	5.78s	6.78s	5.40s
2D-GAN (Ours)	SR	**100%**	**99.5%**	**100%**	**100%**	**100%**	**100%**
	AS	4.16s	2.87s	3.01s	3.44s	4.10s	2.81s

1D-AF, 1D-LF, 2D-AF and 2D-LF methods to attack GeeTest CAPTCHA v4, although achieveing high success rates, they do not achieve 100%. This is because the sliding trajectories generated by these methods may cause the sliders to first exceed the notch positions before returning during the sliding process. If slides touch the boundary of the CAPTCHA background images, the verification will fail.

Using the one-dimensional sliding trajectories to attack Tencent CAPTCHA, GeeTest CAPTCHA v3 and Shumei CAPTCHA, the success rates are low. Using the one-dimensional sliding trajectories to attack Tencent CAPTCHA, only the first few attacks are successful, so the success rates are about 7.0%. For GeeTest CAPTCHA v3, the trajectories generated by DD method and 1D-AF method fail to pass the verification. The sliding trajectories generated by 1D-SF method and 1D-LF method have a short pause when attacking, which may lead to the improvement of the success rates, achieving 32.0% and 4.0%, respectively. The one-dimensional sliding trajectories cannot pass the detection of Shumei CAPTCHA.

Using the two-dimensional sliding trajectories to attack Tencent CAPTCHA, GeeTest CAPTCHA v3 and Shumei CAPTCHA, the success rates are significantly improved compared with the one-dimensional sliding trajectories. The two-dimensional sliding trajectories generated by 2D-SF, 2D-AF, 2D-LF, 2D-GAN methods all break Tencent CAPTCHA and Shumei CAPTCHA with success rates close to 100%. GeeTest CAPTCHA v3 also adopts strict detection

of two-dimensional sliding trajectories. Only the two-dimensional sliding trajectories generated by 2D-GAN method achieve the success rate of 99.5%, while the attack success rates of the two-dimensional trajectory generated by 2D-SF, 2D-AF, 2D-LF methods are only 83.0% ,11.0% and 96.5%, respectively.

According to the experimental results, the two-dimensional sliding trajectories generated by 2D-GAN method can effectively pass the trajectory detection, and the attack speeds are significantly shorter than other kinds of two-dimensional sliding trajectories. However, this method also has shortcomings. The sliding trajectories generated by the above process cannot completely cover all the sliding distances, especially the longer ones. However, by increasing the number of datasets, select the sliding trajectories with longer sliding distances to train GAN, we can obtain the sliding trajectories with longer sliding distances. This experiment shows that the human-like two-dimensional sliding trajectories generated by GAN can effectively simulate the characteristics of human sliding, and break the trajectory detection of slide puzzle CAPTCHAs.

Parameter analysis experiment Before constructing 2D-ST-Datasets, it is necessary to determine the optimal time interval Δt for collecting human sliding trajectories. The human sliding time is generally 1-2 seconds. For Tencent CAPTCHA, GeeTest CAPTCHA v3 and Shumei CAPTCHA with trajectory detection, when the time interval $\Delta t = 0.8s, 0.6s, 0.4s, 0.2s, 0.1s, 0.05s$, we respectively collected 50 verified two-dimensional human sliding trajectories with different sliding distances.

The collected sliding trajectories are used for replay attacks. When replaying each sub-trajectory of a trajectory, the sliding time is as same as the time interval Δt when the trajectory is collected. The number of collected trajectories is small, and the trajectories cannot cover all sliding distances of CAPTCHAs. In the replay attack, only experiments are recorded in which, after the sliding distance is determined, the trajectory of the corresponding sliding distance is collected. Each scheme is experimented 200 times. The experimental results are shown in Table 5, and the line chart of success rates is shown in Fig. 7.

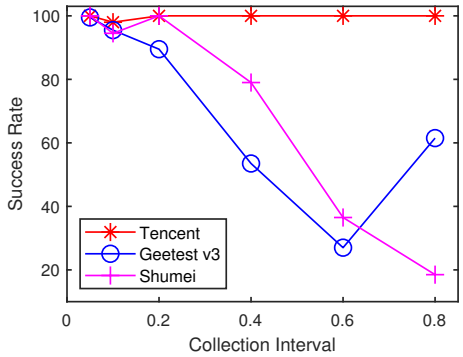


Fig. 7. Line chart of the success rates of the trajectories replay.

Table 5. Attack results on slide puzzle CAPTCHAs, where the metrics are success rate (SR) and average attack speed (AS, expressed in seconds).

Collection interval (Δt)		CAPTCHA		
		Tencent	GeeTest v3	Shumei
$\Delta t = 0.8s$	SR	**100%**	61.5%	18.5%
	AT	5.11s	3.09s	3.12s
$\Delta t = 0.6s$	SR	**100%**	27.0%	36.5%
	AT	3.67s	2.85s	2.8s
$\Delta t = 0.4s$	SR	**100%**	53.5%	79.0%
	AT	3.78s	2.58s	2.95s
$\Delta t = 0.2s$	SR	**100%**	89.5%	**100%**
	AT	3.79s	2.54s	2.79s
$\Delta t = 0.1s$	SR	98.0%	95.5%	94.5%
	AT	3.54s	2.45s	2.66s
$\Delta t = 0.05s$	SR	**100%**	**99.5%**	**100%**
	AT	**3.32s**	**2.41s**	**2.24s**

According to the experimental results, replaying the trajectories collected at time interval $\Delta t = 0.05s$, Tencent CAPTCHA, GeeTest CAPTCHA v3 and Shumei CAPTCHA all achieve the highest success rates of replay, which are 100%, 99.5%and 100% respectively. Therefore, the optimal collection time interval $\Delta t = 0.05s$.

6 Conclusion

Slide puzzle CAPTCHAs are becoming more and more popular due to user-friendliness and diverse defense mechanisms. In order to test the security of the existing slide puzzle CAPTCHAs, we select six kinds of very popular slide puzzle CAPTCHAs including Tencent CAPTCHA, NetEase CAPTCHA, GeeTest CAPTCHA v3, GeeTest CAPTCHA v4, Shumei CAPTCHA and Dingxiang CAPTCHA.

Firstly, we construct and publicly release a ND-Dataset and a 2D-ST-Dataset of each type of CAPTCHA. Secondly, we propose a novel slide puzzle CAPTCHA attack method based on GAN to generate human-like two-dimensional sliding trajectories. Finally, six kinds of slide puzzle CAPTCHAs are broken with the success rates close to 100%. Experimental results show that the attack method proposed in this paper can effectively break the existing mainstream slide puzzle CAPTCHAs, and the security of the existing slide puzzle CAPTCHAs faces great challenges.

References

1. Luis von Ahn, Manuel Blum, and John Langford. Telling Humans and Computers Apart Automatically [J]. Communications of the ACM, 2004, 47(2): 56–60.
2. Mengyun Tang, Haichang Gao, Yang Zhang, Yi Liu, Ping Zhang, Ping Wang. Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha [J]. IEEE Transactions on Information Forensics and Security, 2018, 13(10): 2522-2537.
3. Haichang Gao, Mengyun Tang, Yi Liu, Ping Zhang, Xiyang Liu. Research on the Security of Microsoft's Two-Layer Captcha [J]. IEEE Transactions on Information Forensics and Security, 2017, 12(7): 1671-1685.
4. Qiujie Li. A Computer Vision Attack on the ARTiFACIAL CAPTCHA [J]. Multimedia Tools and Applications, 2015, 74(13): 4583-4597.
5. Guoqin Chang, Haichang Gao, Ge Pei, Sainan Luo, Yang Zhang, Nuo Cheng, Yiwen Tang, Qianwen Guo. The Robustness of Behavior-Verification-Based Slider CAPTCHAs [J]. Journal of Information Security and Applications, 2024, 81: 103711.
6. Carlos J. Hernández-Castro, María D. R-Moreno, David F. Barrero. Using JPEG to Measure Image Continuity and Break Capy and Other Puzzle CAPTCHAs [J]. IEEE Internet Computing, 2015, 19(6): 46-53.
7. Haiqin Weng, Binbin Zhao, Shouling Ji, Jianhai Chen, Ting Wang, Qinming He, Raheem Beyah. Towards Understanding the Security of Modern Image Captchas and Underground Captcha-Solving Services [J]. Big Data Mining and Analytics, 2019, 2(2): 118-144.
8. Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, Jieping Ye. Object Detection in 20 Years: A Survey [J]. Proceedings of the IEEE, 2023, 111(3): 257-276.
9. Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137-1149.
10. Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár. Focal Loss for Dense Object Detection [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020, 42(2): 318-327.
11. Jie Gui, Zhena Sun, Yonggang Wen, Dacheng Tao, Jieping Ye. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications [J]. IEEE Transactions on Knowledge and Data Engineering, 2023, 35(4): 3313-3332.
12. Ping Wang, Haichang Gao, Qingxun Rao, Sainan Luo, Zhongni Yuan, Ziyu Shi. A Security Analysis of Captchas With Large Character Sets [J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(6): 2953-2968.
13. Chunhui Li, Xingshu Chen, Haizhou Wang, Yu Zhang. End-to-End Attack on Text-based CAPTCHAs Based on Cycle-Consistent Generative Adversarial Network [J]. Neurocomputing, 2021, 433: 223-236.
14. Haichang Gao, Wei Wang, Jiao Qi, Xuqin Wang, Xiyang Liu, Jeff Yan. The Robustness of Hollow CAPTCHAs [C], Proceeding of the 20th ACM SIGSAC Conference on Computer & Communications Security (CCS 2013), Berlin, Germany, Novermber 4-8, 2013, pp. 1075–1086.
15. Haichang Gao, Lei Lei, Xin Zhou, Jiawei Li, Xiyang Liu. The Robustness of Face-Based CAPTCHAs [C], Proceeding of 15th IEEE International Conference on Computer and Information Technology (CIT 2015), Livepool, UK, October 26-28, 2015, pp. 2248-2255.
16. Philippe Golle. Machine Learning Attacks Against the Asirra CAPTCHA [C], Proceeding of 15th ACM Conference on Computer and Communications Security (CCS 2008), Alexandria, Virginia, USA, October 27-31, 2008, pp. 535–542.

17. Suphannee Sivakorn, Iasonas Polakis, Angelos D Keromytis. I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs [C], Proceeding of 1st IEEE European Symposium on Security and Privacy (EuroS&P 2016), Saarbruecken, Germany, March 21-24, 2016, pp. 388-403.
18. Rosa Lin, Shi-Yu Huang, Graema B Bell, Yeuan-Kuen Lee. A New CAPTCHA Interface Design for Mobile Devices [C], Proceeding of 12th Australasian User Interface Conference (AUIC 2011), Perth, Australia, January 17-20, 2011, pp. 3-8.
19. Danni Wu, Jing Qiu, Huiwu Huang, Lihua Yin, Zhaoquan Gu, Zhihong Tian. Resnet-Based Slide Puzzle Captcha Automatic Response System [C], Proceeding of 6th International Conference on Artificial Intelligence and Security (ICAIS 2020), Hohhot, China, July 17-20, 2020, pp. 140-153.
20. Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask R-CNN [C], Proceeding of 16th IEEE International Conference on Computer Vision (ICCV 2017), Venice, Italy, October 22-29, 2017, pp. 2980-2988.
21. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C Berg. SSD: Single Shot MultiBox Detector [C]. Proceeding of 14th European Conference on Computer Vision (ECCV 2016), Amsterdam, The Netherlands, October 11-14, 2016, pp. 21-37.
22. Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko. End-to-End Object Detection with Transformers [C], Proceeding of 16th European Conference on Computer Vision (ECCV 2020), Glasgow, UK, August 23-28, 2020, pp. 213—229.
23. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets [C], Proceeding of 27th International Conference on Neural Information Processing Systems (NIPS 2014), Montreal, Canada, December 8-13, 2014, pp. 2672–2680.
24. Binbin Zhao, Haiqin Weng, Shouling Ji, Jianhai Chen, Ting Wang, Qinming He, Reheem Beyah. Towards Evaluating the Security of Real-World Deployed Image CAPTCHAs [C], Proceeding of 11th ACM Workshop on Artificial Intelligence and Security (AISec 2018), Toronto, Canada, October 15-19, 2018, pp. 85–96.
25. Kumar Chellapilla, Kevin Larson, Patrice Y Simard, Mary Czerwinski. Building Segmentation Based Human-Friendly Human Interaction Proofs (HIPs) [C], Proceeding of 2th Human Interactive Proofs (HIP 2005), Bethlehem, PA, USA, May 19-20, 2005, pp. 1-26.
26. Artiom Basulto-Lantsova, Jose A Padilla-Medina, Francisco J. Perez-Pinal, Alejandro I. Barranco-Gutierrez. Performance Comparative of OpenCV Template Matching Method on Jetson TX2 and Jetson Nano Developer Kits [C], Proceeding of 10th Annual Computing and Communication Workshop and Conference (CCWC 2020), Las Vegas, NV, USA, January 6-8, 2020, pp. 812-816.
27. Samah Rahamneh, Lina Sawalha. Efficient OpenCL Accelerators for Canny Edge Detection Algorithm on a CPU-FPGA Platform [C], Proceeding of 14th International Conference on ReConFigurable Computing and FPGAs (ReConFig 2019), Cancun, Mexico, December 9-11, 2019, pp. 1-5.