IET Research Journals

Mixaptcha: A CAPTCHA Based On Distinguishing Overlaid Images

ISSN 1751-8644
doi: 0000000000
www.ietdl.org

Siyu Sun[1], Zonghan Yang[1], Yuheng Zhang[1]

[1] Zhiyuan College, Shanghai Jiao Tong Univeristy, Minhang District, Shanghai 200240, China
These authors contribute to the article equally.
** E-mail: {sunsiyu, yveh1999, fstqwq}@sjtu.edu.cn*

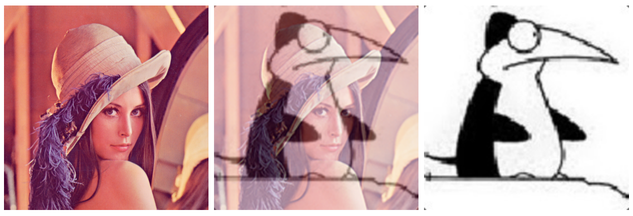
Abstract: We present a new CAPTCHA that requires users to distinguish between single images and mixtures of two images. We also propose some techniques to select and generate proper test sets. The test set can be automatically generated by our offline method, which requires unlabeled image sources only, while human users over-perform algorithms with a significant gap. Compared to the most famous CAPTCHAs, the test set can be automatically generated by our offline method that requires unlabeled image source only while human users over-perform algorithms with a significant gap. The verification process is also more pleasant than the most popular CAPTCHA system.

1 Introduction

With an increasing need for bot detection as the network crawlers deluge, CAPTCHA, known as Completely Automated Public Turing test to tell Computers and Humans Apart, was created by Luis von Ahn and Manuel Blum in 2000 [1] [2]. CAPTCHA has been proven efficient in reducing the chance of spamming and attacks, including password cracking and Denial-of-service attack, and thus plays a crucial role in clean and secure web applications. The most famous and most traditional form of CAPTCHA is text recognition, which presents distorted English letters and Arabic numbers hidden in the noise. There are many three-party CAPTCHA providers offer some new types of CAPTCHAs, including reCAPTCHA [3], "What's Up" CAPTCHA [4].

However, there are many issues with popular CAPTCHAs. The most famous text-based CAPTCHAs are intensively studied [5] [6] and can be easily hacked by Optical Character Recognition with help of modern methods [7] [8]. The most widely-applied reCAPTCHA is more effective, which, however, is a commercial project and requires labeled images of a large number, and there are also successful attacks to reCAPTCHA.

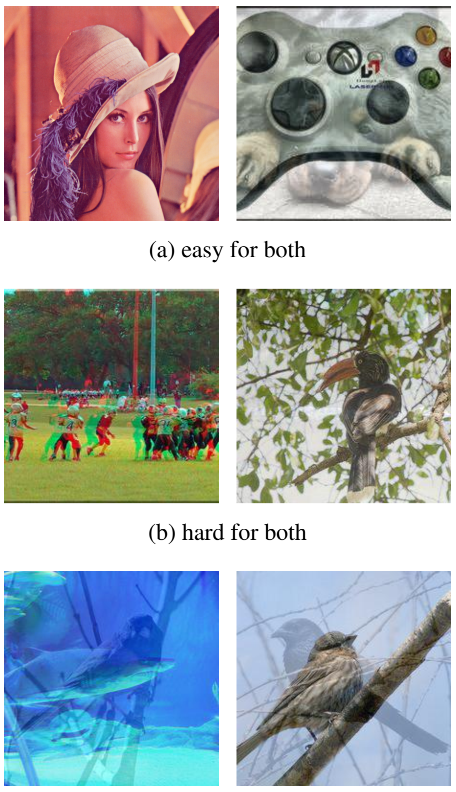
To fix these issues, we present a new CAPTCHA method called Mixaptcha, which requires users to distinguish if the presented images are natural or mixed up by two. The mixing procedure is simply a linear combination illustrated in Figure 1.



(a) original (b) mixed (c) original

Fig. 1: Illustration of mixture by two images.

The difficulty of distinguishing mixed images varies from the images chosen. There are mainly three levels of difficulty, listed below. Figure 2 shows examples of them, respectively.



(a) easy for both

(b) hard for both

(c) easy for human only

Fig. 2: Example of combinations varied in difficulty.

(a) Some combinations of images can be easily detected by both humans and computers. Typically, they consist of single familiar objects, for example, human faces — one most studied case in object recognition.

(b) Some combinations of images can be recognized by neither human nor computer. One cause for the phenomenon is that the images are noisy or containing misleading information, like images with visual glitches. Another reason is that some combinations are too shallow to be distinguished in the mixture: the second example is a mixture, but no tester — human or machine — gives a positive answer.

(c) Some combinations are easy for humans but very difficult for the computer since humans can extract semantic information from images more effectively. The first image shown is one instance: water ripples are ordinary underwater, therefore experienced machines — like pre-trained networks — may consider the overlaid layer as ripples. But it's easy for a human to point out the bird on the tree branch since the image is unnatural enough. The second image is a mixture of two birds. Expert machines may recognize that there are two birds on tree branches, in which case machines tend to give a negative answer. Nevertheless, a human can point out the picture is not natural and will give positive feedback.

The last case is capable of a CAPTCHA test set. To efficiently filter suitable images, we purpose an iterative adversarial method that sieves easy cases away without human intervention, while the generating process can be done offline using some image data sets. On the sets generated, the accuracy of an expert machine is less than 0.75, from which we can claim that it understood little about mixing in practice. In comparison, a typical human accuracy on the same set is more than 0.95, which shows a performance gap, and therefore it's a proper task to distinguish bots.

2 Related Works

2.1 Text-based CAPTCHA

The oldest type of CAPTCHA like Figure 3 is based on text recognition, which requires users to type down the text presented in the image. The direction of text-based CAPTCHA is simple, and it saves network traffic since low quality is required for the images. It also enables people who do not have comprehensive English ability to finish the verification process.

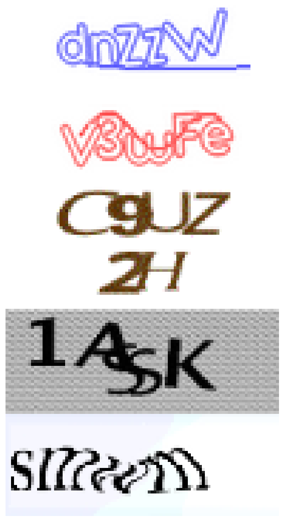


Fig. 3: Text-based CAPTCHAs. Sampled from websites include Apple, Baidu and Wikipedia. [9]

The drawback of text-based CAPTCHA is clear: the knowledge conveyed in a random combination of characters is poor. As recent paper [8] mentioned, the best method using Cycle-Consistent Generative Adversarial Network achieves an average accuracy better than 0.9 on the top-visited websites. Also, when more distortion and noise are applied to lower machine accuracy, human accuracy significantly drops since it's not only a hard problem for computers. Therefore, most modern websites abandon text-based CAPTCHA nowadays.

2.2 "What's Up" CAPTCHA

To enhance CAPTCHA, one way is to emphasize the semantic understanding gap between human and computer. One solution is to use images. "What's Up" CAPTCHA is one of the popular CAPTCHA nowadays and was invented by a small group of Google researchers [4]. The main challenge shown in Figure 4 used is asking users to drag the image provided and lift it to an up-straight state. It's more convenient and more fun for users to finish the verification process, and it requires attackers to write a more complex automated program to interact with the interface since the probability of a no-effort attack is very low, and it requires precise mouse control. It's one widely-adopted method currently for these advantages over text-based CAPTCHA.

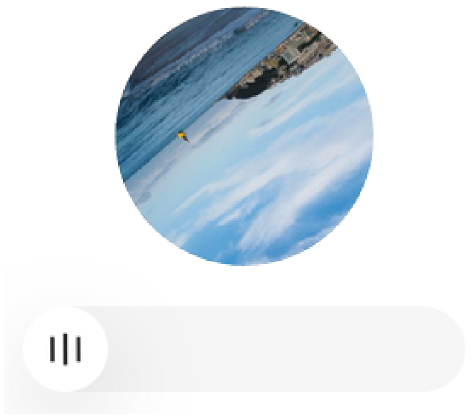


Fig. 4: Sample of "What's Up" CAPTCHA challenge implemented by Baidu.

One noticeable drawback for "What's Up" CAPTCHA is that the image used for orientation should be carefully selected and labeled by human, which is heavy labor work. In fact, by counting the popular "What's Up" CAPTCHA applications, it's an averagely 20 request to acquire the same image. By Birthday Paradox, we can conclude that only a few (≤ 1000) images are in use in practice, which significantly increases the chance of being hacked. There are also successful general hacking attempts. The very surprising fact is that an easy 360-classification approach can solve the puzzle with high probability [10].

2.3 reCAPTCHA

reCAPTCHA was introduced by CyLab of Carnegie Mellon University and von Ahn. It was initially intended to use the scanned book as the source of word input and then build a digital library at the same time as avoiding spam. Google acquired reCAPTCHA in 2009 and then pivoted it away from the original purpose and made it a commercial project towards user experience and crowd-sourcing revenue. Along with other techniques to disrupt users as little as possible, it currently uses object detection as the main challenge that requires users to find out semantic targets of one type in complex conditions, such as recognizing traffic lights from street views. Figure 5 shows a sample of reCAPTCHA that requires finding traffic lights.

Attacking reCAPTCHA requires more advanced techniques and are usually failed, yet there are deep learning and reinforcement learning techniques that can still fairly finish the job [11] [12]. Another issue for reCAPTCHA is that it needs labeled images of prodigious numbers. To fulfill such a need, reCAPTCHA uses user-based statistics to label the images. Due to Google's monopoly, it can collect billions of clicks per day to support the system and a huge revenue by the crowd-sourcing labeling and advanced version fees [13] [14]. For individual users who wish to reproduce reCAPTCHA, it's not likely to succeed with a small group of users and a limited size of the database. Therefore, users are forced to pay (at least by human resource) for the services provided by Google to get the permission of using the version of reCAPTCHA.

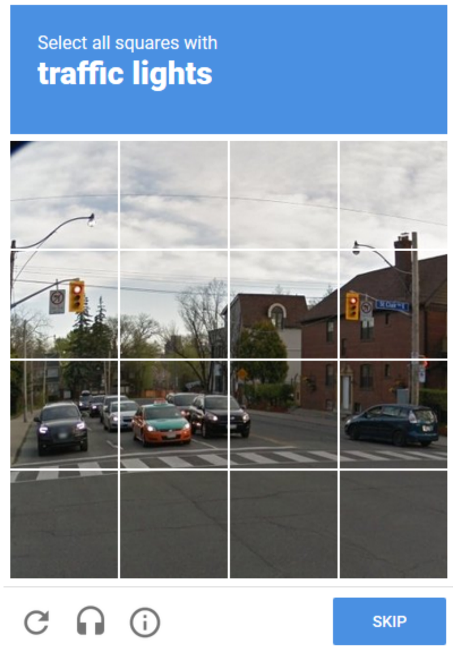


Fig. 5: Sample of reCAPTCHA challenge.

3 Method and experiments

For a base image set, the basic random method of generating test set is straightforward. Suppose the base data set is large enough, we can simply sample one image to be an 'original' image in the test set, or sample two images and mix them up to be a 'mixed' image. Mixed images can be simply linear combination of two chosen images, with some specific $\mathrm{mixrate}$. Formally, let two images $P_{\mathrm{original}_1}, P_{\mathrm{original}_2} \in (\mathbb{R}^3)^{n\times n}$ represented in RGB, the mixture of them is calculated as

$$P_{\mathrm{mix}} = P_{\mathrm{original}_1} \cdot \mathrm{rate}_{\mathrm{mix}} + P_{\mathrm{original}_2} \cdot (1 - \mathrm{rate}_{\mathrm{mix}}).$$

By the equation above, we can also describe the hardness of the task from a high-level perspective: the natural, single images can be viewed as many small clusters in the image space. They are very sparse by the fact that a random value image is almost surely a noise image. Machines learn the distribution by a small subset of the whole large image space, and thus it's hard for machines to imitate the sensitivity of humans by only a few shots of the sample. Taking a weighted average of images can be viewed as pick a point on the two-point-line with some specific parameter, which will lead the mixture to fall outside the original clusters determined by humans. Learning the exact border of every cluster is quite hard for machines.

However, directly using the randomly generated set may lead to high machine accuracy as we can see for the following experiments, since it may contain too many easy tasks for machines. Methods to reduce machine accuracy, therefore, are considered to be on demand in order to fix the issue.

3.1 *Image preprocessing*

To detect mixture, the adversary may learn pixel distributions difference that is knowledge-free since the mixtures' lightnesses will be more concentrated and the contrasts of mixed images will reduce under the assumption that images are chosen randomly. There are three aspects in consideration:

1. Lightness: Mixed images' lightnesses are more concentrated around expectation.
2. Contrast: Mixed images' contrasts are likely to be lower than original.
3. Noise: There are noise caused by image compression and film texture. Mixed images' noise has a more average distribution.

The solution we use is quite simple. We simply adjust the lightness of all images by random values, as well as increasing the contrasts by random values using Contrast Limited Adaptive Histogram Equalization algorithm[15], then use JPEG compression [16] to reduce the size of the image and lose fine-grained texture. The process of random adjustment can be viewed as one step on the Markov Chains of lightness or contrast, whose initial configurations can be determined by the distribution of the data set, and therefore the distribution will be mixed up by such operation. Since no extra information is provided in the preprocessing process, the result is at least as good as those not preprocessed. Algorithm 1 describes the procedure using pseudocode.

Algorithm 1 Algorithm for image preprocessing

```
Input: raw image in RGB color space
Output: processed image in RGB color space
1: procedure PREPROCESS(RGBImage)
2:     $(L, A, B) \leftarrow \mathtt{RGBtoLAB}(\mathrm{RGBImage})$
3:     $L \leftarrow L + \mathtt{Random}(-\mathrm{Max}\Delta L, \mathrm{Max}\Delta L)$
4:     $L \leftarrow \mathtt{CLAHE}(L, \mathtt{clipLimit} = \mathtt{Random}(\cdots))$
5:     return $\mathtt{JPEGCompression}(\mathtt{LABtoRGB}((L, A, B)))$
6: end procedure
```

3.2 *Adversarial iterative sieving*

The random selection and combination of images from a large scale image set will lead to a bad result since it contains many images having the same difficulty on machines and humans, i.e., the cases (a) (b) mentioned in Figure 2. Therefore, a proper selection of test sets should be applied.

The selection process should be automated to avoid wasting expensive human resources. One idea is to generate base data set using Generative Adversarial Networks [17], to decrease machine accuracy. However, there should not be a good definition of 'natural' original images for machines. Humans should exclusively hold the 'natural' standard. Otherwise, the hard problem we are based on is not hard anymore. Therefore, instead of generating a natural base data set, we propose a method to filter the proper CAPTCHA set based on a given base image set by iteratively deleting the easily detected images.

The 'easily detected' standard is based on the imitated adversary we use, ResNet [18]. The neural network has a very strong expression ability and avoids heavy feature engineering, though there are problems in training, including over-fitting and gradient explosion. ResNet solves the above problems by applying the residual unit and proved to be an efficient method to achieve the best accuracy in many computer vision tasks. To imitate possible attackers, we use a linear classification layer that follows ResNet-152.

Although it may be doubted that the attacker we consider is too simple, it should be noticed that many attackers in practice use small models in order to save the cost of hacking CAPTCHA, whose abilities are far poorer than the $117,364,032$-parameter model that requires expensive computing resources. Therefore, ResNet-152 is actually a very strong standard for machines distinguishing the CAPTCHA.

We could also prove the extraordinary capacity of ResNet by experiments. We use a subset of images in the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) [19] to generate a mixture of pictures in a random fashion. The training set contains $500,000$ images, and a pre-trained model from PyTorch [20] which was trained on the same set is taken for the initial parameter configuration of ResNet.

To compare with, we invited 5 candidates as human testers to view a total number of $1,000$ images using a small program shown in Figure 6.

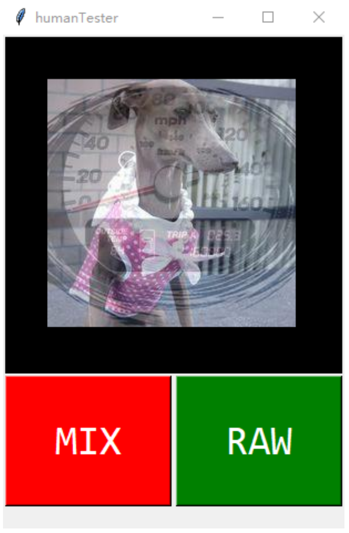


Fig. 6: Human testing GUI program.

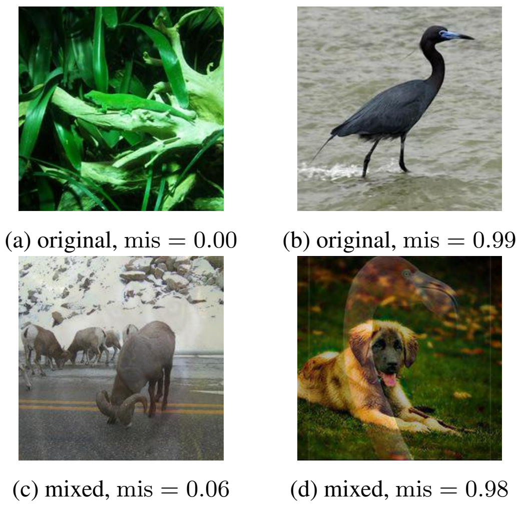


Fig. 7: Sample of images with various misunderstanding.

The result is shown in Table 1, which expresses a very narrow gap between humans and ResNet for high mixing rate, and it even over-performs humans when the mixing rate reaches 0.1. It is proof that ResNet can extract common features from the images — even some features somehow illegible for humans. Therefore, ResNet can be a faithful and robust adversary and the standard of our method.

Table 1 Accuracy comparison for random generated test sets.

Mix Rate	0.1	0.2	0.3	0.4	0.5
ResNet-152 Acc.	0.759	0.842	0.912	0.959	0.982
Human Acc.	0.733	0.890	0.952	0.986	0.991

We can then describe the 'easily detected' standard: the lower confidence the adversary has in the image, the better the image is. Namely, for one image p with label $l_p \in \{0, 1\}$, suppose the value output by ResNet model is $v(p)$ which is a real value in $[0, 1]$, the *misunderstanding* of the image is

$$\mathrm{mis}(p) \triangleq |l_p - v(p)|.$$

The higher misunderstanding, the more difficult it for attackers — especially for those having limited ability, since they can only extract features simpler than ResNet-152. A little confusion for ResNet-152 can be a fatal error for limited machines. Figure 7 shows some images that varied in misunderstandings, which may express the difference of focuses between human vision and computer vision.

Our sieving algorithm uses ideas that somehow appeared in GAN and genetic algorithm [21] to eliminate images with low misunderstanding. The algorithm starts with randomly generated initial sets. In each iteration, we train the model using the random partition of sets into a train set and test set, then delete those with very low misunderstanding in the test set. To fill up the slots for deleted images, generate randomly from the base data set except the last iteration, in which case we simply return the resulted sets.

The pseudo-code implementation for one iteration of the sieving algorithm is described in Algorithm 2. It has some minor differences from the description to be concise: it accepts an empty initial set and needs no extra operation at the last iteration.

The model trained in each iteration may be biased since it's based on small, partial, and immature candidate image distribution. However, we argue that by training a brand-new adversary model in each iteration generalizability is not important from the temporary model, instead, we can take advantage of the potential over-fitting. Over-fitting is to extract simple features and make use of them, and

Algorithm 2 Algorithm for iteration

```
Input: two sets of original and mixed images
Output: two sieved sets of original and mixed images
 1: procedure ITERATION($\mathrm{Original}, \mathrm{Mixed}$)
 2:     while $\mathrm{size}(\mathrm{Original}) \le \mathrm{SizeLimit}$ do
 3:         Insert an image from base data set into $\mathrm{Original}$
 4:     end while
 5:     while $\mathrm{size}(\mathrm{Mixed}) \le \mathrm{SizeLimit}$ do
 6:         Insert a newly mixed image into $\mathrm{Mixed}$
 7:     end while
 8:     $\mathrm{OS}_1, \mathrm{OS}_2 \leftarrow \mathrm{RandomPartition}(\mathrm{Original}, \mathrm{TrainRatio})$
 9:     $\mathrm{MS}_1, \mathrm{MS}_2 \leftarrow \mathrm{RandomPartition}(\mathrm{Mixed}, \mathrm{TrainRatio})$
10:     Train model with train=$\mathrm{MS}_1 \cup \mathrm{OS}_1$,test=$\mathrm{MS}_2 \cup \mathrm{OS}_2$
11:     for $x \in \mathrm{MS}_2 \cup \mathrm{OS}_2$ do
12:         if $|\mathtt{label}(x) - \mathtt{model}(x)| \le \mathrm{PrLimit}$ then
13:             Remove $x$ from set it belongs to
14:         end if
15:     end for
16:     return $(\mathrm{OS}_1 \cup \mathrm{OS}_2, \mathrm{MS}_1 \cup \mathrm{MS}_2)$
17: end procedure
```

over-fitted model can help us eliminate such candidates. Therefore, problems in normal training can actually benefit out result in the accuracy gap and efficiency.

In our experiment, we test the $\mathrm{rate}_{\text{mix}} \ge 0.3$, under which user experience will significantly drop since the accuracy is less than 0.9. Some other hyperparameters including $\mathrm{TrainRatio}$ and $\mathrm{PrLimit}$ are crucial to efficient convergence, which acts like learning rates in machine learning. $\mathrm{TrainRatio}$ should be moderate, for the too-small test set will significantly reduce the efficiency in sieving, and too small train set will lead to instability. In practice, we use $\mathrm{TrainRatio}$ within the range $[0.5, 0.75]$. $\mathrm{PrLimit}$ is hard to determine beforehand. However, we can run a trail round to observe the distribution and decide what $\mathrm{PrLimit}$ should we take. $\mathrm{PrLimit}$ can also be chosen adaptively. For example, take the median value of the current iteration.

By choosing proper hyperparameters, the algorithm can obtain a stable, good result within a few iterations in an experiment. The sieving method worked on the $500,000$ subset of ILSVRC as base data set and generated a test set with a total of $8,000$ images. Training on such a small set is meaningless, so the ResNet-152 in the table is trained using a randomly generated set of the $500,000$ subset. Table 2 shows the result. Figure 8 includes both random generated set and sieved set for comparison.

It's clear that for $\mathrm{rate}_{\text{mid}} = 0.3$ the method enlarged the performance gap most, we now focus on such case. Table 3 lists

Table 2 Accuracy comparison for sieved test sets.

Mix Rate	0.3	0.4	0.5
ResNet-152 Acc.	0.731	0.854	0.899
Human Acc.	0.973	0.993	0.992

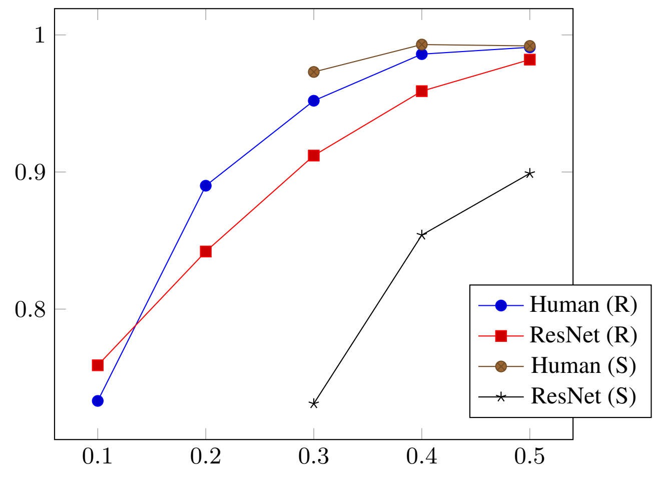


Fig. 8: Accuracy comparison.
R for random generated set, S for sieved set.

comparison of the case $\text{rate}_{\text{mid}} = 0.3$ separately, in which we calculate the advantage of human over machine as the expected number of error made by machine before human made a mistake, i.e.

$$\text{Advantage} = \frac{1 - \text{Acc}_{\text{machine}}}{1 - \text{Acc}_{\text{human}}}.$$

Table 3 Accuracy gap, for $\text{rate}_{\text{mid}} = 0.3$.

Set \ Acc.	Human	ResNet-152	Advantage over machine
Random	0.952	0.912	1.833
Sieved	0.973	0.731	9.962

Even though the model had met all of the images in the base test set before, it turns out to be very hard for the model to distinguish on the selected set — it can not even answer half of the tasks since its accuracy is lower than 0.75.

It also works surprisingly well in helping human tester to distinguish the CAPTCHA. One possible explanation for such phenomenon is that those unsure pictures ResNet selected for us are exactly the third type in Figure 2, that humans can distinguish them by semantic relations within the image much more efficiently than machines.

Therefore, the sieving method improved the quality of the set from both human and machine aspects.

3.3 Enlarge Base Data Set

Our method is based on semantic information conveyed in the image. To enlarge the human-machine gap, we have an optional choice to vary the base data set. There are many forms of images acceptable by humans, while neural networks can store limited knowledge. If the base data set is chosen from a broader range of image sources, it's more likely that machines can't efficiently learn all kinds of images' characteristics.

The sieving method provides us with a straightforward approach to achieving varying processes: add whatever kinds of images into the base data set. Some data sets indeed have features easy to learn, include bold lines and large color blocks. However, the sieving method guarantees that it's at least as good as the original base data set: bad images will be sieved away during the process. Therefore, it's unnecessary to check the image source's quality to make our method works well.

Since our method only uses unlabeled images, adding images from different sources requires little human interference and can be done by automatic tools. With proper configuration, the Mixaptcha system can automatically update the test set every day using images in the Internet's public domain, which also guarantees that it cannot be hacked by collecting the whole test set.

4 Application Performance

User experience is also an essential part of CAPTCHAs. We design the verification procedure of one verification process that is made up of 9 images sampled from a test set like reCAPTCHA. The verification finishes only if the user gives the correct response to 9 images in one round. Figure 9 illustrates the design of user interface.

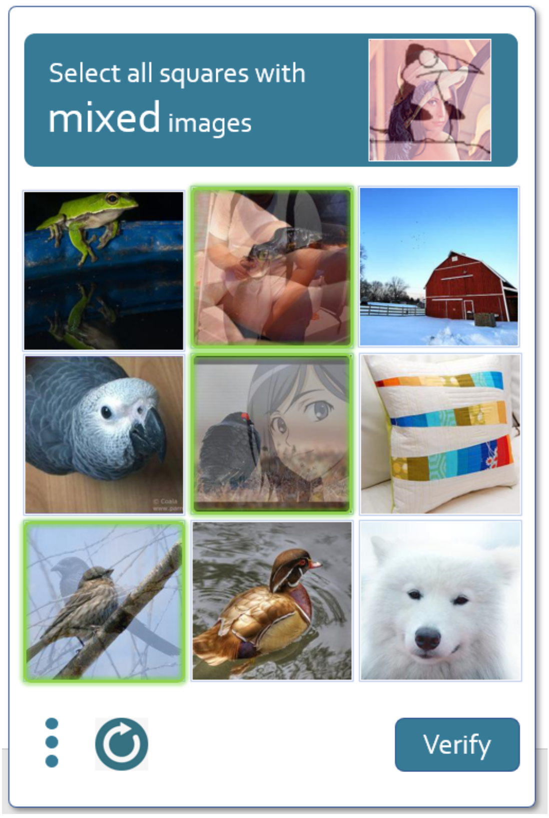


Fig. 9: User interface design.

Let the probability of the accuracy be p, the number of rounds is t. Assume that pictures are uniformly drawn from test set, then the probability of passing one round is

$$P_1 = p^9,$$

and thus the probability that succeed within t round is

$$P_t = 1 - (1 - p^9)^t.$$

Table 4 shows the success probabilities of human (with $p = 0.97$), machine (with $p = 0.75$) and random clicking (with $p = 0.5$).

From the table, we can conclude that Mixaptcha does efficiently distinguish bots from human users, and it's expected to finish the verification process for human users no more than 2 rounds.

Table 4 Success probabilities of human, machine and random clicking.

Rounds	1	2	3	4
Human	76%	94.25%	98.6%	99.7%
ResNet-152	7.50%	14.4%	20.8%	26.8%
Random	0.20%	0.39%	0.58%	0.78%

It is also more pleasant to finish the verification process than reCAPTCHA. To show it, we asked 27 volunteers to complete sample tasks on Mixaptcha and reCAPTCHA, and the preferences on the two kinds are shown in Figure 10.

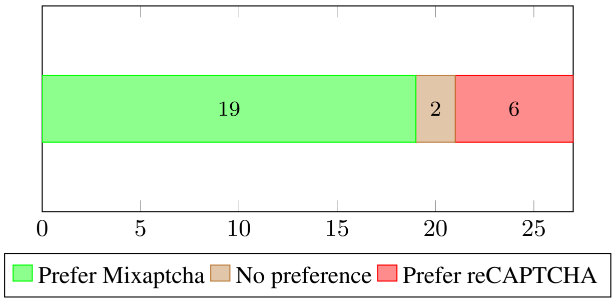


Fig. 10: Preference survey, 27 valid answers

More than half of the participants give a positive answer to our work, mostly because it is more enjoyable and requires only intuition towards options. For those who dislike Mixaptcha, one mentioned Mixaptcha was not a usual challenge, and people are more familiar with tasks used in reCAPTCHA. It's undeniable that reCAPTCHA has a dominant position among all CAPTCHAs, and familiarity is a significant aspect of user-friendliness. However, it doesn't affect the fact that our method itself is satisfying for users. Another negative answer was given for the opinion that people with visual weakness may have difficulty distinguishing these pictures, and we will discuss it in the last section.

5 Conclusion and future work

We've presented a new method of CAPTCHA challenge. The test set can be generated using unlabeled images with a little quality requirement. Humans can over-perform machines with a significant gap on the task, which requires users to distinguish the picture is mixed or original. We use an iterative adversarial generating method to enlarge the performance gap by improving humans' experience and reducing the accuracy of the machine.

There is still room for improvement in our method. Although experiments and tests show a positive result that we can effectively distinguish testers and machines, it may be a challenge for some specific users. Some people consider it an effortless task and reach high accuracy, while others have difficulty seeing the images. One reason is that the display device may have different configurations. The screen with too high or too low contrast ratio may negatively influence the result. Also, different people may show different sensitivities to the challenge based on visual conditions. Table 5 shows the different accuracies for different testers on one subset of test set.

Table 5 Accuracy using random generated set, mix rate = 0.3.

Human Tester	1	2	3	4	5
Accuracy	0.980	0.921	0.987	0.964	0.908

To make it more suitable for human users, one possible solution is to improve user experience by using extra scoring models to select good test set. The scoring model can use the ratings given by user and learn which images are more suitable. It would also be helpful in filtering images containing uncomfortable contents, and make the verification process more satisfying.

Acknowledgment

This work is under the supervision of Prof. Cewu Lu.

6 References

1 Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. In *International conference on the theory and applications of cryptographic techniques*, pages 294–311. Springer, 2003.
2 Luis Von Ahn, Manuel Blum, and John Langford. Apart. *Communications of the ACM*, 47(2):57, 2004.
3 Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.
4 Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up captcha? a captcha based on image orientation. In *Proceedings of the 18th international conference on World wide web*, pages 841–850, 2009.
5 Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, and John C. Mitchell. The end is nigh: Generic solving of text-based captchas. In *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, San Diego, CA, August 2014. USENIX Association.
6 M. Tariq Banday and N. A. Shah. A study of captchas for securing web services, 2011.
7 Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. Yet another text captcha solver: A generative adversarial network based approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, CCS '18, page 332348, New York, NY, USA, 2018. Association for Computing Machinery.
8 Chunhui Li, Xingshu Chen, Haizhou Wang, Yu Zhang, and Peiming Wang. An end-to-end attack on text-based captchas based on cycle-consistent generative adversarial network. *arXiv preprint arXiv:2008.11603*, 2020.
9 Wikipedia contributors. Captcha — Wikipedia, the free encyclopedia, 2020. [Online; accessed 1-January-2021].
10 Daniel Saez. Correcting image orientation using convolutional neural networks: A hands-on introduction to deep learning applications. `https://d4nst.github.io/2017/01/12/image-orientation`.
11 Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. Im not a human: Breaking the google recaptcha. *Black Hat*, pages 1–12, 2016.
12 Suphannee Sivakorn, Iasonas Polakis, and Angelos D Keromytis. I am robot:(deep) learning to break semantic image captchas. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 388–403. IEEE, 2016.
13 J. Lung. Ethical and legal considerations of recaptcha. In *2012 Tenth Annual International Conference on Privacy, Security and Trust*, pages 211–216, 2012.
14 Eric Schenk, Claude Guittard, et al. Crowdsourcing: What can be outsourced to the crowd, and why. In *Workshop on open source innovation, Strasbourg, France*, volume 72, page 3. Citeseer, 2009.
15 Robert Hummel. Image enhancement by histogram transformation. *ieht*, 1975.
16 Japanese Standards Association et al. Digital compression and coding of continuous-tone still images: Requirement and guidelines. *JIS X 4301*, pages 12–28, 1995.
17 Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.
18 Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.
19 Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
20 Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
21 Melanie Mitchell. *An introduction to genetic algorithms*. 1998.