Regular Paper

Evaluation of Ergonomically Designed CAPTCHAs using Deep Learning Technology

Tomoka Azakami[1,a)] Chihiro Shibata[1,b)] Ryuya Uda[1,c)]

Received: November 10, 2017, Accepted: June 8, 2018

Abstract: Although most existing text-based CAPTCHAs use distorted images of alphanumerics, they have been criticized because large image distortions make it difficult for human beings to recognize the characters, despite the ease with which computers can eliminate distortions and consequently recognize them. Ergonomically designed CAPTCHAs, which exploit human-specific phenomena, are a solution to this problem. Ergonomic design enables humans to momentarily recognize them, while significantly increasing computational costs for machines to recognize characters. Recently, owing to the development of deep learning, the image recognition capability of machines has improved dramatically. Characters are easily recognized within reasonable time by deep convolutional neural networks (DCNNs), which have similar architectures to visual perception mechanisms of the brain. In this paper, to clarify whether ergonomically designed CAPTCHAs can withstand state-of-the-art methods of deep learning, we use several kinds of DCNNs to measure the classification rates of the characters displayed. With respect to ergonomic designs, we first use Amodal CAPTCHA proposed by Mori et al., which exploits the two human-specific phenomena of amodal completion and aftereffects. We secondly modify Amodal CAPTCHA by adding jagged lines to the edges of characters, aiming to prevent DCNNs from recognizing them correctly, since edges are one of the most fundamental features for DCNNs. Experimental results, however, show that both naive and jagged-lined Amodal CAPTCHAs are almost completely broken. Another approach we conducted is to use only complete characters without shielding as training data, assuming that attackers have no information about how amodal completion and jagged edges were applied. However, even for this assumption, the classification rate of DCNNs is still sufficiently high. On the whole, our results in this paper show that any ergonomic effects such as amodal completion and jagged edges are no longer countermeasures against character recognition by DCNNs.

Keywords: CAPTCHA, amodal completion, ergonomic effects, convolutional neural networks

1. Introduction

Completely automated public Turing test to tell computers and humans apart (CAPTCHA) is a technology that prevents computers from creating new web service accounts. CAPTCHA was first proposed in 1950 by Turing [1]. He advocated an artificial intelligence (AI) test to evaluate "awareness" in computers. Coates et al. forwarded the opinion that CAPTCHA is a "reverse" Turing test [2]. CAPTCHA is used to distinguish computers from humans, because autonomous computers (i.e., bots) are common tools for attackers. Often, attackers use bots to participate in online voting activities directly or indirectly related to finance, because public sentiment pushes value trends, ultimately benefiting the attackers. Web service providers wish to eliminate bot accounts, because the unfair, non-human activities can inflate value and harm credit.

CAPTCHA has been commonly used to thwart bots as it is effective and cheap. CAPTCHA provides questions that are easily answered by human beings, but difficult for bots to answer. If a person correctly answers a question, the CAPTCHA judges the user to be a human. There are several types of CAPTCHA, the most popular being text-based. With this method, alphanumeric characters are randomly shown on the screen. The characters are distorted and noise is added, but they can be most often recognized by humans. However, the characters are sometimes extremely distorted to avoid machine optical character reader (OCR) attacks, thus decreasing accessibility to humans. OCR methods have rapidly improved. Thus, it has been recently reported that all text-based CAPTCHA algorithms are now breakable by bots [3].

In addition, recently George et al. succeeded in solving CAPTCHA more than 300 times more efficiently than the conventional method using the model called Recursive Cortical Network (RCN) [4]. In this thesis it is stated that it is possible to read characters that are arranged side by side rather than split characters.

The inefficiency of CAPTCHA has long been warned about for by researchers. For instance, the website, PWNtcha [*1], displays various images of CAPTCHAs defeated by the PWNtcha project [5]. In practice, there are other types of CAPTCHA methods, such as image and audio CAPTCHA. The basis for our OCR read-resistant CAPTCHA as described in Section 3.1 is that it takes too much time for a computer to emulate amodal comple-

[1] Tokyo University of Technology Graduate School, Hachioji, Tokyo 192–0914, Japan
[a)] g2117001ca@edu.teu.ac.jp
[b)] shibatachh@stf.teu.ac.jp
[c)] uda@stf.teu.ac.jp

[*1] http://caca.zoy.org/wiki/

tion. However, we admit that it may be possible for newer machine learning techniques to defeat Amodal CAPTCHA without amodal completion. If so, Amodal CAPTCHA will no longer be safe.

Furthermore, we had experiments in Section 7 in which it was tested whether samples with obstacles was able to be recognized by learned model by samples without obstacles. If the accuracy of the experiments exceeds 99%, it is suggested that malware can break through CAPTCHA without relearning since samples for learning have no knowledge of obstacles on samples for test. That is, if DCNN marks a good score in the experiments, algorithms for appearance of obstacles have no influence on the classification of characters on DCNN. Change of the algorithms or whether they are secret or not is no longer a matter of the DCNN classification. Therefore, malware does not need relearning. Malware that attacks CAPTCHA is considered to be independent. The reasons are described below. Attackers hijack someone else's computer and execute the malware, but they dislike that the owner is noticed that it is being hijacked or that the communication record is specified. Therefore they do not like opening the backdoor all the time to communicate.

Computers which can easily be attacked are those which have poor security because the OS is old and updates have not been carried out. Attackers dislike fully using a CPU for deep learning on a victim's PC if it leads to it being noticed more easily.

Of course, malware which contains the same learned model can be detected by pattern matching method since the same model is constructed by the same binary patterns. However, in this paper, we focus unknown malware. In other words, unknown malware cannot be detected by pattern matching while it can be detected by behavior analysis such as communicating with attackers' site, fully using a CPU, etc. When new malware with new models appears, no one can detect it by pattern matching. Moreover, the binary patterns of the model will be veiled in polymorphic malware.

Therefore, in this study, we evaluate our new CAPTCHA by subjecting it to machine learning recognition algorithms. Thus, we gain an understanding of how deep learning captures and analyzes edges of characters as features. We then break the edges by making them jagged and evaluate deep learning recognition again in order to investigate whether the final accuracy of deep learning can be lowered.

This paper is the revised version of a paper [27] in COMPSAC. In the paper in COMPSAC, we thought that jagged edges can decrease the accuracy of deep learning, we were wrong. We discuss it in this paper. Furthermore, we also investigated whether deep learning can break the CAPTCHA without the knowledge of appearance algorithms of obstacles for amodal completion.

2. Related Works

CAPTCHA has also been developed with images and sounds, because text-based CAPTCHA is no longer safe. However, most new CAPTCHA techniques have problems and complications. Several of those methods are introduced in this section.

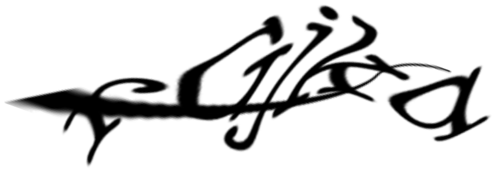


Fig. 1 Example of Text-Based CAPTCHA.

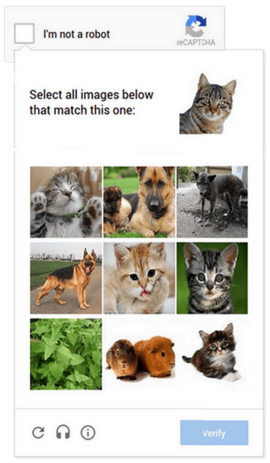


Fig. 2 Example of image CAPTCHA.

2.1 Text-Based CAPTCHA

The most popular type of CAPTCHA is text-based. **Figure 1** is an example of text-based CAPTCHA that can be easily analyzed with OCR algorithms. According to Mori et al. [6], text-based CAPTCHA is 92% breakable. Therefore, text-based CAPTCHA's human-readability continues to decrease as a countermeasure, rendering it nearly useless.

2.2 Image CAPTCHA

Image CAPTCHA was proposed as an alternative to text-based CAPTCHA. To pass this CAPTCHA, humans select from several available images those that help answer a question. The most famous of image CAPTCHA is reCAPTCHA, by Google. **Figure 2** is example of reCAPTCHA. It is difficult for both humans and bots to read reCAPTCHA, because image content is quite distorted. Even so, reCAPTCHA and other image CAPTCHAs can be attacked using a database, because images are reused. Thus, attackers can replicate the CAPTCHA database by using a human to solve the problems in advance.

2.3 Audio CAPTCHA

Audio CAPTCHA distinguishes bots from humans by transmitting sounds to the user, requiring a variety of responses. This CAPTCHA does not require visuals, such as text and images. However, because CAPTCHA pronunciation is the same for all questions, it can be defeated by collecting sound registration patterns in a database.

2.4 Video CAPTCHA

Video CAPTCHA is considered a subspecies of image CAPTCHA. Several images are sequentially displayed, making machinery analysis more difficult. An example is NuCAPTCHA, by Leap Marketing Technologies, a Canadian software company [7]. In NuCAPTCHA, alphanumeric characters are randomly generated using multiple fonts, and moved around on the screen. Human beings can easily recognize the characters, because they remain grouped together in motion. Furthermore, NuCAPTCHA alters recognition difficulty per the prediction of whether an examinee is a human or a bot. Thus, both security and usability remain high. However, NuCAPTCHA can also be broken, because video can be separated into multiple static images, which are readable with OCR technology.

2.5 Image Recognition and Machine Learning

2.5.1 Text Recognition using Neural Networks

Neural networks automatically infer rules for distortion or noise in recognition of alphanumeric characters. With ordinary pattern-matching methods, character patterns are collected to infer their generation logic. If the generation logic changes, inference cannot be reused, and recognition algorithms also change. Alternatively, with neural networks, the same algorithm is adaptable to different CAPTCHA algorithms. Chellapilla et al. proved that one neural network algorithm can analyze characters generated from multiple CAPTCHA [*2] engines [3]. In their report, the correct answer rate was 5–60%, indicating that their algorithm is a threat to existing text-based CAPTCHAs, because bots can retry any number of times. The recognition rate of characters by human beings compared to bots is reported in another paper by Chellapilla et al. [8]. CAPTCHA analysis procedures consist of two steps: segmentation and recognition. Segmentation extracts the shape of each character, and recognition infers each extracted character. Segmentation is notably more difficult for computers. In the paper by Chellapilla el al., the recognition accuracy of bots is compared to that of humans, and bots' recognition accuracy is shown to be better after the character is extracted from a picture. Thus, we should focus on making it more difficult for bots to recognize the characters in the first place. Human vision and cognition have been long-researched, and various models of completion have been proposed. Fukushima et al. [9] used neural network technology for character recognition. They showed that characters can be easily recognized when they are not very distorted or obscured.

2.5.2 Convolutional Neural Networks

Deep learning technology has made a significant breakthrough in the field of AI in the last several years. This technology is now bringing widely innovative changes to various fields where machine learning techniques are potentially available. Notably, progress is dramatic in the areas of image recognition, speech recognition, and natural language processing, which are closely related to machine learning. In deep learning, we optimize a neural network, which has many neurons with many layers and weight parameters. There are several types of famous network architectures used for deep neural networks. Convolutional neural network (DCNN) is one of them. Various types of DCNN have been proposed recently such as LeNet, AlexNet, VGGNet GoogLeNet and Residual Network (ResNet) [10], [11], [12], [13], [14].

DCNN significantly improves the capability of machines to complete various tasks of image recognition. This is achieved by combining many convolutional and pooling layers, by increasing computational speed, by developing new learning methods, and by using appropriate nonlinear functions. As far back as 1989, Cun et al. used DCNN for character recognition [15], [16]. A DCNN architecture called AlexNet, proposed by Krizhevsky et al. achieved the highest classification result in a competition called ImageNet Large-Scale Visual Recognition Challenge (ILSVRC), in 2012. Hence, more sophisticated and deeper structures of DCNNs have been proposed and developed in search of even higher accuracies.

VGGNet is one of the networks that produced good results in image recognition [12]. It finally outputs the classification probabilities through three layers of all connections after repeating the combination of 3×3 convolution and 2×2 max pooling multiple times.

GoogLeNet is a neural network structure that has 100 layers (i.e., building blocks) with several million parameters. State-of-the-art studies have shown that, by learning from large dataset (e.g., ILSVRC), DCNNs can classify images as well as or even better than human beings. In our experiments, we use AlexNet.

3. CAPTCHA with Amodal Completion

3.1 Overview of CAPTCHA with Amodal Completion

We introduce a concept and the history of our Amodal CAPTCHA in this chapter. The first appearance of CAPTCHA with amodal completion was that by Mori et al. [17]. We call it MUK CAPTCHA in this paper. The screen shot of MUK CAPTCHA is shown in **Fig. 3**. Parts of characters of "A", "B", "C" and "D" are moving on the screen and the parts make one character at a time. In this figure, "A" is completely made at this moment. Each character is made one by one, but not at the same moment. The computer does not know when a character is complete so that it must keep reading characters frame by frame. Computer must also emulate amodal completion when OCR is applied since each character is shielded by obstacles.

Before it, all text-based CAPTCHA methods depended on distortion or noise. This was the first challenge to make CAPTCHA difficult for computers and not to make it difficult for human beings at the same time. Mori et al. focused on amodal completion which is one of the abilities of human beings. It was also known that amodal completion by human beings was able to be emulated

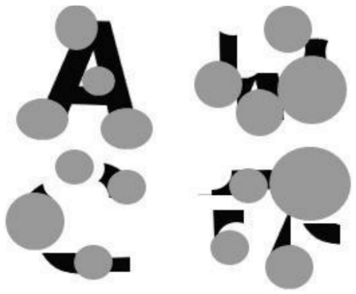


Fig. 3 MUK CAPTCHA.

[*2] Chellapilla did not use the word, "CAPTCHA", but instead used the word "HIP", which means "human interaction proof", CAPTCHA was named by a team at Carnegie Mellon University. We use the word "CAPTCHA" in this paper, because it is most popular in common use.

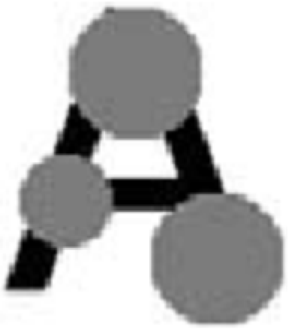


Fig. 4 SU CAPTCHA.

by computers. The most significant point of MUK CAPTCHA was easily force the heavy calculation to bots for the emulation so that the calculation never ended in the given time limit. On the other hand, MUK CAPTCHA was easy to be recognized by human beings with amodal completion. Human beings can quickly and commonly recognize objects even when they are obscured; this feature is called amodal completion in the field of perceptual psychology. Masnou et al. describe amodal completion via level line structures [18]. Amodal completion is a theory advocated by Kanizsa in 1996. It describes the visual acuity of human beings [19]. We, therefore, apply amodal completion for alphanumeric CAPTCHA character objects.

Amodal completion is emulated by computers with neural network technology, as mentioned by Fukushima et al. Instead of being used to prevent bots from recognizing characters, it assists humans with recognition, according to Mori et al. In their method, humans can recognize characters at four moments within ten seconds, when three hundred images are included. Therefore, bots must analyze three hundred images to find the four in which characters can be recognized: an easy task for humans using amodal completion. The cost for machine analysis for three hundred images is very high, and cannot be ignored by attackers. Unfortunately, this type of CAPTCHA has a problem. Character recognition is easy for human beings when they can choose images from a video. However, choosing requires high concentration. Completion of characters is not simultaneous as shown in Fig. 3. That is, for human beings, recognition is easy, but segmentation is difficult.

This problem was addressed by Sawada et al. [20]. They extended the time for recognition by combining aftereffects with amodal completion. This CAPTCHA name is Sawada, Uda CAPTCHA(SU CAPTCHA). The screen shot of SU CAPTCHA is shown in **Fig. 4**. A character "A" is drawn with obstacles in this figure. Characters gradually appear in SU CAPTCHA so that visibility of a character increases by aftereffects. An aftereffect is the phenomenon where human beings can continue seeing an object just after it disappears. We apply aftereffects to CAPTCHA to prevent bots from gaining an advantage from analyzing one frame of a video. Furthermore, it helps humans recognize characters within a period that is longer than one frame of video. That is, recognition time is extended by aftereffects. However, Sawada's algorithm did not work well, because the aftereffects were too weak to support amodal completion. Thus, we added colors to enhance recognition by aftereffects [21]. We investigated combinations of complementary colors and luminance in experiments. Hence, we succeed in prolonging the viewable time for humans. This is called Amodal CAPTCHA as shown in **Fig. 5**, and Amodal CAPTCHA is used for all experiments in this paper.

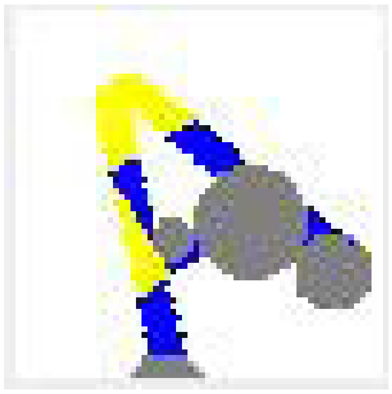


Fig. 5 Amodal CAPTCHA.

Our Amodal CAPTCHA had been a good solution for bots until deep learning became famous in the world. Because, as mentioned in Section 1, our CAPTCHA takes too much time to emulate amodal completion. When limiting the solution time of CAPTCHA, taking too much time to answer it is fatal to bot. Especially, DCNN (Deep Convolutional Neural Network) is widely used for image recognition now.

The most significant features of DCNN are flash classification and high accuracy. On the other hand, security of our Amodal CAPTCHA depends on the heavy calculation of amodal completion by computer. That is, if DCNN can recognize Amodal CAPTCHA without emulating amodal completion, not only usual text-based CAPTCHA but also special CAPTCHA with heavy calculation is no longer safe. Therefore, we tried to evaluate how accurate DCNN recognized Amodal CAPTCHA in Section 5.

To tell the truth in advance, DCNN almost perfectly broke Amodal CAPTCHA without emulating amodal completion. After this first experiment, we applied jugged edge to characters of Amodal CAPTCHA since we thought that the accuracy of DCNN depends on the edge of characters. Of course, we eliminated adding distortion and noise from our experiments since it had been known that they had decreased the visual performance of human beings. The experiments and evaluation is mentioned in Section 6. We also evaluated the size and frequency of jugged edges which did not decrease the visual performance of human beings.

Finally, we tried to eliminate the last possibility of a trick in the recognition by DCNN since we had doubts about extreme high accuracy by DCNN against Amodal CAPTCHA. It is the position of shields in front of characters. The position is fixed in our algorithm while the size is different. We thought that DCNN depends on the position in recognition. Therefore, we experimented and evaluated it in Section 7.

3.2 Amodal CAPTCHA and Other Text-based CAPTCHA

All major text-based CAPTCHA algorithms have been broken by OCR with a success of 92% according to the research by Mori,

C. et al. [6]. Of course, strong distortion of characters can decrease the readability of OCR. However, it also decreases the readability of human beings so that another solution is desired. Mori, T. et al. invented the first Amodal CAPTCHA as a countermeasure of OCR reading [17]. They succeeded to increase the cost for calculation on reading by OCR while human beings were able to recognize characters easily with amodal completion.

Amodal CAPTCHA has not been used in practice since image CAPTCHA such as Google reCAPTCHA has emerged. Image CAPTCHA overcame the reading by OCR. However, it requires huge amount of images and only a few companies such as Google can collect them. Therefore, text-based CAPTCHA is still used in services by small companies although it is no longer safe.

That is, text-based CAPTCHA which has tolerance to OCR is only Amodal CAPTCHA and only text-based CAPTCHA can be used without depending on a big company such as Google. This is the reason why we focus on Amodal CAPTCHA as a representative of CAPTCHA algorithms. Furthermore, DCNN was not major when the first Amodal CAPTCHA appeared. Therefore, we investigated whether Amodal CAPTCHA had tolerance to DCNN and we found not [28]. In this paper, we put jagged edges on characters of Amodal CAPTCHA and investigate how much the jagged edges interfere with DCNN.

4. Assumption of Threat Models

Threat models for Amodal CAPTCHA are described in this chapter. The threat we assume is automatic image recognition of Amodal CAPTCHA by CNN while Amodal CAPTCHA has resistance against OCR as mentioned by Mori et al. Attacks in the models are classified into some cases.

The first classification is "with the knowledge of the algorithms" and "without the knowledge of the algorithms". With the knowledge of the algorithms, attackers can create many images for machine learning when our CAPTCHA is published as open source software. Malware may be created with learned models. Experiments which assume the threat are described in Sections 5.3 and 5.4. Without the knowledge of the algorithms, attackers can also create images by manual screen shot of Amodal CAPTCHA on web browsers. However, it requires much time so that we do not think that it is one of the threats. If it does, the corresponding experiments are also in Sections 5.3 and 5.4.

The second classification is "follow the algorithms" and "ignore the algorithms" without the knowledge of the algorithms. If attackers follow the algorithms without the knowledge of the algorithms, they collect images manually as mentioned above. However, they can also attack by ignoring the algorithms, especially appearance algorithms of obstacles. Experiments which assume the threat are described in Section 7.

5. Evaluation of Amodal CAPTCHA using DCNNs

As mentioned above, while SU CAPTCHA is resistant to traditional methods for image recognition, it is newly threatened by DCNNs now. In fact, using deep learning to break CAPTCHA has already been accomplished. According to Sivakorn et al., reCAPTCHA is solved by deep learning 70.78% of the time [22]. It only takes 19 seconds. Levellines used DCNN for classification of a small data set of Baidu CAPTCHA. Thus, it gained an accuracy of 98.4% [23]. As mentioned earlier, Amodal CAPTCHA is resistant to conventional threats. However, it is necessary to check to see if it is resistant to deep learning.

We first test whether DCNN will break our system, which displays characters exploiting amodal completion and aftereffects. Via our implementation, each character gradually appears and disappears, because aftereffects are applied. Because each image consists of only a small part of the whole image of the character, and each is displayed each for just a moment, humans can recognize the entire shape owing to the aftereffects over many sequential images. It is difficult for DCNN to process such images quickly enough, so, as an attacker, we must choose to either preprocess images appropriately before learning, or to compose another neural network architecture to handle them.

One approach to solve the attack problem is simply overlaying sequential images while a character is drawn. In other words, we pile up a sequence of instantaneous images to make one image. In this case, what the neural network must learn is not only how to classify characters, but also, how to distinguish those already drawn from those still being drawn.

Another potential approach is using the combination of a DCNN and LSTM. Such combinations may be preferable to capturing both aftereffects and amodal completion at the same time, but the computational cost for learning and prediction grows much higher over DCNN alone, because we must input many sequential images to the neural network for each character.

We take the former approach in this paper, because it is simpler and sufficient for our purpose.

5.1 Preparation of Data

We prepared two datasets of Amodal CAPTCHA. They consist of English capital alphanumerics. See **Fig. 6**. All images are obtained by sequentially superimposing images appearing within the problem. For each character, we divide images into three categories since characters are drawn as the images. The first is a set of *complete characters* (COMPLETE), which are correct characters that human beings recognize via amodal completion. The second is a set of *incomplete characters* (INCOMPLETE), which are characters that human beings recognize when the drawing is not yet finished. The last is a set of *dummy characters* (DUMMY), which are randomly generated drawings that human beings do not recognize as either complete characters or digits. The small dataset consists of only six characters: {'A,' 'B,' 'K,' 'R,' 'M,' and 'W'}, whereas the large dataset consists of 24 characters, (i.e., all characters of the alphabet except for 'C' and 'O') and 10 digits. The reason for the exception is that 'C' and 'O' are difficult to be classified by human beings when obstacles shield these characters. Mori et al. did not use 'C' and 'O' in MUK CAPTCHA and neither do we.

The reason for choosing those six characters for the small data set is explained below. 'A' is the first alphabet character which is usually shown as a representative of alphabets. 'B,' 'R,' and 'K' appear similar while each character is being drawn, although obscured, as in **Fig. 7**. 'M' and 'W' have almost the same feature points. We prepared a large dataset as described in Sec-

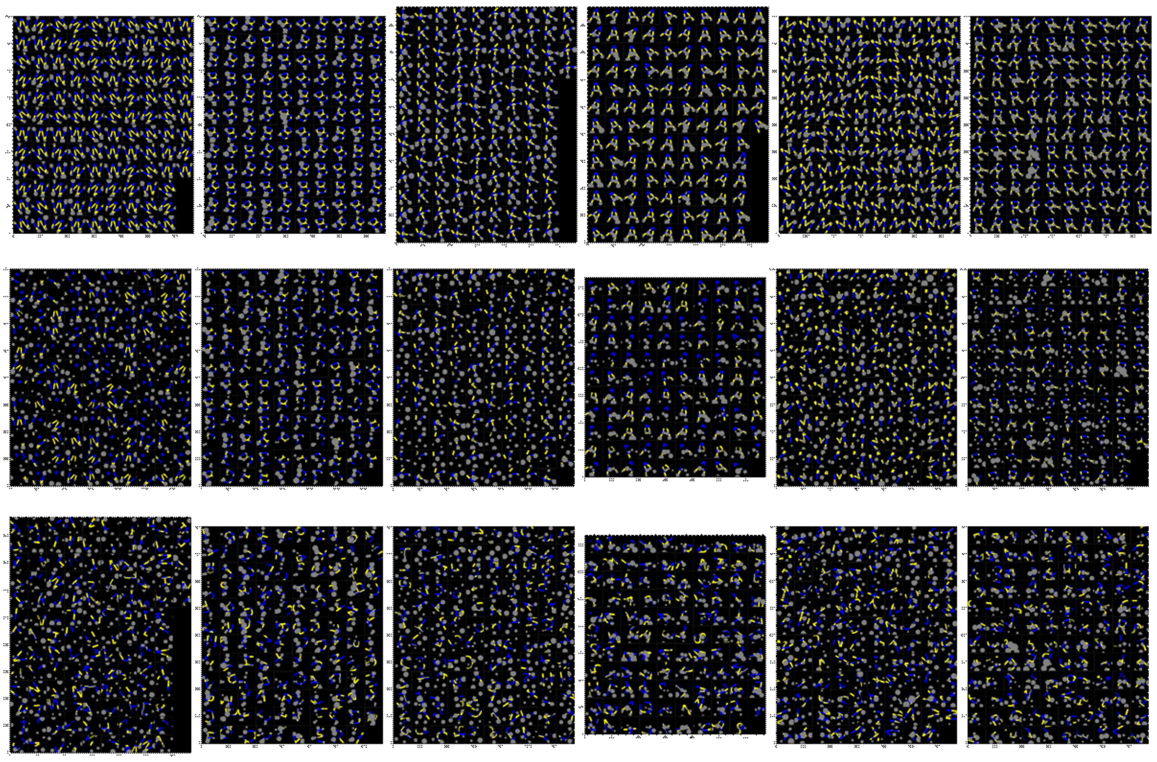


Fig. 6 Objective dataset. A couple of dozen sample characters, each of which is represented as a blue and yellow line, are shown in each image. All characters in each image represent an identical letter. The top, middle and bottom images show characters drawn as the complete, the incomplete and the dummy, respectively. The small numbers beside the axes of each figure represent the numbers of pixels.

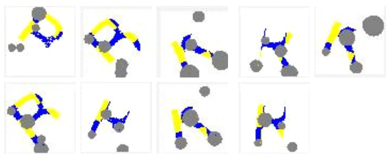


Fig. 7 Similarity of characters.

Table 1 Summary and comparison of used DCNN architectures.

referred name (varient)	#layers	#conv.*	#pool.	#bn.	#fc.	#params
AlexNet	14	5	3	3	3	13.4 M
VGGNet	22	13	5	1	3	20.5 M
GoolgeLeNet	163	70	15	73	5	14.5 M

* conv: convolution layers, bn: batch-normalization layers, pool: pooling layers, fc: full-connected layers

tion 5.4 later and a small dataset as also described in Section 5.3 later. Thirty four characters which contain alphabets are in the large data set, and six alphabets are chosen for the small dataset. As a result of the classification of the large dataset, the prediction accuracy went up to approximately 0.999. We tried to put jagged edges on characters in the small dataset to investigate how much they prevented the classification. If the jagged edges do not prevent the classification effectively, other characters in the large dataset will be less prevented since similar shaped characters are more difficult to be classified.

The small data set has 13 labels. They are labeled COMPLETE and INCOMPLETE for each of the six characters. There is one common, labeled DUMMY. We collect 120 images independently for each category pair and character. Therefore, we have 2,160 images total. We describe details of the large dataset in Section 5.4.

5.2 Network Architectures and Implementations

To evaluate Amodal CAPTCHA by deep learning, we conduct an experiment to find the optimum DCNN. The naive classification method is predicting 13 labels. However, as we can see from the figure, it is difficult to distinguish between INCOMPLETE and DUMMY. But, because the purpose of this experiment is determining whether deep learning can read CAPTCHA with amodal completion, we also evaluate the case when we merged INCOMPLETE and DUMMY into one label in the experiment.

Furthermore, we prepared three kinds of DCNNs to find the optimal one to solve this classification problem. The three are Krizhevsky's DCNN (i.e., AlexNet) as well as two different well-known architectures: VGGNet and GoogLeNet. AlexNet has 14 layers, except for pooling layers, and approximately 13 million parameters to be optimized. The size of input for DCNNs is 67×67. The output is obtained via the softmax function, which is learned to output optimal probabilities of 13 or 7 labels. We modify each architecture to fit the target datasets.

We also use batch normalization, proposed by Ioffe et al. [24], which is known to accelerate learning and to prevent over-fitting. It exploits the average and the variance over each batch. We add a batch normalization layer at the bottom of each DCNN, instead of normalizing input images.

A comparison of DCNN architectures used is shown in **Table 1**.

We implement learning, prediction and evaluation code exper-

Table 2 Comparison of accuracies and run times among DCNNs.

name	mean acc.	variance of acc.	prediction time	learning time
AlexNet	**0.990**	0.006	**0.6 ms**	2.7 ms
VGGNet	0.986	0.008	0.7 ms	3.4 ms
GoolgeLeNet	0.965	0.014	1.5 ms	8.7 ms

Table 3 Comparison the scores between 7-labels and 13-labels classification.

		COMPLETE[*]			INCOMPLETE[†]		
#labels	accuracy (σ)	P	R	F	P	R	F
7	**0.990** (0.006)	0.98	0.995	0.985	-	-	-
13	0.954 (0.010)	0.96	0.995	0.978	0.94	0.94	0.93

(*) P/R/F averaged over six characters labeled COMPLETE.
(†) P/R/F averaged over six characters labeled INCOMPLETE.

iments with a library named chainer 1 and the existing architecture, modified as follows [25]. First, we used stochastic gradient descent (SGD) with a step-size of 0.01 and a momentum of 0.9. Second, the step-size decays exponentially with a factor of 0.97 in the latter optimization steps. Finally, in the training phase, we iterate 40 epochs with the batch size of 32. We use commodity PC with a GPU (CPU: core i7-5820k, GPU: GeForce GTX 980 TI) by computational environment. We adopt a tenfold cross validation for the evaluation of the prediction result. On average, each image (67×67 pixels) was processed in approximately 2.7 ms for learning, and in approximately 0.6 ms for prediction, using AlexNet.

5.3 Experimental Results for the Small Data

The consumed time for classification was less than 1 ms for each character, or less than 30 ms for each batch. This is fast enough to process images that are given sequentially in real-time.

Generally, AlexNet is the simplest neural network architecture, and GoogLeNet has the highest classification capability for image recognition tasks.

Although, **Table 2** shows that AlexNet achieves the best accuracy [*3] (i.e., 0.990) among three types of DCNNs, its computational cost is lowest at 0.6 ms. The reason is that our classification problem is easier than other image recognition tasks, such as ImageNet. From this experiment, it can be argued that it is appropriate to apply AlexNet to this classification problem.

Table 3 shows a summary and comparison of the experimental results for 7- and 13-labeled data. Because this is multi-class classification, we measure the results by the precision/recall/F-measure (P/R/F), as well as the accuracy.

The averages are acquired over ten cross-validation (CV) trials. Table 3 shows the accuracy of 13-labels classification is lower than that of 7-labels, but the standard deviation is larger than that of 7-labels. This instability and low capability of 13-label classification arises from the difficulty of recognizing incomplete and dummy characters.

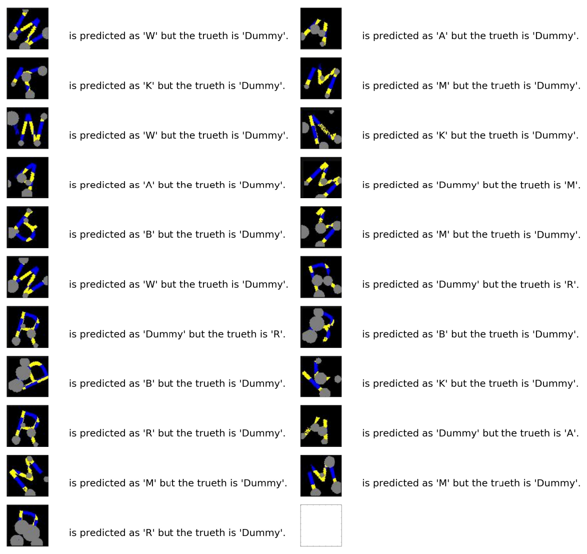
The columns marked (*) show the P/R/F averaged over six complete characters, whereas the columns marked (†) show P/R/Fs averaged only among six incomplete characters. As we can see from the table, the P/R/F of 13-labels are as high as those for 7-labels, in the case of complete characters.

[*3] The word 'accuracy' in this paper denotes the rate where labels are correctly predicted.

Table 4 P/R/Fs for each character.

	W	B	K	A	M	R
precision	0.98	0.96	0.96	0.94	0.95	0.97
recall	1.0	0.99	1.0	0.98	1.0	1.0
F-measure	0.99	0.98	0.98	0.96	0.97	0.98

The results of 7-labels classification are shown.
All values are evaluated by 10-fold CV.



is predicted as 'W' but the trueth is 'Dummy'.
is predicted as 'K' but the trueth is 'Dummy'.
is predicted as 'W' but the trueth is 'Dummy'.
is predicted as 'A' but the trueth is 'Dummy'.
is predicted as 'B' but the trueth is 'Dummy'.
is predicted as 'W' but the trueth is 'Dummy'.
is predicted as 'Dummy' but the trueth is 'R'.
is predicted as 'B' but the trueth is 'Dummy'.
is predicted as 'R' but the trueth is 'Dummy'.
is predicted as 'M' but the trueth is 'Dummy'.
is predicted as 'R' but the trueth is 'Dummy'.
is predicted as 'A' but the trueth is 'Dummy'.
is predicted as 'M' but the trueth is 'Dummy'.
is predicted as 'K' but the trueth is 'Dummy'.
is predicted as 'Dummy' but the trueth is 'M'.
is predicted as 'M' but the trueth is 'Dummy'.
is predicted as 'Dummy' but the trueth is 'R'.
is predicted as 'B' but the trueth is 'Dummy'.
is predicted as 'K' but the trueth is 'Dummy'.
is predicted as 'Dummy' but the trueth is 'A'.
is predicted as 'M' but the trueth is 'Dummy'.

Fig. 8 Examples of misrecognition by DCNNs.

Conversely, for incomplete characters, the F-measure of 13-labels is obviously lower than that of complete. The error rate of the F-measure drops less than one-third (7% to 2.2%). This result denotes that the category INCOMPLETE is difficult for one to distinguish from other categories. If we compare the accuracies of only complete characters, there is almost no difference between 13-labels and 7-labels.

Another observation made from this table is that the average recall among complete characters is almost 1.0. Thus, even if some parts of the target character are hidden, machines rarely fail to collect characters when they are completely drawn through amodal completion.

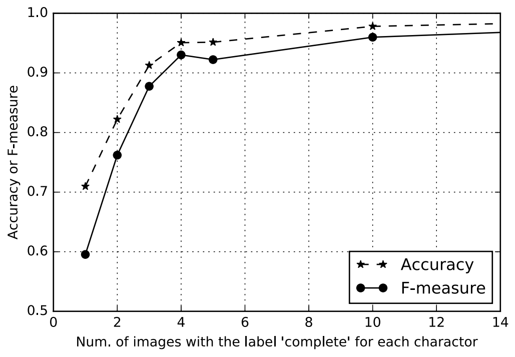
In short, DCNN can break Amodal CAPTCHA.

The P/R/F for each character are shown in **Table 4**. Additionally, precision vary depending on characters. Yet, the recall frequency is continually close to 1.0.

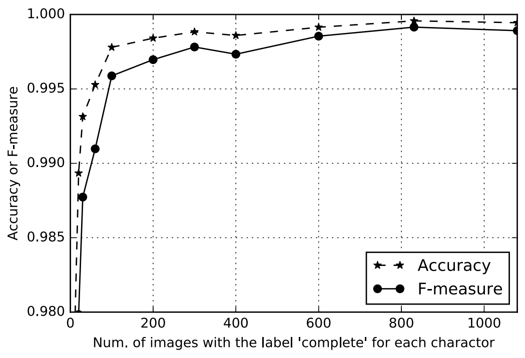
5.4 Experimental Results for the Large Data

This dataset consists of 34 characters, '0'-'9' and 'A'-'Z' (except 'C' and 'O').

We classify images into four categories: COMPLETE, INCOMPLETE-A, INCOMPLETE-B, and DUMMY. Although the overall configuration is the same as those for the small dataset, we split the category INCOMPLETE into two types: INCOMPLETE-A consists of images that humans may recognize as a correct character, whereas INCOMPLETE-B consists of images that human beings cannot recognize or may recognize as another character. The reason of the classification into four categories is the difficulty of recognizing incomplete and dummy characters on 13-label classification in Section 5.3. We confirm that humans can correctly recognize the images in INCOMPLETE-A with aftereffects over sequential frames.



(a) Accuracy and F-measure with few samples.



(b) Accuracy and F-measure with sufficiently many samples.

Fig. 9 Prediction accuracy and F-measure as a function of the number of input samples. The x-axis shows the number of images for each pair of a category and a character.

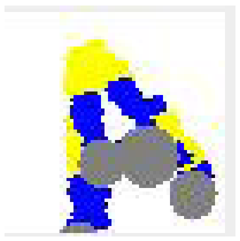


Fig. 10 fillRect.

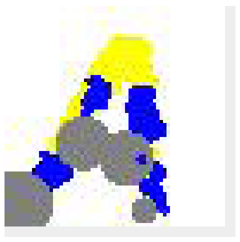


Fig. 11 fillOval.

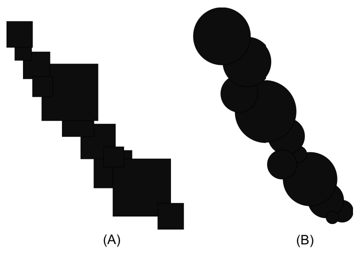


Fig. 12 Examples of the edge between fillRect and fillOval.

We collected 1,200 images, each by category and character. Because we regard DUMMY as a common label, we merge categories into two types: {COMPLETE, INCOMPLETE-A} and {INCOMPLETE-B, DUMMY}. We selected an image of the 36th frame as COMPLETE since one character is drawn with 36 frames. The middle frame is 18th frame in 36 frames. We selected an image of the 27th frame as INCOMPLETE-A since it is at the center between the middle and the last frame. It means that the image is almost complete but not complete. We also selected an image of the 9th frame as INCOMPLETE-B since it is at the center between the middle and the first frame. It means that the image is almost nothing but something drawn.

This dataset can complete a classification problem of 35 labels with 82,800 images in total.

Experimental results show that the prediction accuracy goes up to approximately 0.999 when we use all training data (**Fig. 9** (b)). The x-axis of both Fig. 9 (b) and 9 (a) represents the number of images for each category and character pair. From Fig. 9 (a), we know that both accuracy and F-measure go beyond 0.9 when the number of images is greater than four. This means that, for each character, only four images labeled COMPLETE are needed to achieve accuracy and an F-measure of 0.9.

6. Effect of Jagged-lined Amodal CAPTCHA as Ergonomic Design

Our method is completely broken by deep learning, as mentioned in Section 5. However, we found that edges of characters are mainly used for analyzing samples in deep learning procedures. Therefore, in this section, we add a new algorithm to our CAPTCHA method to break the edges. In this algorithm, a character consists of a cluster of random size objects so that edges of the character appear as jagged lines. In our method, each character is written as an aggregate of pixels. However, in our new algorithm, each character is written as an aggregate of objects consisting of many pixels square. There are two kinds of objects in our experiments. One is a square, drawn by "fillRect" and the other is a circle, drawn by "fillOval" using Java. **Figures 10** and **11** show the obtained "fillRect" and "fillOval".

Examples of jagged line are shown in **Fig. 12**. The edge of a character by "fillRect" is drawn as (A). It is made by the collection of many rectangles of different sizes. The edge of a character by "fillOval" is drawn as (B). It is made by the collection of many circles of different sizes.

6.1 Purpose of the Experiments

We found that knowledge of character edges is very important for deep learning recognition. Therefore, by making jagged edges of CAPTCHA, we should not decrease the visual performance of humans. First, we investigate the differences among jagged line types, including how jagged they can be while maintaining human recognition. Second, we investigate the influence of jagged lines on deep learning recognition.

6.2 Preparation of Subjective Parameters

Characters with jagged edges are created as a cluster of objects, and the cluster consists of many squares or circles. We deemed

Table 5 Evaluation of visual performance by human beings.

Examinee	age	sex	fillRect	fillOval
A	21	M	18	18
B	21	M	18	16
C	21	F	18	18
D	21	M	12	18
E	21	M	10	6
F	21	M	11	15
G	21	M	13	15
H	21	M	11	9
I	19	M	9	15
J	19	M	13	11

the smallest size of an object to be three pixels by three pixels, because zero pixel objects do not appear on the screen, and one- or two-pixel objects are too small to make a jagged line. We also deemed the maximum object size to be eighteen pixels by eighteen pixels, because we feel that larger characters are hard to recognize. Moreover, the object size is incremented by two pixels each. Finally, as the size of each object is from three to eighteen-pixel square, the rate of the decision of the size is the same. We call this object a "big size object." There is also a "small size object" consisting of one pixel. One character consists of 3/40 big size objects and 37/40 small size objects, because too many big size objects only thicken the character and erase the jagged lines. The objective of our research is learning whether character recognition with deep learning is impeded by jagged-edged characters, and by how much.

6.3 Visual Performance for Human Beings

We evaluated our jagged-edge algorithm by conducting an experiment with ten examinees. All examinees are healthy students between 19 and 21 years old and their sexes and ages were added in **Table 5**. Their ages are those at April 1st, 2017 out of consideration of their privacy. The experiments were done in different days. The resolution of the monitor for the experiments by examinee F, G, H, I and J is 226 dpi (dot/inch), and the panel of the monitor is 2,304 by 1,440 square dots. Examinee A, B, C, D and E used their own PCs. A: 13.3 inch, 1,366 by 768 dots, 117.8 dpi; B: 11.6 inch, 1,366 by 768 dots, 135.1 dpi; C: 13.3 inch, 1,366 by 768 dots, 117.8 dpi; D: 13.3 inch, 2,560 by 1,440 dots, 220.8 dpi; E: 11.6 inch, 1,366 by 768 dots, 135.1 dpi. One character is drawn with 36 frames and the drawing speed is 60 fps (frame/second).

In this evaluation, there are two characters on the screen. The characters are moving by our CAPTCHA method, as mentioned in Section 3. The maximum object size is randomly set at the initial appearance. Examinees choose the one with better visibility. The size of the character not chosen is then changed randomly. The maximum object size is incremented by two pixels so that odd number sizes only appear in the random case. After that, examinees choose one of two characters again. The enlargements and choices are repeated until the maximum object size of one of the character is incremented to eighteen. The last chosen character has the best maximum object size for the examinees, and the size is recorded as a score. Results of the evaluation are shown in Table 5. In both "fillRect" and "fillOval", eighteen-pixel square is the best recognized maximum object size for some examinees, but not for others. However, the maximum object size is randomly chosen, and this evaluation stops when the maximum object size of one of the characters is incremented to eighteen. Therefore, all sizes do not always appear in an experiment. However, almost all of the examinees do not possess the scrutiny to recognize a large character with jagged lines, because lines appear smoother when the maximum object size is smaller. Results show that some examinees have the same subjective size judgment that we do.

6.4 Preparation of Data

We prepared two kinds of data sets, which were drawn with "fillRect" and "fillOval." The maximum object size was set to eighteen for both data sets. We used the same DCNN models as those in Section 5.3. The letter "A" was chosen as a representative of characters for the data sets. There were also three types of patterns: 'complete,' 'incomplete,' and 'dummy,' which are the same as the types from our previous research. We collected 600 sheets of images for each type. Via our previous experiment, AlexNet, with two labels, was chosen for the DCNN model, because it is the most accurate and the fastest.

6.5 Results of Experiments

The left column in **Tables 6**, **7** and **8** shows the number of loops in learning phase. "TRAIN" represents the accuracy of training samples, and "TEST" represents the accuracy of test samples. The attribute, 'mean,' is average; and 'std' is standard deviation. Results of "fillRect" and "fillOval" are shown in Table 7 and Table 8. The average of results of "original" is shown in Table 6. We tested ten times for results without jagged edges. The average of results of "original" is shown in Table 6, and results of "fillRect" are shown in Table 7. It appears that the numbers in the tables have certain differences between "fillRect" (or "fillOval") and "original". For instance, the mean of the accuracy in the initial loop in TEST decreases from 0.907 to 0.893 when jagged edges by "fillOval" are applied. However, the corresponding standard deviations are much larger than the differences between those numbers as the last columns in the tables. The tables rather show that the difference of the accuracies between "fillRect" (or "fillOval") and "original" for each loop is smaller than the standard errors of the means, which are approximately equal to the standard deviations divided by the square root of the number of trials. Thus, as the results of the experiments, the jagged edges do not significantly affect the recognition rate by the CNN in statistical meaning.

6.6 Consideration

The results shown in Section 6.5 indicate that character edges are not important for recognition by CNNs, just as they do not influence visual performance in humans. In addition, for any character edge, the accuracy finally exceeds 99%. This fact suggests that CAPTCHA can be fully broken by independent malware that we regard as an enemy. The decrease of the accuracy at Loop 00 is not so effective to prevent attackers from training since they can use their own high-spec computers for the training. Someone may think that it is effective to prevent independent malware from training since malware is often executed on victims' low-spec computers. However, independent malware cannot learn from

Table 6 Result by original drawing.

	mean		std	
Loop	TRAIN	TEST	TRAIN	TEST
0	0.714	0.907	0.020	0.046
1	0.933	0.960	0.013	0.022
2	0.962	0.975	0.007	0.010
3	0.966	0.977	0.014	0.007
4	0.978	0.982	0.005	0.006
5	0.981	0.985	0.003	0.011
6	0.985	0.987	0.005	0.006
7	0.986	0.989	0.004	0.006
8	0.990	0.986	0.003	0.012
9	0.988	0.990	0.004	0.008
10	0.996	0.994	0.002	0.004
11	0.998	0.995	0.001	0.003
12	0.998	0.995	0.001	0.004
13	0.998	0.995	0.001	0.004
14	0.999	0.995	0.001	0.003
15	0.999	0.996	0.001	0.003
16	0.999	0.996	0.001	0.004
17	0.999	0.996	0.000	0.003
18	1.000	0.995	0.000	0.004
19	1.000	0.995	0.000	0.003

Table 7 Result by fillRect.

	mean		std	
Loop	TRAIN	TEST	TRAIN	TEST
0	0.744	0.930	0.020	0.065
1	0.935	0.957	0.012	0.041
2	0.950	0.959	0.013	0.038
3	0.961	0.965	0.006	0.028
4	0.965	0.967	0.009	0.032
5	0.969	0.969	0.007	0.029
6	0.972	0.967	0.010	0.030
7	0.972	0.965	0.012	0.045
8	0.973	0.975	0.010	0.031
9	0.979	0.983	0.009	0.024
10	0.992	0.991	0.003	0.012
11	0.995	0.991	0.002	0.011
12	0.995	0.992	0.003	0.010
13	0.997	0.993	0.002	0.009
14	0.997	0.994	0.001	0.007
15	0.998	0.994	0.001	0.007
16	0.999	0.996	0.001	0.006
17	0.999	0.994	0.001	0.007
18	0.999	0.996	0.001	0.005
19	0.999	0.997	0.001	0.004

Table 8 Result by fillOval.

	mean		std	
Loop	TRAIN	TEST	TRAIN	TEST
0	0.712	0.893	0.019	0.074
1	0.906	0.934	0.013	0.051
2	0.940	0.945	0.015	0.040
3	0.958	0.966	0.007	0.025
4	0.962	0.972	0.010	0.019
5	0.972	0.980	0.008	0.011
6	0.979	0.980	0.007	0.017
7	0.982	0.986	0.007	0.011
8	0.989	0.986	0.005	0.009
9	0.989	0.990	0.006	0.012
10	0.994	0.996	0.007	0.004
11	0.997	0.995	0.001	0.004
12	0.997	0.996	0.001	0.004
13	0.998	0.997	0.001	0.004
14	0.998	0.996	0.001	0.004
15	0.998	0.997	0.001	0.004
16	0.999	0.995	0.001	0.005
17	0.999	0.997	0.001	0.004
18	0.999	0.997	0.000	0.005
19	0.999	0.996	0.000	0.004

images since it cannot put labels on the images by itself, as mentioned in Section 1. On the other hand, directly controlled malware can always renew its models for attacks by updating them from outside.

We prepare character images in experiments both in this paper and in our previous work, because we should set images of characters to DCNNs for comparison. Alternatively, computers prepare the images from frames of a video in actual attacks, and there are many frame combinations, because computers do not know when the target character appears. Furthermore, highly specialized computers are used for the experiments, as shown in Section 6. Yet, computers which are used for actual attacks are rarely highly specialized; bots are usually housed on ordinary computers.

7. Evaluation of DCNN learned from Non-shielded Data

Experiments thus far have used datasets consisting of images of characters with attached shields for both training and testing. However, in situations where CAPTCHAs are actually applied, attackers are less likely to have a priori information about the shielding of the image based on amodal completion. Although all experiments so far show negative results with respect to the effects of ergonomic designs against DCNNs, they are unfair for the CAPTCHAs since DCNNs are learned from the images of characters with shielding. This assumption that images with shielding is in the training data as well as the validation data means that attackers have a priori information about the positions of the shields based on amodal completion. Therefore, in order to measure the ability of DCNN against Amodal CAPTCHA in more realistic situations than in the previous experiments, we used training data consisting of images of characters without shielding. The experiments in this section are disadvantageous to DCNNs, since the locations or even the existence or absence of shields are not known by the machines.

7.1 Preparation of Data

We conducted an experiment with CAPTCHA without occlusion as training data and CAPTCHA with shielding as test data. We acquired the same number of images and the same characters as CAPTCHA without shielding as a small data set described in Section 5.1. An acquired image is shown in the **Figs. 13–18**. We used AlexNet, the same as for the small data set in Section 5.1. We captured each of {'A,' 'B,' 'K,' 'R,' 'M,' and 'W'} CAPTCHAs with shields; CAPTCHA without shields each acquired 120 images, totaling 1,420 images. Thus, this experiment is a six-image classification problem. Other hyperparameters such as the learning rate are also the same as in Section 5.1.

7.2 Results of Experiments

We carried out ten trials, the test rates of which are all over 95%. The final accuracy amounted to 0.990. This was exactly the same number as the result of Table 4 of Section 5.3. DCNN was able to break CAPTCHA despite being in a disadvantageous situation. In other words, it was able for the DCNN to recognize Amodal CAPTCHA precisely even if learning was carried out

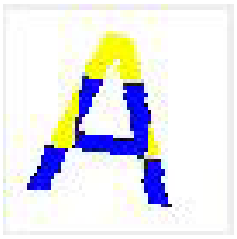


Fig. 13 Original A.

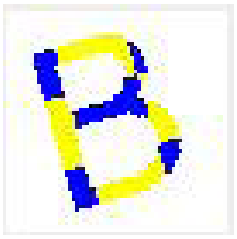


Fig. 14 Original B.

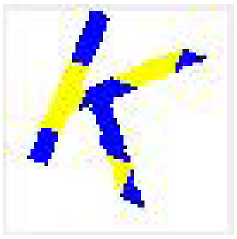


Fig. 15 Original K.

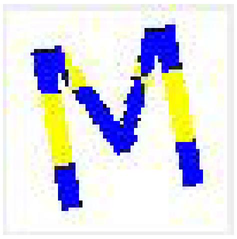


Fig. 16 Original M.

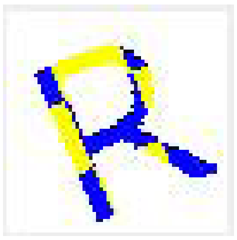


Fig. 17 Original R.

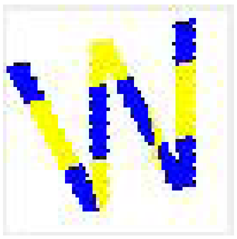


Fig. 18 Original W.

Table 9 Average of the evaluation of original and Amodal.

Loop	TRAIN	TEST	P	R	F
0	1.0	0.977	0.912	0.887	0.890
1	1.0	0.960	0.952	0.952	0.950
2	1.0	0.975	0.969	0.964	0.966
3	1.0	0.977	0.971	0.968	0.969
4	1.0	0.982	0.978	0.975	0.976
5	1.0	0.985	0.981	0.980	0.980
6	1.0	0.987	0.984	0.982	0.982
7	1.0	0.989	0.986	0.984	0.985
8	1.0	0.986	0.981	0.985	0.982
9	1.0	0.990	0.989	0.984	0.986

only with images without shielding. When we used fillRect or fillOval, DCNN was still able to recognize characters precisely even if the edges of characters were randomly changed. From this, we must conclude that the collapse of some edges due to shielding will not affect the recognition ability of DCNNs. Although this result is surprising, it is estimated that the generalization ability of DCNNs worked well in this case. Since the topological shape of the character does not change on the whole even if a shield is on it, its silhouette does not significantly destroyed.

8. Conclusion

Our Amodal CAPTCHA had been effective against threats before DCNN became famous in image classification. First, we explained the difference of Amodal CAPTCHA from other normal text-based CAPTCHA in terms of visibility for human beings. Next, we evaluated Amodal CAPTCHA by DCNN. As a result, we learned that DCNN can almost perfectly break Amodal CAPTCHA without emulating amodal completion. DCNN recognized characters of Amodal CAPTCHA at better than 95% in a small dataset. The rate reached to 99% in a large data set. Moreover, we found that only four images were required for learning to maintain accuracy.

We tried to decrease the accuracy of DCNN to Amodal CAPTCHA without applying distortion or noise. We applied ergonomically jagged edges to characters of Amodal CAPTCHA by drawing various rectangles and circles. This succeeded in decreasing the accuracy of several initial loops of DCNN in training, but the accuracy soon recovered to the values which were similar to those of the previous experiment. When the jagged edge was created by drawn circles, the accuracy of the first loop was lower than before. However, attackers of CAPTCHA can let DCNN learn many times with their own high-spec machines so that the decrease does not affect the prevention of attacks.

We also confirmed that DCNN can recognize characters not by positions of shields but by the shape of characters. We let DCNN learn from characters without shields and tested it using characters with shields. DCNN almost perfectly classified the characters in this experiment so that we concluded that the position of shields does not affect the character recognition by DCNN.

Our results in this paper show that any ergonomic effects such as amodal completion and jagged edges are no longer countermeasures against character recognition by computers. We think that it is necessary to apply image processing specialized for deep learning such as the adversarial examples of Goodfellow et al. [26]. Adversarial examples are images which contain perturbations calculated from the images in training. It is known that adversarial examples can make DCNN misclassify images. In future, we will try to apply the adversarial examples to our Amodal CAPTCHA and will evaluate the rate of misclassification by DCNN.

References

[1] Turing, M.A.: Computing Machinery and Intelligence, *Journal of the Mind Association* (*MIND*), Vol.LIX, No.236, pp.433–460 (1950).

[2] Coates, A.L., Baird, H.S. and Fatenan, R.J.: Pessimal print: A reverse Turing test, *Proc. 6th International Conference on Document Analysis and Recognition*, pp.1154–1158 (2001).

[3] Chellapilla, K. and Simard, P.Y.: Using Machine Learning to Break Visual Human Interaction Proofs (HIPs), *Proc. Advances in Neural Information Processing Systems 17*, pp.265–272 (2004).

[4] George, D., Lehrach, W., Kansky, K., et al.: A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs, *Science*, Vol.358, No.6363 (2017).

[5] PWNtcha, available from ⟨http://caca.zoy.org/wiki/PWNtcha⟩.

[6] Mori, G. and Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA, *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003*, Vol.1, pp.I-134–I-141 (2003).

[7] NuCAPTCHA, available from ⟨http://www.nucaptcha.com/⟩.

[8] Chellapilla, K., Larson, K., Simard, P., et al.: Computers Beat Humans at Single Character Recognition in Reading Based Human Interaction Proofs (HIPs), *2nd Conference on Email and Anti-Spam CEAS* (2005).

[9] Fukushima, K.: Recognition of partly occluded patterns: A neural network model. *Biological Cybernetics*, Vol.84, No.4, pp.251–259 (2001).

[10] LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P.: Gradient-based learning applied to document recognition, *Proc. IEEE* (1998).

[11] Krizhevsky, A., Sutskever, I. and Hinton, E.G.: ImageNet Classification with Deep Convolutional Neural Networks, *Advances in Neural Information Processing Systems*, Vol.25, pp.1097–1105 (2012).

[12] Simonyan, K. and Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recoginition, *Intl. Conf. Learning Representations* (*ICLR*) (2015).

[13] Szegedy, C., Liu, W., Jia, Y., et al.: Going Deeper with Convolutions, *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp.1–9 (2015).

[14] KarSimonyan, S. and Zisserman, A.: Very Deep Convolutional Networks for Large-Scale Image Recognition, *Intl. Conf. Learning Representations* (*ICLR*) (2015).

[15] LeCun, Y.: Generalization and Network Design Strategies, *Technical Report, CRG-TR-89-4* (1989).

[16] LeCun, Y., Boser, B., Denker, S.J., et al.: Backpropagation applied to handwritten zip code recognition, *Neural Computation 1*, pp.541–551 (1989).

[17] Mori, T., Uda, R. and Kikuchi, M.: Proposal of Movie CAPTCHA Method Using Amodal Completion, *Proc. IEEE/IPSJ 12th International Symposium on Applications and the Internet* (*SAINT2012*), pp.11–18 (2012).

[18] Masnou, S. and Morel, M.J.: Level lines based disocclusion, *Proc. 1998 International Conference on Image Processing, ICIP 98*, p.7 (1998).

[19] Kanizsa, G.: Original title: Grammatica del vedere, *La Grammaire du Voir. Diderot* (1996).

[20] Sawada, K. and Uda, R.: Effective CAPTCHA with Amodal Completion and Aftereffects, *Proc. 10th International Conference of Uniquitous Information Management and Communication* (*IMCOM'16*), Article No.53 (2016).

[21] Azakami, T. and Uda, U.: Effective CAPTCHA with Amodal Completion and Aftereffects by Complementary Colors and Difference of Luminance, *Proc. 30th IEEE International Conference on Advanced Information Networking and Applications Workshops* (*WAINA-2016*), pp.232–237 (2016).

[22] Sivakorn, S., Polakis, I. and Keromytis, D.A.: I am Robot: (Deep) Learning to Break Semantic Image CAPTCHAs, *Security and Privacy* (*EuroS&P*), *2016 IEEE European*, pp.21–24 (2016).

[23] Lv, G.: Recognition of Multi-Fontstyle Characters Based on Convolutional Neural Network, *4th International Symposium on Computational Intelligence and Design* (*ISCID 2011*), pp.28–30 (2011).

[24] Ioffe, S. and Szegedy, C.: Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, *Proc. 32nd International Conference on Machine Learning, JMLR*, Vol.37, pp.448–456 (2015).

[25] Tokui, S., Oono, K., Hido, S., et al.: Chainer: A Next-Generation Open Source Framework for Deep Learning, *Proc. Workshop on Machine Learning Systems* (*LearningSys*) *in the 29th Annual Conference on Neural Information Processing Systems* (*NIPS*) (2015).

[26] Goodfellow, J.I., Shlens, J. and Szegedy, C.: Explaining and Harnessing Adversarial Examples, *Published as a Conference Paper at ICLR* (2015).

[27] Azakami, T., Shibata, C. and Uda, R.: Challenge to Impede Deep Learning against CAPTCHA with Ergonomic Design, *Proc. IEEE 41st Computer Software and Applications Conference* (*COMPSAC*), pp.637–642 (2017).

[28] Azakami, T., Shibata, C. and Uda, R.: Challenge of Deep Learning against CAPTCHA with Amodal Completion and Aftereffects by Colors, *Proc. 19th International Conference on Network-Based Information Systems* (*NBiS*), pp.127–134 (2016).

Tomoka Azakami is currently a student of Tokyo University of Technology graduate school since 2017. Her research interest covers fields of information security and artificial intelligence. She received Excellent Paper Award and Excellent Presentation Award at IPSJ Multimedia, Distributed, Cooperative and Mobile Symposium 2017 (DICOMO 2017).

Chihiro Shibata is a senior assistant professor of computer science in the Graduate School of Bionics, Computer and Media Sciences at Tokyo University of Technology. Her research areas are the applications of machine learning such as natural language processing and image recognition and statistical learning theories.

Ryuya Uda received his B.E., M.E. and Ph.D. degrees in Engineering from Keio University in 1998, 2000 and 2002 respectively. In 2002, he was a researcher at Katayanagi Advanced Research Laboratory, Tokyo University of Technology, Japan. Currently, he is a senior assistant professor at School of Computer Science, Tokyo University of Technology, Japan. His research interest covers fields of information security, including network security, digital forensics, etc. He received the Best Student Paper Award at IFIP SEC 2002.