3E-Solver: An Effortless, Easy-to-Update, and End-to-End Solver with Semi-Supervised Learning for Breaking Text-Based Captchas

Xianwen Deng[1*], **Ruijie Zhao**[1*†], **Yanhao Wang**[2], **Libo Chen**[1], **Yijun Wang**[1†], **Zhi Xue**[1]

[1] Shanghai Jiao Tong University, Shanghai, China
[2] NIO Security Research, Shanghai, China

{2594306528, ruijiezhao, bob777, ericwyj, zxue}@sjtu.edu.cn, wangyanhao136@gmail.com

Abstract

Text-based captchas are the most widely used security mechanism currently. Due to the limitations and specificity of the segmentation algorithm, the early segmentation-based attack method has been unable to deal with the current captchas with newly introduced security features (e.g., occluding lines and overlapping). Recently, some works have designed captcha solvers based on deep learning methods with powerful feature extraction capabilities, which have greater generality and higher accuracy. However, these works still suffer from two main intrinsic limitations: (1) many labor costs are required to label the training data, and (2) the solver cannot be updated with unlabeled data to recognize captchas more accurately. In this paper, we present a novel solver using improved FixMatch for semi-supervised captcha recognition to tackle these problems. Specifically, we first build an end-to-end baseline model to effectively break text-based captchas by leveraging encoder-decoder architecture and attention mechanism. Then we construct our solver with a few labeled samples and many unlabeled samples by improved FixMatch, which introduces teacher forcing, adaptive batch normalization, and consistency loss to achieve more effective training. Experiment results show that our solver outperforms state-of-the-arts by a large margin on current captcha schemes. We hope that our work can help security experts to revisit the design and usability of text-based captchas. The source code of this work is available at https://github.com/SJTU-dxw/3E-Solver-CAPTCHA.

1 Introduction

Since automated robots are widely used to carry out large-scale malicious activities on web applications, captcha (Completely Automated Public Turing test to tell Computers and Humans Apart) was proposed to distinguish computer programs from humans [Von Ahn *et al.*, 2003]. An overly complex captcha will make users feel disgusted and may require multiple attempts to pass the challenge, which means that the ideal captcha scheme should meet the condition that is difficult for computer programs to solve but easy for humans. Currently, captchas mainly include three types: text-based captchas, image-based captchas [Gossweiler *et al.*, 2009], and voice-based captchas [Gao *et al.*, 2010]. Among them, text-based captchas are widely used by most websites due to their user-friendliness. Thus, it is essential to comprehensively evaluate whether the existing text-based captcha schemes can withstand computer program attacks.

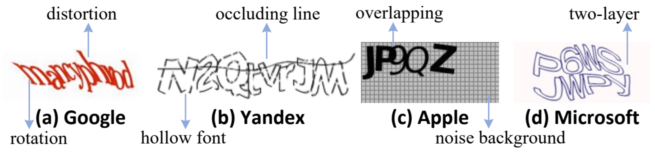


Figure 1: Security features of text-based captchas.

Over the past decade, many different methods have been proposed to break the text-based captchas, which can be divided into two types according to the solving process: segmentation-based methods [Yan and El Ahmad, 2008; Bursztein *et al.*, 2011; Gao *et al.*, 2013; Gao *et al.*, 2017] and end-to-end methods [Zhan *et al.*, 2018; Li *et al.*, 2021]. Segmentation-based methods require three steps to perform attacks: preprocessing, segmentation, and recognition, among which the segmentation step significantly affects the recognition performance. However, as shown in Figure 1, more sophisticated security features are applied to the current captchas. It is challenging for segmentation algorithms to deal with sophisticated security features, which means that these segmentation-based methods are no longer applicable. Recently, some end-to-end methods based on deep learning algorithms with better generality and recognition capabilities have succeeded in text-based captchas breaking. However, they usually require a lot of labor costs to label samples and can not leverage unlabeled data. Thus, we argue that the solver for breaking text-based captchas should adhere to the following requirements:

- **Effortless.** Labeling training data is a labor-intensive and time-consuming process, so we should train the solver with as few labeled captchas as possible.

[*] Xianwen Deng and Ruijie Zhao contributed equally to the work.
[†] Corresponding authors: Ruijie Zhao and Yijun Wang.

- **Easy-to-Update.** Real-world unlabeled captchas are easy to obtain and collect, and many of them cannot be accurately recognized by the original solver. Thus, it is necessary to update the solver with unlabeled captchas to improve recognition performance.
- **End-to-End.** To meet the requirements of real-time and generality, attackers should not adopt any preprocessing methods, which means that breaking the captcha requires an end-to-end solver.

To address the challenges mentioned above, we propose an effortless, easy-to-update, and end-to-end solver (3E-Solver) to automatically solve text-based captchas based on semi-supervised learning. Unlike previous deep learning-based methods, our approach requires significantly fewer labeled samples but can achieve higher accuracy. To be specific, we first design an attention-based encoder-decoder baseline model, denoted as AED-Net, for end-to-end recognition without any preprocessing operations on the captcha, and take this AED-Net as both the student network and the teacher network. After that, the improved FixMatch, a state-of-the-art semi-supervised framework for captcha recognition tasks based on FixMatch, is applied to train the solver through supervised loss and unsupervised loss effectively. We perform the teacher forcing strategy for labeled data to obtain supervised loss. Then, combined with adaptive batch normalization to make batch normalization (BN) layers adaptive to discrepant data, the student network and the teacher network predict strongly augmented and weakly augmented unlabeled captchas, respectively. Finally, it combines consistency regularization and pseudo-labeling to compute the unsupervised loss on unlabeled data, which is added to the supervised loss for training. We evaluate our solver on eight text-based captcha schemes of the top-50 popular websites ranked by alexa.com, including Google, Yandex, Microsoft, Apple, Sina, Weibo, Wikipedia, and Ganji. Results show that our solver can more accurately recognize captchas than the state-of-the-arts with a few labeled samples. The major contributions of the proposed work are three-fold:

- We develop an end-to-end baseline model (AED-Net) to effectively break text-based captchas by leveraging encoder-decoder architecture and attention mechanism. It should be noted that AED-Net can directly break captchas without relying on any preprocessing method.
- We design an improved FixMatch framework to fuse the supervised loss and unsupervised loss to train our solver, which introduces teacher forcing, adaptive batch normalization, and consistency loss to improve performance. To our best knowledge, this is the first investigation in this direction.
- Our 3E-Solver outperforms the state-of-the-art methods by a large margin on current captcha schemes using a few labeled samples.

2 Related Work

Segmentation-Based Methods. Early attack methods include three steps: preprocessing, segmentation and recognition, and most of them work well only on the easy-to-segment captcha scheme. Based on shape context matching by finding distances between characters [Mori and Malik, 2003], the earliest attempt successfully broke two simple captcha schemes. To break the captchas with distortion technology [Converse, 2005] (i.e., the pixels in each column of the image are translated up or down), [Yan and El Ahmad, 2007] performed pattern recognition by counting the pixels of each character, which effectively broke the captcha schemes provided at captchaservice.org. Color filling segmentation (CFS) algorithm [Gao *et al.*, 2015] was used to detect connected components and broke 19 captcha schemes. Besides, a convolutional neural network (CNN) with powerful feature extraction capabilities has also been introduced for accurate recognition [Tang *et al.*, 2018]. However, these segmentation-based methods require complex preprocessing and rely on segmentation algorithms for each captcha scheme. With the emergence of more sophisticated security features, characters are difficult to separate, making these methods invalid.

End-to-End Methods. The development of artificial intelligence technology makes it possible to break captchas end-to-end. Different deep learning networks have been proposed to solve text-based captchas, including CNN [Stark *et al.*, 2015] and recursive cortical networks [George *et al.*, 2017]. Then several sequence models were proposed to recognize variable-length captchas. For example, recurrent neural network (RNN) [Le *et al.*, 2017] and attention-based RNN [Zi *et al.*, 2019] show strong generality on various captcha schemes. However, these methods require a large number of labeled captcha samples to train the solver for better robustness.

Recently, attack methods with fewer annotated samples have become a research focus. [Ye *et al.*, 2018] proposed a generative adversarial network (GAN) to generate synthetic captchas, which were utilized for training a CNN-based solver. Furthermore, the generator can also process the captcha with security features into a clean captcha. Later, inspired by the remarkable progress made by representation learning in unsupervised learning tasks, a novel attack method [Tian and Xiong, 2020] based on representation learning with fewer labeled captchas was proposed, and the decomposer was used as a core component to remove noisy backgrounds. Although these methods significantly reduce the number of labeled captchas and have much better recognition performance than previous work, their solvers still have difficulty in breaking sophisticated captcha schemes, especially for the Google scheme. Besides, the above methods all excessively depend on removing the noisy backgrounds through the preprocessing model, and they all focus on background processing rather than character recognition, so they cannot deal with the captcha with some security features on characters (e.g., rotation and distortion). In this paper, we first propose an attention-based encoder-decoder model, which is capable of recognizing captchas with all kinds of security features. Then the improved FixMatch is used to train this model with a few labeled captcha samples and many unlabeled captcha samples. Results show that our method outperforms state-of-the-arts on various captcha schemes, as detailed in the later experiment section.

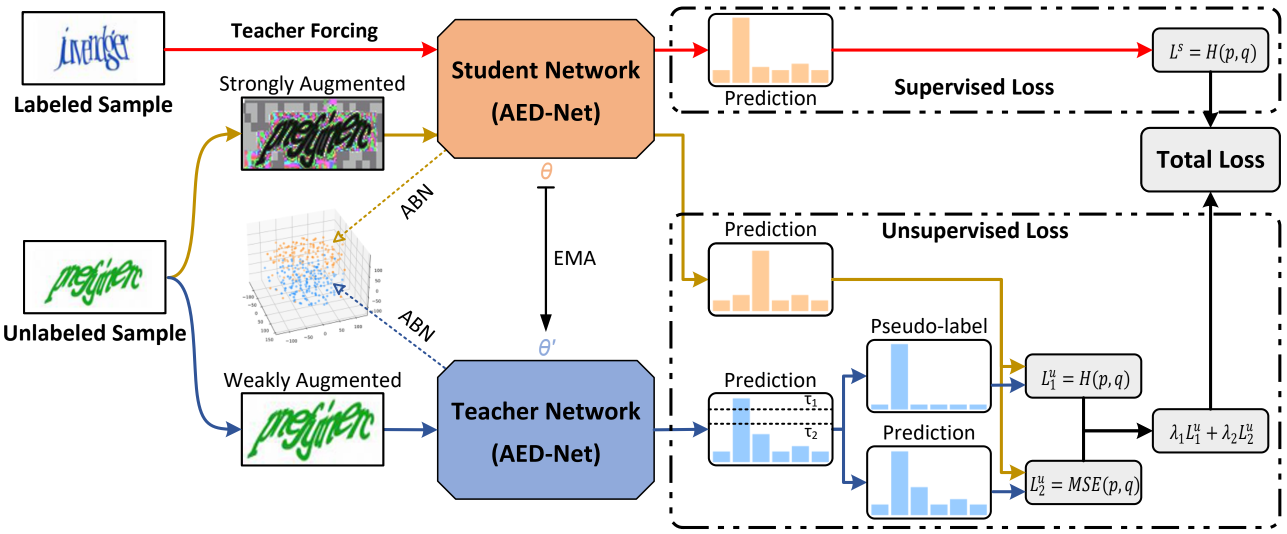


Figure 2: The schematic illustration of our improved FixMatch framework. We first develop an end-to-end baseline model (AED-Net; see Figure 3) to break text-based captchas by leveraging encoder-decoder architecture and attention mechanism. After that, we train our 3E-Solver with a few labeled samples and many unlabeled samples by improved FixMatch.

3 Methodology

The proposed attention-based encoder-decoder model and the improved FixMatch are described in this section. The overview of our approach is shown in Figure 2.

3.1 End-to-end Baseline Model (AED-Net)

Our baseline model (AED-Net) deploys an attention-based encoder-decoder architecture, as illustrated in Figure 3.

Encoder. The CNN-based encoder is used to extract feature vectors, representing the left-to-right information of the captcha image. The input is directly a captcha image. For convenience, we resize different schemes of captchas to the same size, that is 64 × 128. The encoder uses ResBlocks to extract features and the maximum pooling layer to compress the size of the feature map. ResBlock [He *et al.*, 2016] leverages residual structure to address the degradation problem of deep neural networks, but it does not change the size of the feature map. Therefore, we carefully design the kernel size of the maximum pooling layers to make the height of the final feature map become 1, corresponding to the height of the captcha image. The final feature map $\mathbf{F} \in \mathcal{R}^{1\times W\times P}$ can be split into W feature vectors, $f_i \in \mathcal{R}^{1\times P}, i = 1, \cdots, W$, where W is the width of the final feature map, and P is the number of channels. These W feature vectors represent the left-to-right information of the image.

Decoder. The decoder is composed of two GRU modules, one attention module, and one prediction module. First, since the corresponding receptive field of a feature vector f_i may be smaller than the area of a character, we use a GRU module to allow adjacent feature vectors to exchange information: $\mathbf{X} = GRU(\mathbf{F})$, where $\mathbf{X} \in \mathcal{R}^{1\times W\times P'}$.

In the decoding phase, at time step t, the another GRU takes the output of the decoding process at time step $t-1$ as input, and generates the output a_t: $a_t = GRU(y_{t-1}, h_{t-1})$,

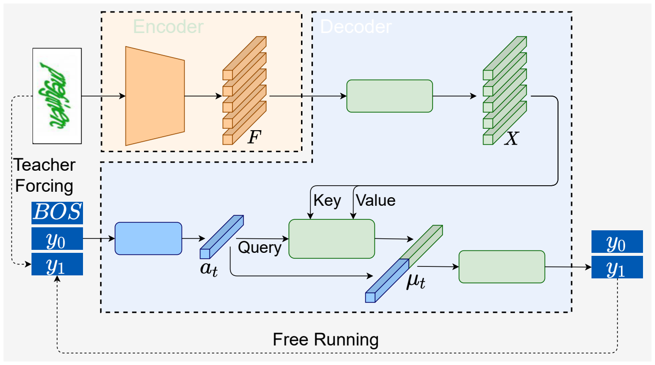


Figure 3: The schematic illustration of the proposed AED-Net in Figure 2, which is composed of an encoder and a decoder.

where h_{t-1} is the hidden state of the GRU at time step $t-1$, y_{t-1} is the output of the decoding process at time step $t-1$.

a_t is considered as $query$ of the attention module. Then the attention score is computed:

$$A_i^t = softmax(a_t^T x_i), i = 1, \cdots, W, \tag{1}$$

where a_t^T means the transpose of a_t, and x_i is the i-th feature vector in $\mathbf{X}$. Then the weighted average method is used:

$$\mu_t = Concat(\sum_{i=1}^{W} A_i^t x_i, a_t), \tag{2}$$

where $Concat$ means concentrating two vectors into one. Finally, we use the prediction module, a fully connected layer, to obtain the final decoding output at time step t:

$$y_t = FC(\mu_t). \tag{3}$$

In traditional free-running mode, the GRU module takes the decoding output at time step $t-1$ as input, as mentioned

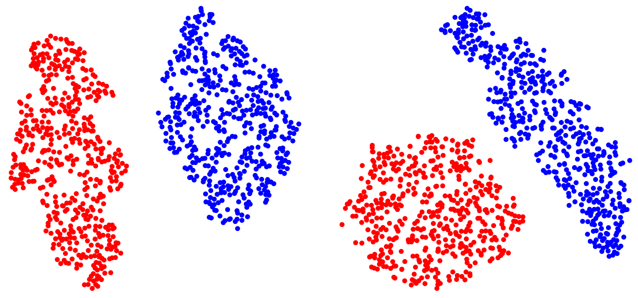

(a) Shallow layer distributions (b) Deep layer distributions

Figure 4: The t-SNE visualization of the BN statistics in both shallow and deep layers. Each dot represents the BN statistics in a mini-batch. Red dots come from weakly augmented Google captchas, while blue dots come from strongly augmented Google captchas. The size of each mini-batch is 64.

above. While in teacher-forcing mode, the GRU directly uses the ground truth character as input. The teacher-forcing mode can accelerate model convergence and improve model performance with limited labeled samples.

3.2 Improved FixMatch Framework

In the original FixMatch, the model predicts weakly augmented samples and takes the predicted results with high confidence as pseudo-labels. Then the model is trained using strong augmented samples and these pseudo labels. Such a simple method has achieved outstanding results in semi-supervised learning, but it still has two flaws for training a high-performance captcha solver. First, this method uses a single model to predict both weakly and strongly augmented data. However, the statistics in BN layers [Ioffe and Szegedy, 2015] from weakly and strongly augmented data can be very different. To prove this, we product a pilot experiment. By leveraging the Inception-v3 model [Szegedy *et al.*, 2016] pretrained on ImageNet Dataset [Russakovsky *et al.*, 2015], we compute the BN statistics from strongly and weakly augmented Google captchas in both shallow and deep layers. The mean and variance in the same BN layer are concentrated into a vector, and then we use t-SNE [Van der Maaten and Hinton, 2008] for dimensionality reduction. Results are shown in Fig. 4. It's clear that the statistics in BN layers from weakly and strongly augmented data have a large gap. Moreover, the pseudo-label is only retained if the model produces a high-confidence prediction. The threshold is usually set to as high as 0.95, which makes some hard samples with low confidence be wasted.

To enhance the performance of semi-supervised learning, we make three improvements to the original FixMatch.

- **Teacher Forcing.** We employ the teacher forcing strategy when computing supervised loss, which can maximize the use of labeled samples.
- **Adaptive Batch Normalization.** The teacher model is introduced and only predicts weakly augmented images in the training stage. Since weakly augmented samples are closer to the real samples, the estimated statistics in each BN layer are suitable for testing captcha images.
- **Consistency Loss.** For predictions of the teacher model with lower confidence, we directly use the softmax output to compare with the student model output.

Figure 2 shows the workflow of the proposed 3E-Solver that integrates labeled data and unlabeled data by using the improved FixMatch semi supervised learning. The loss function for our approach consists of a supervised loss term (denoted as $\mathcal{L}^s$) and two unsupervised losses (denoted as $\mathcal{L}_1^u$ and $\mathcal{L}_2^u$).

Supervised Loss on Labeled Captchas

Formally, let $X^l = \{(x_n^l, y_n) : n \in (1, \cdots, N)\}$ be a mini-batch of labeled samples, where N is the mini-batch size of labeled data and y are one-hot labels. Let $X^u = \{x_n^u : n \in (1, \cdots, N')\}$ be a mini-batch of unlabeled samples, where N' is the mini-batch size of unlabeled data. Denote the weights of student model and teacher model as θ and θ'.

$\mathcal{L}^s$ is computed through cross-entropy loss based on weakly augmented labeled data:

$$\mathcal{L}^s = \frac{1}{N}\sum_{n=1}^{N} H(y_n, f_{TF}(x_n^{l,W}, \theta, y_n)), \tag{4}$$

where $f_{TF}(x_n^{l,W}, \theta, y_n)$ means the student model makes predictions in teacher-forcing mode by using ground truth y_n for the weakly augmented labeled input $x_n^{l,W}$.

Unsupervised Loss on Unlabeled Captchas

Similar to FixMatch, our method calculates an artificial label for unlabeled sample. First, we use the teacher model to predict weakly augmented sample: $q_n = f(x_n^{u,W}, \theta')$. Then, we get the pseudo-label $\hat{q}_n = argmax(q_n)$. If the confidence of $\hat{q}_n$ is high, we leverage it as a label for the student model to predict strongly augmented samples:

$$\mathcal{L}_1^u = \frac{1}{N'}\sum_{n=1}^{N'} \mathbb{1}(max(q_n) \geq \tau_1) H(\hat{q}_n, f(x_n^{u,S}, \theta)), \tag{5}$$

where $x_n^{u,S}$ is the strongly augmented unlabeled sample, and τ_1 is the confidence threshold.

To leverage the information of hard samples, we choose a lower threshold τ_2. If the confidence of the output of the teacher model is greater than this threshold, we compare this output with the output of the student model through mean square error (MSE):

$$\mathcal{L}_2^u = \frac{1}{N'}\sum_{n=1}^{N'} \mathbb{1}(max(q_n) \geq \tau_2) ||q_n - f(x_n^{u,S}, \theta)||^2. \tag{6}$$

Our Network

The total loss of our network is

$$\mathcal{L}_{total} = \mathcal{L}^s + \lambda_1 \mathcal{L}_1^u + \lambda_2 \mathcal{L}_2^u, \tag{7}$$

where λ_1 and λ_2 are both scalar hyperparameters. The total loss is leveraged to update weights of the student network, then the weights of the teacher network in each training step

Algorithm 1 The Pipeline of Our Method

```
Require: Labeled dataset, unlabeled dataset, two confidence thresholds $\tau$ and $\tau'$, two scalar hyperparameters $\lambda_1$ and $\lambda_2$, the number of iterations $T$.
 1: Initial weights of the student($\theta$) and teacher($\theta'$) models
 2: for $i = 1$ to $T$ do
 3:     Sample a labeled batch $X^l = \{(x_n^l, y_n)\}$ of size $N$ and
        an unlabeled batch $X^u = \{x_n^u\}$ of size $N'$
 4:     $\mathcal{L}^s = \frac{1}{N}\sum_{n=1}^{N} H(y_n, f_{TF}(x_n^{l,W}, \theta, y_n))$
 5:     for $n = 1$ to $N'$ do
 6:         $q_n = f(x_n^{u,W}, \theta')$
 7:         $\hat{q}_n = argmax(q_n)$
 8:     end for
 9:     $\mathcal{L}_1^u = \frac{1}{N'}\sum_{n=1}^{N'} 1(max(q_n) \geq \tau) H(\hat{q}_n, f(x_n^{u,S}, \theta))$
10:     $\mathcal{L}_2^u = \frac{1}{N'}\sum_{n=1}^{N'} 1(max(q_n) \geq \tau') ||q_n - f(x_n^{u,S}, \theta)||^2$
11:     $\mathcal{L}_{total} = \mathcal{L}^s + \lambda_1 \mathcal{L}_1^u + \lambda_2 \mathcal{L}_2^u$
12:     Update the student model using gradient descent algorithm
13:     $\theta' = \alpha\theta' + (1 - \alpha)\theta$
14: end for
```

are updated via the exponential moving average (EMA) strategy in [Tarvainen and Valpola, 2017]:

$$\theta' = \alpha\theta' + (1 - \alpha)\theta, \tag{8}$$

where α is a fixed parameter, usually set to 0.999. The complete algorithm is illustrated in Algorithm 1.

4 Experiment

4.1 Experiment Settings

Data Preparation

We select eight current text-based captcha schemes of the top 50 most popular websites ranked by Alexa.com. For each captcha scheme, we collect 7200 captcha images, 2200 of them are manually labeled. Only 700 or even fewer labeled captchas are used to train the model, the other 1500 labeled images are used in the test phase. 5000 unlabeled samples are leveraged in our semi-supervised learning framework.

Implementation Details

In the training phase, we train the student model with an SGD optimizer, and the learning rate is set to 2×10^{-2}. The first confidence threshold τ_1 is set to 0.95, and the second threshold τ_2 is set to 0.3 for Google captchas and 0.6 for other schemes, respectively. The weight of the first unsupervised loss λ_1 is 2, while the second weight λ_2 gradually increases from 0 to 100. We run our approach 700 epochs for each captcha scheme, and then report experimental results. The proposed approach is implemented using PyTorch 1.9.0 and trained on the PC with Intel® Core™ i9-11900K@3.50 GHz, 64 GB RAM, and an NVIDIA GeForce RTX3090 GPU.

4.2 Evaluation on Current Captcha Schemes

The purpose of our work is to achieve an effortless, easy-to-update, and end-to-end solver for breaking text-based captchas, so we train our solver with improved FixMatch based on small-scale labeled samples. For each scheme, we use 500 labeled samples and 5000 unlabeled samples.

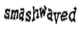
Scheme	Example	Accuracy	Attack Speed
Google		$76.4 \pm 0.3\%$	16.76 ms
Microsoft		$96.4 \pm 0.2\%$	15.89 ms
Yandex		$90.4 \pm 0.2\%$	16.23 ms
Wikipedia		$98.5 \pm 0.1\%$	16.43 ms
Weibo		$92.5 \pm 0.1\%$	15.16 ms
Sina		$97.1 \pm 0.1\%$	14.73 ms
Apple		$92.9 \pm 0.3\%$	14.51 ms
Ganji		$99.4 \pm 0.1\%$	14.11 ms

Table 1: Attack results on eight schemes of text-based captchas using 500 labeled samples and 5000 unlabeled samples. Attack speed represents the time to recognize a captcha image with the CPU mode.

Scheme	Unlabeled Samples				
	0	500	1000	3000	5000
Google	22.4%	26.3%	34.7%	61.7%	76.4%
Microsoft	45.1%	75.7%	83.4%	94.7%	96.4%
Yandex	62.7%	67.9%	76.0%	87.9%	90.4%

Table 2: Attack results with 500 labeled samples and different number of unlabeled samples on three schemes of text-based captchas.

Table 1 shows the success rate and the recognition time of 3E-Solver. Our solver accurately recognizes all captcha schemes, and its breaking success rate ranges from 76.4% to 99.4%. It can be seen that the Ganji captcha scheme has very few security features and a very short character length. Hence, our solver achieves the highest success rate of 99.4%, indicating that the Ganji scheme is the easiest for breaking among the eight captcha schemes. The success rates of the solver on Google scheme and Yandex scheme are 76.4% and 90.4%, respectively, which are lower than that of attacks on other schemes. Obviously, security features such as distortions and hollows can reduce the recognition accuracy of our solver to a certain extent. However, our solver can still break these captchas with a high success rate. Besides, the results show that the solver can recognize a captcha image of any scheme within 17ms. It can be concluded that 3E-Solver can accurately recognize various captchas and has a very fast recognition speed for real-time breaking.

We can obtain a large number of unlabeled captchas from various websites through web crawler technology, so it is easy to update the solver based on unlabeled captchas. To study the update capability of 3E-Solver, we use different numbers of unlabeled samples to train the solver. Specifically, we first selected three schemes of Google, Microsoft, and Yandex for experiments. Among them, the Google and Yandex schemes have the most complex security features, and the Microsoft scheme is the only two-layer structure scheme among the

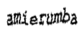
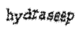
Dataset A	Example	[Tang *et al.*, 2018]	[Tian and Xiong, 2020]	3E-Solver (Ours)
Microsoft		50.9%	56.6%	**69.3 ± 3.2%**
Yandex		56.0%	63.2%	**77.9 ± 0.3%**
Wikipedia		90.0%	87.0%	**90.9 ± 0.1%**
Weibo		51.2%	88.2%	**92.5 ± 0.4%**
Sina		75.0%	84.0%	**91.9 ± 0.2%**
Apple		47.3%	68.4%	**84.7 ± 0.6%**
Dataset B	**Example**	**[Ye *et al.*, 2018]**	**[Tian and Xiong, 2020]**	**3E-Solver (Ours)**
Google		3.0%	9.0%	**69.5 ± 1.9%**
Microsoft		69.6%	75.5%	**85.7 ± 1.0%**
Wikipedia		78.0%	66.5%	**95.0 ± 0.4%**
Weibo		44.0%	82.5%	**99.0 ± 0.1%**
Sina		52.6%	63.5%	**69.8 ± 0.8%**

Table 3: Results of our 3E-Solver comparison with three state-of-the-art attacks on two datasets. We use the same number of labeled samples with them or even fewer.

Method	Google			Microsoft			Yandex		
	300	500	700	300	500	700	300	500	700
Mean Teacher	2.2%	13.8%	18.7%	0.7%	4.6%	18.8%	49.1%	69.5%	76.9%
FixMatch	20.5%	40.4%	47.5%	5.1%	73.2%	86.0%	86.4%	87.0%	89.3%
Ours w/o TF	19.0%	43.7%	51.7%	3.0%	48.7%	89.0%	85.5%	89.3%	**91.2%**
Ours w/o ABN	**58.1%**	68.4%	72.5%	**93.8%**	94.6%	95.5%	86.7%	88.4%	89.5%
Ours w/o CL	55.5%	74.9%	78.1%	93.0%	96.3%	97.0%	86.7%	89.7%	90.9%
Ours	57.7%	**76.4%**	**78.7%**	93.2%	**96.4%**	**97.0%**	**87.7%**	**90.4%**	90.7%

Table 4: Comparison with Mean Teacher and FixMatch using different number of labeled samples and 5000 unlabeled samples, and ablation experiments for three key improvements, namely teacher forcing (TF), adaptive batch normalization (ABN), and consistency loss (CL).

eight schemes. After that, in the experiment of each scheme, the number of labeled captchas is fixed to 500, and then these labeled samples are combined with 0, 500, 1000, 3000, and 5000 unlabeled samples for training. Results are shown in Table 2. Because these three schemes of captchas are designed with complex security features, it's hard to obtain high success rates with limited labeled samples. For example, the success rate of Google captchas is only 22.4% with 500 labeled samples. However, our 3E-solver can learn from unlabeled data, and the success rate is effectively increased to 76.4% with another 5000 unlabeled Google captchas.

4.3 Comparison with Prior Attacks

Table 3 compares our 3E-Solver to previous attacks. To ensure fair comparison, we perform on the same set of attack tasks with the training and test datasets used in [Tang *et al.*, 2018; Ye *et al.*, 2018; Tian and Xiong, 2020]. [Tang *et al.*, 2018] provided *dataset* A and used 2000 labeled samples. [Ye *et al.*, 2018] provided dataset B and used 500 labeled samples. [Tian and Xiong, 2020] used the same number of labeled samples with them and used additional 1000 unlabeled samples. We use only 500 labeled samples and additional 5000 unlabeled samples for training. Apparently, our method outperforms all these previous methods on all captcha schemes, especially for Google captchas. The key reasons for the obvious performance improvement are that our attention-based model can learn more useful features by modeling the global dependencies among nodes within input data, as well as the the improved FixMatch framework can further improve recognition performance by leveraging unlabeled captchas. The representation learning-based solver [Tian and Xiong, 2020] is the second best-performing method, which uses Double-DIP, an unsupervised image decomposition approach to remove noisy backgrounds. However, this method cannot solve captchas with a strong similarity between the character layer and background layer, and the recognition speed is slow (about two seconds for a captcha on an NVIDIA Tesla P100 GPU). The method of [Tang *et al.*, 2018] suffers from requiring a large number of labeled captchas for training and the GAN network adopted by [Ye *et al.*, 2018] is peculiarly prone to mode collapse, making both difficult to apply in practice. Besides, the above methods all excessively depend on re-

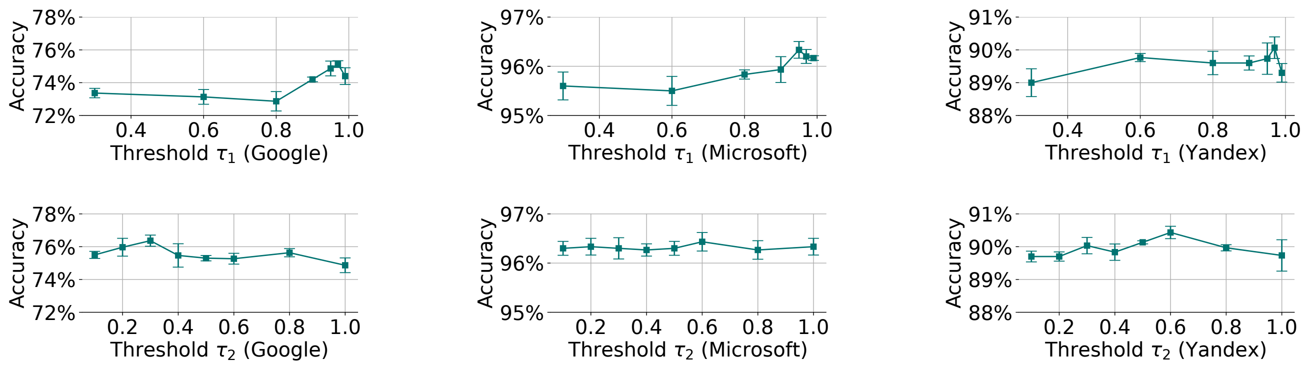


Figure 5: The impact of threshold τ_1 and τ_2. Experiments are conducted on *Google*, *Microsoft*, and *Yandex* captchas with 500 labeled and 5000 unlabeled samples.

moving the noisy backgrounds through the processing model, while ignoring the security features on characters, such as rotation and distortion. Compared with them, our approach can recognize captchas with all kinds of security features.

4.4 Ablation Study

To further examine the benefits that three improvements of our semi-supervised learning framework based on FixMatch bring to the performance, we conduct an ablation study on Google, Microsoft, and Yandex captchas, all of which employ complex security features. We also present the results of Mean Teacher [Tarvainen and Valpola, 2017] and FixMatch as baselines. The evaluation results are reported in Table 4. First, the removal of teacher forcing causes a significant performance drop on Google and Microsoft captchas, which illustrates the necessity of teacher forcing for recognizing captchas with complex security features (Google captchas employ excessive character distortion and overlap, while Microsoft captchas use the two-layer structure). Moreover, our approach w/o ABN is much less powerful on three schemes of captchas. Thus it can be concluded that adaptive batch normalization has a considerable contribution to our approach, since it uses two different models to estimate the statistics of weakly and strongly augmented data, respectively. Finally, the performance of our approach w/o CL shows that consistency loss can leverage information of low-confidence samples and brings performance enhancement.

4.5 Investigation on the Impact of Thresholds

In this section, we investigate the impact of the two most important hyperparameters τ_1 and τ_2 in our framework. To clearly show the impact of the two hyperparameters on performance, we focus on Google, Microsoft, and Yandex with 500 labeled and 5000 unlabeled samples.

We first investigate the impact of threshold τ_1 without considering consistency loss (CL), and the results are shown in the first row of Figure 5. It can be seen that using a small threshold τ_1 (i.e., 0.3) compared with large thresholds (i.e., 0.95 and 0.97), the success rates drop a large margin. In fact, threshold τ_1 controls the balance between the quality and quantity of pseudo-labels, that is, a higher threshold means that less quantity but higher quality pseudo-labels are leveraged.

Fixing the first threshold τ_1 to 0.95, the impact of the second threshold τ_2 is shown in the second row of Figure 5. To leverage information in hard samples, results with confidence levels higher than threshold τ_2 are used to calculate consistency loss $\mathcal{L}_2^u$. For the most difficult Google captchas, When threshold τ_2 is set to 0.3, the accuracy rate is the highest. For others, a higher threshold (i.e., 0.6) is more suitable. Different thresholds τ_2 will affect the accuracy, but the fluctuation of the accuracy rate does not exceed 3% for Google captchas and 1% for others.

5 Conclusion

In this paper, we propose an effortless, easy-to-update, and end-to-end solver for breaking text-based captchas. Specifically, we first develop an end-to-end baseline model to effectively recognize text-based captchas, which adopts encoder-decoder architecture and attention mechanism. Then we employ the improved FixMatch framework to leverage unlabeled samples to improve the recognition performance further. In addition, since our solver does not rely on any preprocessing model or algorithm, it can break captchas end-to-end and in real-time. Our approach is evaluated on eight text-based captcha schemes of the top-50 popular websites ranked by alexa.com. Experimental results show that our 3E-Solver outperforms three prior state-of-the-arts with a few labeled samples, especially for Google captchas. We hope that our work can help security experts to revisit the design and usability of text-based captchas.

Acknowledgments

We thank Prof. Haichang Gao and Prof. Zhanyong Tang for providing their datasets for fair comparison. This work was supported by the Foundation Item: Cyber Security from the National Key Research and Development Program of Shanghai Jiao Tong University under Grant 2019QY0703.

References

[Bursztein *et al.*, 2011] Elie Bursztein, Matthieu Martin, and John Mitchell. Text-based captcha strengths and weaknesses. In *CCS*, pages 125–138, 2011.

[Converse, 2005] Tim Converse. Captcha generation as a web service. In *Human Interactive Proofs*, pages 82–96, 2005.

[Gao *et al.*, 2010] Haichang Gao, Honggang Liu, Dan Yao, Xiyang Liu, and Uwe Aickelin. An audio captcha to distinguish humans from computers. In *International Symposium on Electronic Commerce and Security*, pages 265–269, 2010.

[Gao *et al.*, 2013] Haichang Gao, Wei Wang, Jiao Qi, Xuqin Wang, Xiyang Liu, and Jeff Yan. The robustness of hollow captchas. In *CCS*, pages 1075–1086, 2013.

[Gao *et al.*, 2015] Haichang Gao, Xuqin Wang, Fang Cao, Zhengya Zhang, Lei Lei, Jiao Qi, and Xiyang Liu. Robustness of text-based completely automated public turing test to tell computers and humans apart. *IET Information Security*, 10(1):45–52, 2015.

[Gao *et al.*, 2017] Haichang Gao, Mengyun Tang, Yi Liu, Ping Zhang, and Xiyang Liu. Research on the security of microsoft's two-layer captcha. *IEEE Transactions on Information Forensics and Security*, 12(7):1671–1685, 2017.

[George *et al.*, 2017] Dileep George, Wolfgang Lehrach, Ken Kansky, Miguel Lázaro-Gredilla, Christopher Laan, Bhaskara Marthi, Xinghua Lou, Zhaoshi Meng, Yi Liu, Huayan Wang, et al. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358(6368), 2017.

[Gossweiler *et al.*, 2009] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up captcha? a captcha based on image orientation. In *WWW*, pages 841–850, 2009.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[Le *et al.*, 2017] Tuan Anh Le, Atilim Giineş Baydin, Robert Zinkov, and Frank Wood. Using synthetic data to train neural networks is model-based reasoning. In *IJCNN*, pages 3514–3521, 2017.

[Li *et al.*, 2021] Chunhui Li, Xingshu Chen, Haizhou Wang, Peiming Wang, Yu Zhang, and Wenxian Wang. End-to-end attack on text-based captchas based on cycle-consistent generative adversarial network. *Neurocomputing*, 433:223–236, 2021.

[Mori and Malik, 2003] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *CVPR*, 2003.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[Stark *et al.*, 2015] Fabian Stark, Caner Hazırbas, Rudolph Triebel, and Daniel Cremers. Captcha recognition with active deep learning. In *Workshop New Challenges in Neural Computation*, volume 2015, page 94, 2015.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[Tang *et al.*, 2018] Mengyun Tang, Haichang Gao, Yang Zhang, Yi Liu, Ping Zhang, and Ping Wang. Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. *IEEE Transactions on Information Forensics and Security*, 13(10):2522–2537, 2018.

[Tarvainen and Valpola, 2017] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NIPS*, pages 1195–1204, 2017.

[Tian and Xiong, 2020] Sheng Tian and Tao Xiong. A generic solver combining unsupervised learning and representation learning for breaking text-based captchas. In *WWW*, pages 860–871, 2020.

[Van der Maaten and Hinton, 2008] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008.

[Von Ahn *et al.*, 2003] 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, 2003.

[Yan and El Ahmad, 2007] Jeff Yan and Ahmad Salah El Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *Annual Computer Security Applications Conference (ACSAC)*, pages 279–291, 2007.

[Yan and El Ahmad, 2008] Jeff Yan and Ahmad Salah El Ahmad. A low-cost attack on a microsoft captcha. In *CCS*, pages 543–554, 2008.

[Ye *et al.*, 2018] 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 *CCS*, pages 332–348, 2018.

[Zhan *et al.*, 2018] Hongjian Zhan, Shujing Lyu, and Yue Lu. Handwritten digit string recognition using convolutional neural network. In *International Conference on Pattern Recognition (ICPR)*, pages 3729–3734, 2018.

[Zi *et al.*, 2019] Yang Zi, Haichang Gao, Zhouhang Cheng, and Yi Liu. An end-to-end attack on text captchas. *IEEE Transactions on Information Forensics and Security*, 15:753–766, 2019.