Computers & Security 146 (2024) 104058

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Transformer-based end-to-end attack on text CAPTCHAs with triplet deep attention

Bo Zhang, Yu-Jie Xiong *, Chunming Xia, Yongbin Gao

School of Electronic and Electrical Engineering, Shanghai University of Engineering Science, Shanghai, 201620, China

ARTICLE INFO

Keywords:
Text-based captchas
Triplet deep attention
Query enhancement
Transfer learning strategy
Chinese captchas

ABSTRACT

Websites frequently use text-based captcha images to distinguish whether the user is a person or not. Previous research mainly focuses on different training strategies and neglects the characteristics of the text-based captcha images themselves, resulting in low accuracy. For text-based captcha images characterized by rotation, distortion, and non-character elements, we propose an end-to-end attack using a Transformer-based method with triplet deep attention. Firstly, the features of text-based captchas are extracted using ResNet45 with triplet deep attention module and Transformer encoder. The TDA module is capable of learning rotational and distortion features of characters. Subsequently, based on self-attention mechanism, design query, key, and value, and adopt the query enhancement module to enhance the query. The query enhancement module can strengthen character localization and reduce attention drift towards non-character elements. Finally, the feature maps are transformed into probabilities of character for the final text recognition. Experiments are conducted on captcha datasets based on Roman characters from 9 popular websites, achieving average word accuracy of 91.14%. To evaluate the performance of our method on data with small samples, experiments are conducted different scales of training data. Additionally, we use the method on Chinese text-based captcha tasks and achieve average word accuracy of 99.60%. The effectiveness of the method is also explored under conditions of lack of illumination and scene text recognition, where background interference is present.

1. Introduction

Completely Automated Public Turing Test to Tell Computers and Humans Apart (CAPTCHA) (Von Ahn et al., 2003) is a technique used to recognize the difference between a human and an automated software program. It determines whether a user is a real human by showing the user an image and asking them to enter the correct information. Different types of captchas often appear on websites during registration, login, voting and other steps that require confirmation (Xu et al., 2020). Among them, text-based captcha images are a widely used type due to their low cost (Nian et al., 2022). Text-based captchas rely on the user's ability to recognize text and accurately enter the corresponding characters to complete human and machine verification of the website. To resist automated programs, researchers typically increase the difficulty of recognition by adding elements such as noise, background clutter and distorted characters to text images to achieve maximum discrimination between humans and programs.

According to the framework for text-based captcha attack methods (Wang et al., 2023a), they can be categorized into traditional multi-stage attacks, one-stage attacks. Traditional multi-stage attacks involve a three-step process: preprocessing, segmentation and recognition. In these attacks, the target characters are determined by performing specific preprocessing operations tailored to the characteristics of the images. However, disadvantage of traditional multi-stage attacks is that the feature extraction method must be developed manually, which can be limited by human understanding and experience.

With the development in deep learning, one-stage attacks have become more important. These attacks use a well-trained model based on a deep learning network to directly recognize all characters in an image without the need for additional operations. One-stage attacks have become the standard method as they offer higher recognition accuracy of the model. Due to the limited amount of data, researchers have turned to the strategy of transfer learning to recognize text-based captcha images in the absence of specific amount of data. The primary concept behind transfer learning is to reduce the number of samples required. This is achieved by first training a base model with a synthetic dataset and then fine-tuning it with a small amount of real data. This approach addresses the challenges posed by the limited amount of data and helps to improve the model's performance in recognizing text-based captcha images.

* Corresponding author.
E-mail address: xiong@sues.edu.cn (Y.-J. Xiong).

https://doi.org/10.1016/j.cose.2024.104058
Received 8 February 2024; Received in revised form 12 July 2024; Accepted 13 August 2024
Available online 17 August 2024
0167-4048/© 2024 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

Table 1
The differences among existing methods.

Method	Characteristic	Preprocess	Type	Encoder	Decoder
Chen et al. (2018)	Matrix-based algorithm component merging	IP	Multi-stage	–	CNN
Wang et al. (2021a)	Large character sets 3D image-based scheme	–	Multi-stage	Faster R-CNN	CNN+ATTN
Zhang et al. (2022)	GAN Dark web	GAN,IP	Multi-stage	–	CNN
Zi et al. (2019)	Chinese captcha attention	–	End-to-end	Inception-v3	RNN
Ma et al. (2020)	Arithmetic operation puzzle-based capthcha	–	End-to-end	CNN	LSTM
Dou (2021)	Denoising encoder CRNN	–	End-to-end	Denoising encoder	CRNN
Nian et al. (2022)	Object detection Mask R-CNN	–	End-to-end	ResNet50 FPN	Mask R-CNN
Tian and Xiong (2020)	Priori denoising Semi-supervised	IP	Multi-stage	ResNet101	GRU
Li et al. (2021)	Cycle-GAN transfer learning	GAN,IP	Multi-stage	CNN	LSTM
Wang et al. (2023b)	Cross-domain few-shot	IP	Multi-stage	Meta-training	MAML ProtoNet
Ours	Triplet deep attention transformer	–	End-to-end	ResNet45 with TDA transformer	Transformer

The following abbreviations are used: IP = Image Processing ATTN = Attention.

Previous researchers have used long short-term memory (LSTM) (Yu et al., 2019) as decoder to transform feature maps into probabilities of character. LSTM is a type of recurrent neural network (RNN) (Wang et al., 2022) designed to solve the vanishing gradient problem that can occur in RNN and is widely used in various sequencing tasks (Cahuantzi et al., 2023). However, LSTM processes the data sequentially, resulting in slower model training and inference speed compared to Transformer's self-attention mechanism. When dealing with long sequence data, Transformer's self-attention mechanism is better at capturing long-range dependencies in sequences than LSTM. Text-based captcha recognition is a subset of text recognition. In text recognition, it is important to take into account the dependencies between the characters. Since the Transformer is more effective in capturing such relationships, it is used for text-based captcha recognition.

During our observation of the dataset, we found that text-based captcha images often feature characters that are rotated, distorted, and obscured, with rotated and distorted fonts being particularly common. We considered whether it would be possible to propose a method that transforms the feature dimensions of the image, allowing the model to learn the multidimensional features of the fonts to address this issue. Additionally, the images frequently contain non-character elements that cause attention drift. We considered enhancing the positional information of characters in the images to accurately distinguish between character and non-character elements.

Therefore, we propose a Transformer-based end-to-end method with triplet deep attention (TDA) to attack text-based captchas. Initially, features are extracted from the CAPTCHA image using ResNet45 with the TDA module and Transformer encoder. After obtaining the feature maps, query, key, and value for self-attention are designed, followed by function operations. Ultimately, the target characters are obtained. In the framework of one-stage attacks, our experimental results show excellent performance on 9 Roman real-world datasets. Exploring the accuracy of the model with different samples using transfer learning strategy. In order to facilitate the observation of experimental outcomes, visualizations of feature maps are provided. Furthermore, the method employed on 5 Chinese captcha datasets yielded outstanding results.

Our main contributions are as follows:

- We propose the TDA module to extract shallow and deep features from text-based captcha images by constructing relations between dimensions through rotation operations and residual connections to learn the dimensional features of rotated and distorted characters.
- We adopt the QE module to enhance query, which aims to improve character localization and reduce attention drift.
- Compared to existing studies, our method shows significant performance improvements on 9 captcha schemes with Roman characters and 5 captcha schemes with Chinese characters. In addition, we explored the method's performance on various real-world samples through transfer learning strategy.

2. Related work

2.1. Captcha attacks

Text-based captcha attacks were categorized into two types: traditional multi-stage attacks and one-stage attacks (Wang et al., 2023a). The characteristics of attack methods are shown in Table 1.

The traditional multi-stage attacks can be divided into three steps. The first step is preprocessing, which removed some of the noise from the image or converted the color image into monochrome images. This step needed to be performed depended on the specific task. The second step is segmentation, in which the text lines of the image were broken down into individual character images. The third step is recognition using machine learning algorithms such as SVM (Chandra and Bedi, 2021) or deep learning algorithms such as AlexNet (Krizhevsky et al., 2012), VGG (Simonyan and Zisserman, 2014) and ResNet (He et al., 2016). Multi-stage can be used for manual preprocessing tailored to specific datasets, but doing so may result in lower model generalization. Most segmentation-based captcha solvers follow this framework. Next, we will present some multi-stage methods.

Chen et al. (2018) first repaired the character contour using a thinning operation, obtained solid characters using an inner and outer contour filling algorithm, and then extracted individual characters using a least neighbor merging algorithm, which was the same as in the Ref. Chen et al. (2019), and finally recognized using convolutional neural networks (CNN). Wang et al. (2021a) analyzed the security of captcha with large character set. They first employed Faster R-CNN (Girshick, 2015) for text localization in click-based schemes, followed by CNN and RNN for text recognition. Additionally, they proposed a 3D image-based captcha scheme integrating semantic understanding and drag-and-drop actions, demonstrating through experiments that this scheme is more robust. Zhang et al. (2022) proposed DW-GAN for dark web text captcha, leveraging Generative Adversarial Networks (GAN) (Aggarwal et al., 2021) to counteract background noise. They employed character segmentation techniques to handle variable character lengths in captcha images. DW-GAN can automatically solve captcha challenges in fewer than three attempts.

In recent years, end-to-end methods have become increasingly popular with the further development of deep learning technology. The framework for one-stage attacks was dominated by end-to-end approaches. Researchers increasingly favored this approach, which eliminated the necessity for preprocessing and segmentation. Next, we will present some one-stage methods.

To avoid manual preprocessing steps and make the attack simple and effective. Zi et al. (2019) proposed an end-to-end network for attacking text-based captchas using an encoder–decoder architecture. The network used the Inception-v3 (Szegedy et al., 2016) model to extract the feature maps and then decoded them by LSTM (Yu et al., 2019). Ma et al. (2020) proposed a solution named Neural cpatcha Networks (NCNs), aimed at addressing arithmetic operations and puzzle-based cpatcha problems involving characters and spatial sequences of image features. NCNs primarily consist of three conv blocks and two

BiLSTM, utilizing Connectionist Temporal Classification (CTC) for loss computation.

With the development of object detection technology. Nian et al. (2022) used characters as target in object detection using Mask R-CNN (He et al., 2017) for image character detection. To begin with, the area of interest (ROI) was acquired by extracting the bounding box of the character through the RPN (Fan et al., 2020) network, which was then followed by classification. While Mask R-CNN was proficient in accurately classifying characters, it was not as effective when it came to sticky and overlapping characters. Dou (2021) proposed a denoising encoder aimed at eliminating noise in captchas, with the goal of restoring captcha images as closely as possible to their original form. This process operates automatically and can be considered an end-to-end approach. Additionally, Dou incorporates CRNN (Shi et al., 2016) as a component of his method for recognizing text-based captcha images. Initially, CRNN was primarily used for scene text recognition. Since scene text recognition and text-based captcha recognition shared similar objectives of identifying irregular characters in images. Following this idea, the visual model of ABINet (Fang et al., 2021) for the captcha attack task was reproduced and utilized as the backbone and baseline.

Manually annotating text-based captcha images from real websites incurred higher costs. Therefore, researchers actively explored methods to enhance generalization by training with a limited number of samples. Next, we will present some semi-supervised methods.

To reduce annotation costs, Tian and Xiong (2020) aim to recognize captchas through semi-supervised methods. Initially, they use a decomposer to decompose captcha into their basic components, thereby obtaining clear captcha images. Subsequently, they fed both labeled and unlabeled images into a semi-supervised classifier. Notably, for unlabeled images, they utilize Contrastive Predictive Coding (CPC) (Oord et al., 2018) for representation learning.

Due to the strong performance of Generative Adversarial Networks in generating new data. Li et al. (2021) proposed an approach based on cycle-consistent generative adversarial networks, reducing the cost of data labeling and successfully attacking the captcha schemes deployed by 10 popular websites. First, they use Cycle-GAN to train a captcha synthesizer to generate some fake samples. Then, they train a basic recognizer based on convolutional recurrent neural networks using these fake data. Subsequently, they employ an active transfer learning method to optimize the basic recognizer using a small amount of real labeled captcha samples. Wang et al. (2023b) employed a prototype network and a model-agnostic meta-learning strategy to tackle performance degradation issues resulting from cross-domain variations and class imbalances. They employed MAML (Finn et al., 2017) and ProtoNet (Snell et al., 2017) as classifiers. Subsequent evaluations in 5-shot and 10-shot tasks revealed an average character accuracy surpassing 90%.

2.2. Attention mechanism

Attention is a cognitive process that involves selectively focusing on certain parts of information and ignoring others. The attention mechanism helps perceive the context while refining the perceived information. The attention mechanism implemented with CNN is more focused on channel and spatial information, while the self-attention mechanism in the Transformer is dedicated to capturing similarity between time steps.

In CNN, attention-based methods typically focused on channel and spatial attention in feature map. By establishing mutual dependencies between channels and spatial dimensions, with the goal of enhancing feature representation capability, a series of notable methods emerged. Among them, the most noteworthy included squeeze-and-excitation networks (SENet) (Hu et al., 2018), ECANet (Wang et al., 2020b), and triplet attention (TA) (Misra et al., 2021).

SENet consisted of a squeeze operation and an excitation operation, which could adaptively recalibrate the importance of different channels in feature map, leading to improved performance in tasks such as image classification and object detection. ECA proposes an adaptive method for selecting the size of one-dimensional convolutional kernels. By avoiding dimension reduction, ECA effectively captures interactions across channels. The goal is to utilize global average pooling operations for capturing global contextual information in each channel, thereby contributing to the enhancement of the model's performance and generalization capabilities. The TA consists of three branches, establishing relationships between dimensions through rotation operations and residual connections, emphasizing the importance of multi-dimensional interactions. Moreover, TA has fewer parameters compared to other attention modules (Hu et al., 2018; Woo et al., 2018; Cao et al., 2019; Zhang et al., 2023) and performs well in object detection. These methods ingeniously applied operations such as convolution and pooling enabling the model to more precisely focus on information closely relevant to the current task.

The above was the attention method that people were using based on CNN. With the development of Transformer (Vaswani et al., 2017), more people turned their attention to self-attention.

The self-attention mechanism is a crucial concept in the Transformer, commonly employed in computer vision tasks (Dosovitskiy et al., 2020; Carion et al., 2020; Zhu et al., 2020; Wang et al., 2021b; Liu et al., 2021). In the self-attention mechanism, each element in the input sequence can attend to all other elements, and the attention weights are computed based on the content of the elements. It has two advantages : parallelism and long distance dependency. For each element in the sequence, attention weights are obtained by a weighted sum of all elements, and this process can be computed in parallel. This allows the model to more efficiently handle long sequences, unlike RNN that need to process sequentially. In RNN, information propagation occurs step by step, making it challenging to capture relationships at distant positions. The self-attention mechanism enables the model to consider information from the entire sequence at each position, thus better capturing dependencies at long distances within the sequence. Previous works (Zi et al., 2019; Li et al., 2021; Yusuf et al., 2023) used LSTM, a type of RNN, as decoder to decode extracted features. In comparison to LSTM, self-attention excels in decoding.

3. Methodology

The overall structure of the proposed method is illustrated in Fig. 1. Our method follow one-stage attack framework, employing an end-to-end method to recognize complete character sequences. Firstly, feature extraction is performed on the image, passing through ResNet45 with TDA module, where the TDA module is positioned after the 3×3 convolution in each basic block. Secondly, The feature maps generated by the Transformer encoder undergo further processing to extract key and value information. Concurrently, the query is enhanced through the QE module. In the end, the relationships between the query, key, and value are calculated to obtain the final probability prediction for characters.

3.1. Feature extraction

Feature extraction is responsible for extracting features from images, transforming the image into feature maps. Feature extraction consists of two modules: ResNet45 with TDA module and Transformer encoder.

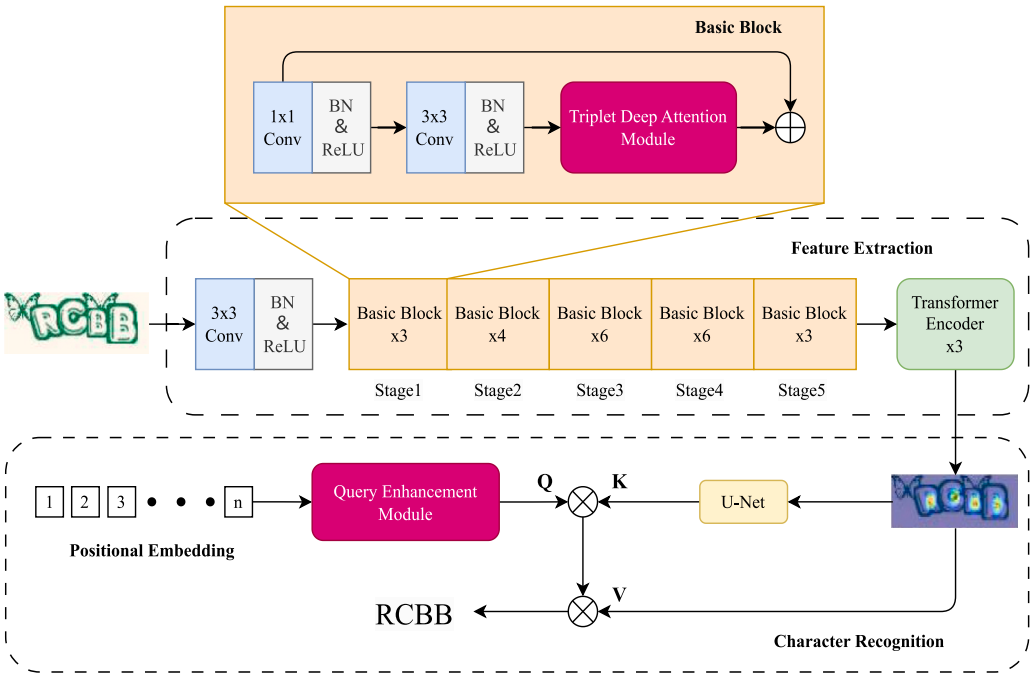

Fig. 1. The schematic illustration of our method for attacking text-based captchas. Using TDA and Transformer to extract features from images and generate feature maps. In the decoding stage, the query vector is initialized with positional encoding, and further abstraction of the query vector is performed through a query enhancement module. After self-attention, the target characters are obtained.

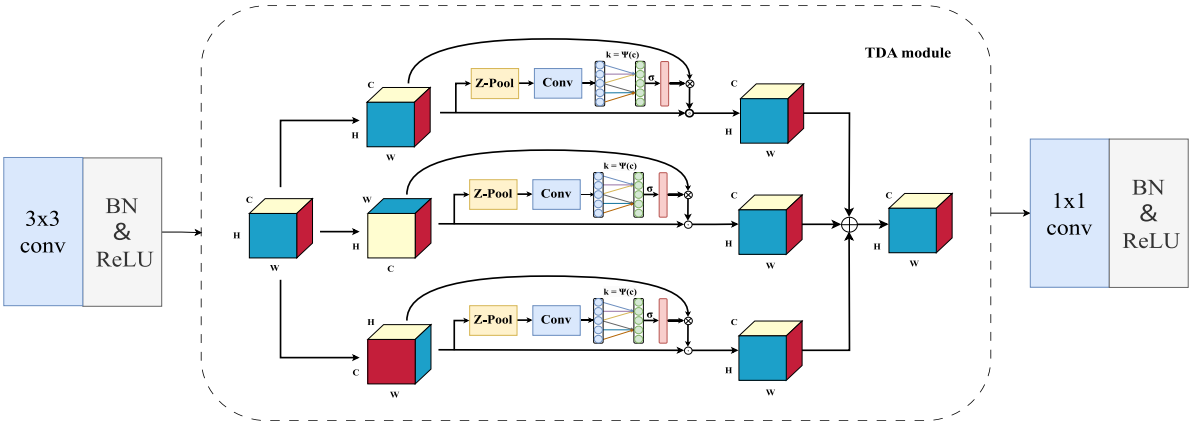

Fig. 2. TDA module.

3.1.1. ResNet45 with triplet deep attention module

The first part of feature extraction for text-based captcha images is ResNet45 with TDA module. ResNet45 consists of 1 convolutional layer with a kernel size of 3 and 5 stages. Each stage consists of 3, 4, 6, 6, and 3 basic blocks, where each basic block consists of a convolutional layer with a kernel size of 1, followed by a convolutional layer with a kernel size of 3. As shown in Fig. 2, the TDA module is located in each basic block of each stage in ResNet45 after 3×3 convolution.

TDA module follows the pipeline of triplet attention, comprising three parallel branches. The first branch is used to build spatial attention by capturing interactions between spatial dimensions and channel dimensions. The other two branches are responsible for capturing the spatial attention between the channel dimensions and the spatial dimensions H or W. Following attention capture, a fast one-dimensional convolution is implemented with an adaptable size of k to scale the second and third dimensions. In the end, the outputs of the three branches are aggregated by taking the average. The following outlines the forward process of TDA module:

Given a tensor $x \in \mathbb{R}^{C\times H\times W}$. Turn $x \in \mathbb{R}^{C\times H\times W}$ into $\hat{x}_1 \in \mathbb{R}^{W\times H\times C}$ and $x \in \mathbb{R}^{C\times H\times W}$ into $\hat{x}_2 \in \mathbb{R}^{H\times C\times W}$. Function Rot_{02} denotes exchanging the zeroth dimension with the second dimension. Function Rot_{01} denotes exchanging the zeroth dimension with the first dimension.

$$\hat{x}_0 = x \tag{1}$$

$$\hat{x}_1 = Rot_{02}(x), \hat{x}_2 = Rot_{01}(x) \tag{2}$$

Z-pool layer is employed to process the maximum pooling feature and the average pooling feature connecting that dimension in order to reduce the zeroth dimension of the tensor to two. The effect of this process is to make the network layer less deep while retaining a rich representation of the actual tensor, thus making further computation more lightweight. The operation of the *Z-pool* layer can be expressed by the following equation.

$$Z\text{-pool}(x) = [\text{MaxPool}_{0d}(x), \text{AvgPool}_{0d}(x)] \tag{3}$$

$$\hat{x^*}_{0,1,2} = Z\text{-pool}(\hat{x}_{0,1,2}) \tag{4}$$

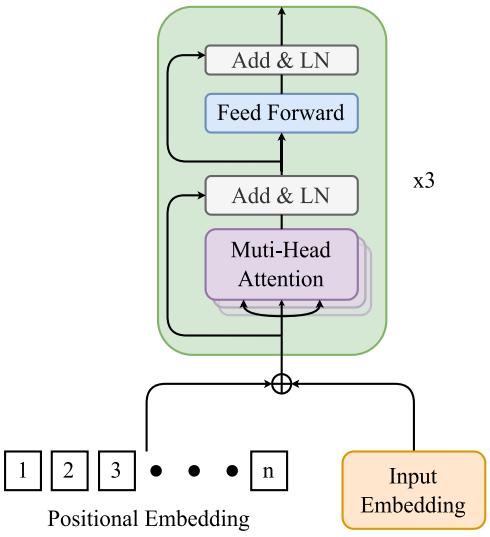

Fig. 3. Transformer encoder.

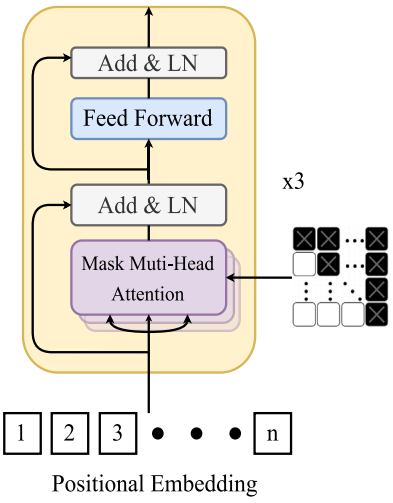

Fig. 4. Query enhancement.

After applying 7 × 7 convolution, one-dimensional convolution operation is carried out using kernel size of k. The weight y for each channel is derived through the activation function σ.

$$y_0 = \sigma(C1D_k(\hat{x_0^*})), \text{ where } \hat{x_0^*} = \left[\hat{x_0^*}^H, \hat{x_0^*}^W\right] \tag{5}$$

$$y_1 = \sigma(C1D_k(\hat{x_1^*})), \text{ where } \hat{x_1^*} = \left[\hat{x_1^*}^H, \hat{x_1^*}^C\right] \tag{6}$$

$$y_2 = \sigma(C1D_k(\hat{x_2^*})), \text{ where } \hat{x_2^*} = \left[\hat{x_2^*}^C, \hat{x_2^*}^W\right] \tag{7}$$

The size of the convolution kernel k is adaptively changed through a function $\psi(c)$.

$$\psi(c) = \left|\frac{\log_2(c)}{\gamma} + \frac{b}{\gamma}\right|, \text{ where } \gamma = 2,\ b = 1 \tag{8}$$

The final step involves restoring the rotated branch to its initial state, and obtaining the ultimate output result through a straightforward averaging operation.

$$\overline{y_1} = Rot_{02}(y_1), \overline{y_2} = Rot_{01}(y_2) \tag{9}$$

$$y = \frac{1}{3}(y_0 + \overline{y_1} + \overline{y_2}) \tag{10}$$

After undergoing feature extraction with ResNet45, y will be fed into the Transformer encoder for the second part of feature extraction.

3.1.2. Enhanced feature extraction use transformer encoder

The second part of feature extraction involves 3 sequentially composed Transformer encoding layers. As shown in Fig. 3, the input tensor sequence undergoes positional encoding, and is then added to the original sequence. Subsequently, correlations between sequences are calculated through 8 attention heads. Multi-head attention divides the parameters of query, key, and value into 8 parts, and then each part is processed independently by a separate attention head. The results from the 8 heads are combined to form the final attention scores. Following that, residual connections and layer normalization are applied. Lastly, the feature map is obtained through a feedforward network, followed by another round of residual connections and layer normalization. The entire Transformer encoding process utilizes the ReLU activation function.

3.2. Character recognition

The core of the character recognition module lies in the carefully crafted positional attention mechanism, namely the decoder, with the goal of transforming visual features into probabilities of character. It consists of two modules: design key and value in decoder and query enhancement.

3.2.1. Design key and value in decoder

After obtaining the feature map, in order to better transform visual features into probabilities of character, a mini-sized U-Net was added to the feature map, serving as the key for the attention mechanism. The U-Net's encoder and decoder each consist of 4 layers. The convolutional kernel size of the first encoder layer is 3, with a stride and padding of 1 and 2, respectively. For the remaining encoder layers, the convolutional kernel size is 3, with a stride and padding of 2 and 2. The upsampling process of the decoder layer employs nearest-neighbor interpolation, doubling both the height and width. A mini-sized U-Net operation is applied to key, while value undergoes an identity mapping. F_e represents feature map of encoder. μ denotes a mini-sized U-Net, and φ denotes an identity mapping. K and V denotes key and value.

$$K = \mu(F_e) \in R^{\frac{HW}{16} \times C} \tag{11}$$

$$V = \varphi(F_e) \in R^{\frac{HW}{16} \times C} \tag{12}$$

3.2.2. Query enhancement module

As shown in Fig. 4, the core of query enhancement module is a self-attention mechanism with upper triangular mask. Upper triangular mask design serves two purposes. The first purpose is to effectively reduce the computational cost associated with calculating blank padding areas by masking out ineffective padding regions, thereby improving overall computational efficiency. The second purpose is that upper triangular mask prevents the model from accessing future information, ensuring that attention module only attends to past and present information during processing. The following will explain the forward process of the query enhancement module, as well as how to use positional attention to transform visual features into character probabilities of character.

The initialization of query vectors involves using sinusoidal encoding, which is a type of relative encoding. Relative positional encoding is more conducive to the model's generalization compared to absolute positional encoding. B denotes batchsize. $\mathcal{T}_{\mathcal{M}}$ denotes mask self-attention and Q denotes query .

Table 2
Roman character-based schemes (px = pixel).

Scheme	Example	Length	Size(px)	Features	Characters Set	Train Set	Test Set
360_1		4~5	100*40	Noise arcs, varied fonts Intertwine, hollow	19	8,500	1,000
360_2		4~5	100*40	Gray foreground, rotation Intertwine, hollow	19	8,500	1,000
Alipay		4	100*30	Overlapping, distortion rotation	26	8,500	1,000
Apple		4~5	160*70	Gray background, rotation overlapping	28	8,500	1,000
Microsoft		4~6	216*128	Diagonal distribution Hollow, overlapping	40	8,500	1,000
QQmail		4	130*53	Interference patch, rotation Intertwine	20	8,500	1,000
Sina		5	100*40	Noise lines, hollow	25	8,500	1,000
Weibo		4	100*40	Distortion, Intertwine	37	8,500	1,000
Wiki		8~10	250*60	Distortion	36	8,500	1,000
SynCAPTCHA1		4~10	100*40	Varied fonts, rotation	62	100,000	20,000

Table 3
Chinese character-based schemes (px = pixel).

Scheme	Example	Length	Size(px)	Features	Characters set	Train set	Test set
Baidu		2	100*40	Noise arcs, rotation overlapping	949	10,000	1000
Douban		3~5	250*40	Complex background distortion, rotation	984	10,000	1000
Dajie		4	80*34	Noise arcs, warping rotation	2468	10,000	1000
It168		4	150*50	Noise arcs, rotation complex background	747	10,000	1000
Renmin		2	120*32	Complex background rotation	484	10,000	1000

$$PE(pos, 2i) = sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{13}$$

$$PE(pos, 2i+1) = cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}) \tag{14}$$

$$Q' = \{PE(pos, 2i), PE(pos, 2i+1)\} \in \mathbb{R}^{B\times 26\times 512} \tag{15}$$

$$Q = \mathcal{T}_{\mathcal{M}}(Q') \tag{16}$$

Implement self-attention on Q, K, and V to convert visual features into probabilities of character. Multiply Q by the transpose of K, then divide by $\sqrt{d_k}$, followed by a *softmax* operation, and multiply the result by V. The purpose of $\sqrt{d_k}$ is to scale the attention weights, aiming to enhance the stability and training effectiveness of the model.

$$Output(Q, K, V) = softmax(\frac{Q \cdot K^T}{\sqrt{d_k}})V \tag{17}$$

The *Output* signifies the predicted character sequence, and the cross-entropy function is subsequently employed to compute the distance between the *Output* and the ground truth. Model parameters are updated through back propagation. Through continuous training, model gradually acquires the ability to generate accurate results.

4. Experiments

4.1. Dataset and implementation details

The dataset used in the experiment is XDCAPTCHA (Wang et al., 2023a). The character set of CAPTCHA images in XDCAPTCHA includes two types: Roman and Chinese.

The Roman character-based dataset includes 15 real-world schemes and 1 synthetic scheme. For each real-world scheme, the training and testing sets of Roman character-based captcha images consist of 8500 and 1000 images. The synthetic dataset, SynCAPTCHA1, comprises 100,000 training images and 20,000 testing images. As shown in Table 2. In the Roman character-based setting, experiments are conducted on 9 real-world schemes and 1 synthetic scheme. The remaining 6 real-world schemes were removed from the experiments as the previous methods had already performed satisfactorily on them.

The Chinese character-based dataset includes 5 real-world schemes and 1 synthetic scheme. For each real-world scheme, the training and testing sets of Chinese character-based captcha images consist of 100,000 and 1000 images. As shown in Table 3. In the Chinese character-based setting, experiments are conducted on 5 real-world schemes. The remaining synthetic scheme, it was removed from the experiments for the same reasons as the 6 real-world schemas that were removed from the Roman character-based Experiments.

The text-based captcha images were preprocessed by resizing them to 32 × 128 dimensions and applying various geometric transformations, including rotation, affine transformations, and perspective operations. Model automatically performs the preprocessing steps, requiring no manual intervention for specific image processing. As shown in Table 5, the entire workflow of the method is executed on a workstation equipped with an NVIDIA GeForce RTX 3080 GPU (10 GB VRAM). During training, a batch size of 20 is utilized, and the model is trained for 200 epochs with a learning rate of 0.0001, utilizing the Adam optimizer to update model parameters.

Table 4
Comparison with existing Works.

Method	360_1	360_2	Alipay	Apple	Microsoft	QQmail	Sina	Weibo	Wiki	Avg
Wojna et al. (2017)[a]	70.8	72	96.6	78.4	77.2	80.4	85.2	86.8	87.8	81.68
Zi et al. (2019)	83	**84.3**	96.4	82.4	74.8	79	86.2	86	90.6	84.74
Wang et al. (2020a)	79.9	66.6	96.9	81.1	67.3	80.5	91.6	91.4	79.5	81.64
Tian and Xiong (2020)	–	–	–	68.4	56.6	–	84	88.2	87	–
Wang et al. (2020c)[a]	77.4	57.2	97.2	88.4	77.4	86.6	90.2	88.6	87.2	83.35
Li et al. (2021)	–	–	–	88.1	53.3	–	90	91	87.5	–
Nian et al. (2022)	67.6	–	–	80.8	70.2	75.6	**92.8**	–	88.8	–
Ours	**92.3**	81.4	**97.3**	**90.3**	**90.8**	**90.3**	91.9	**93.8**	**92.2**	**91.14**

Wang et al. (2023a) reproduced the experiments for capthca attack using the original methods.

[a] Denote the original method of reproduction.

Table 5
Experimental setting.

Setting	Value
CPU	i9-9900K 3.60 Hz
GPU	RTX 3080 (10 GB)
Batch size	20
Epoch	200
Learning rate	0.0001
Optimizer	Adam

4.2. Evaluation metric

To evaluate the effectiveness of the attack, character accuracy (CA) and word accuracy (WA) are used as metrics to assess the success of the attack. CA primarily focuses on the accuracy of each individual character, assessing the model's performance by comparing the generated text with each character's alignment in the target text. WA is more comprehensive, considering the correctness of entire words.

Specifically, N_w is used to denote the number of letters in a word, while N_c represents the number of correctly predicted characters. Additionally, the symbol N_a signifies the all number of words in the entire set, and $N_{w\prime}$ denotes the number of words correctly predicted.

$$Character\ Accuracy = \frac{N_c}{N_w} \times 100\% \tag{18}$$

$$Word\ Accuracy = \frac{N_{w\prime}}{N_a} \times 100\% \tag{19}$$

4.3. Result and comparisons with existing work

According to the results in Table 4, the method has achieved excellent results across 9 Roman real-world schemes, with average word accuracy of 91.14%. Model utilizes TDA and QE modules, with the TDA module proving effective in recognizing rotated captcha images. For instance, the word accuracy for Apple and Microsoft are 90.3% and 90.8%. In schemes like Apple, characterized by significant background noise, model accurately locates the positions of characters, achieving word accuracy of 90.3%. In schemes such as 360_2, characterized by significant foreground noise, the word accuracy is 81.4%, which is 2.9% lower compared to Zi et al. (2019).

There are two reasons why our method word accuracy is 2.9% lower than that of Zi et al.'s method. The first reason is that Zi et al. tested their method with 500 captcha images, which is 500 images less than our test set. Our results with 500 images for the test showed an accuracy of 83.0%, an improvement of 1.6% over the test with 1000 captcha images, but still 1.3% lower than the method of Zi et al. The second reason is that our test set is randomly split, while Zi et al. did not release their test set, causing potential disparities in data distribution between the two methods, resulting in word accuracy 1.3% lower than Zi et al. For other schemes, method is capable of accurately attacking target captcha images.

To validate the effectiveness of our method on other datasets, experiments were conducted on the dark web dataset provided by Zhang et al. (2022). The dark web dataset has additional background noise and variable character lengths. Each scheme was trained on a training set of 400 samples and tested on a set of 100 samples from the dark web dataset after retraining our method. The experimental setting conducted on the dark web dataset is the same as that on the XDCAPTCHA dataset.

Table 6
Experimental results on dark web dataset.

Scheme	Example	Tang et al. (2018)	Zhang et al. (2022)	Ours
Rescator1		88.12	94.40	**95.00**
Rescator2		77.23	97.50	**98.02**
Yellow Brick		93.72	95.98	**97.06**

Table 7
Impact of different branches.

BR_{CHW}	BR_{WHC}	BR_{HCW}	CA (%)	WA (%)
–	–	–	97.33	90.02
✓	–	–	97.52	90.44
–	✓	–	97.32	90.00
–	–	✓	97.26	90.28
✓	✓	–	97.45	90.40
✓	–	✓	97.43	90.32
–	✓	✓	97.42	90.32
✓	✓	✓	**97.56**	**90.65**

The experimental results on the dark web dataset are shown in Table 6. Our method performs well on the dark web dataset. For each scheme, Rescator1, Rescator2, and Yellow Brick, the word accuracy is 95%, 98.02%, and 97.06%, respectively, which represents an improvement of 0.6%, 0.52%, and 1.08% over Zhang et al. (2022).

4.4. Ablation study

Due to the integration of ECANet into the TDA module, it is necessary to assess the impact of ECANet on different branches. The experiment will conduct ablation experiments on each branch of the TDA module to determine the optimal number of branches, thus specifically evaluating the contribution of ECANet within the TDA module.

The results of the experiment are shown in the Table 7. In the single-branch case, BR_{CHW} performs the best, followed by BR_{HCW}, and BR_{WHC} ranks last. The addition of ECANet to the BR_{WHC} branch results in a decrease in performance. In the two-branch case, the parallel combination of BR_{CHW} and BR_{WHC} yields the best results, but it falls below the performance of BR_{CHW} in the single-branch scenario, indicating the dominance of the BR_{CHW} branch within the entire TDA. In the three-branch case, the experimental results are optimal, showing 0.63% improvement in TA performance with the inclusion of ECANet.

QE enhances the embedding by using a multi-headed self-attentive block from vanilla Transformer (Vaswani et al., 2017). A mask is employed on the query vector within the self-attention block, specifically in the upper triangular region. This is done to prevent it from "seeing

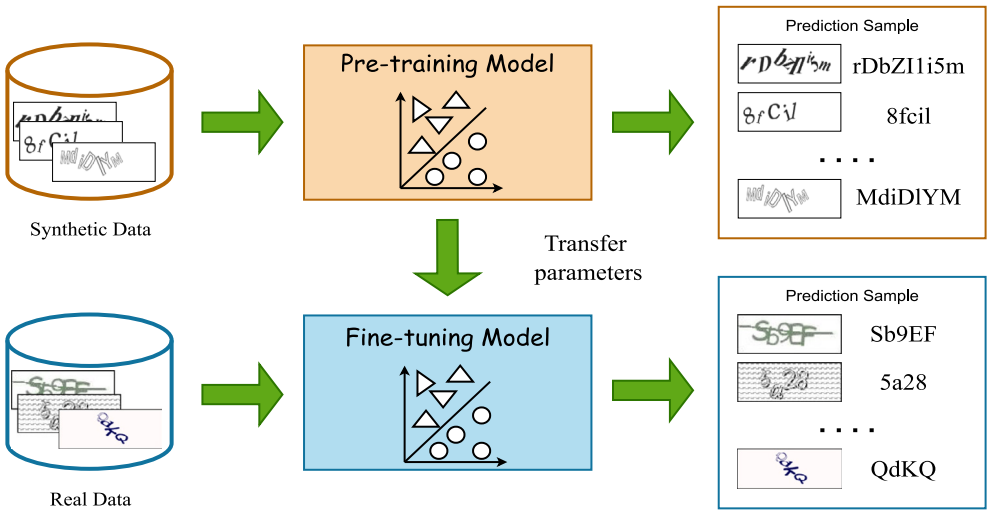

Fig. 5. The transfer learning strategy use training model using synthetic data to obtain a pre-trained model initially. Subsequently, real-world data is used to fine-tune the existing parameters of the pre-trained model.

Table 8
Performance of TDA and QE module.

Model	CA(%)	WA(%)
Baseline (Fang et al., 2021)	97.41	90.10
+ TDA	97.56	90.65
+ QE	97.44	90.17
+ TDA & QE	**97.65**	**91.14**

itself". In other words, this is done to avoid the leakage of information across different time steps (Zheng et al., 2024).

The results of the experiment are shown in the Table 8. Compared to the baseline, adding the QE module independently results in a slight improvement of 0.07%. When both the TDA and QE modules are simultaneously integrated, there is an average word accuracy improvement of 1.04%.

According to the results of ablation experiments, individually adding the TDA and QE modules does not enhance the method as significantly as adding both the TDA and QE modules simultaneously. The combined inclusion of these modules yields a greater improvement, demonstrating their synergistic effect on the overall performance of the method.

4.5. Method training time and attack speed

Training time and attack speed are key metrics for evaluating the effectiveness of the method. Next, we will analyze the training time and compare the performance of attack speed on CPU and GPU with and without the TDA module.

As shown in Fig. 6, the average running time of the method is given in parentheses below each scheme on the x-axis, where "m" represents minutes, the average time for one epoch. Microsoft has the longest training time, about 2.17 min per epoch, while Wiki has the shortest training time, about 1.27 min per epoch. The overall average time is 1.46 min per epoch.

In the Fig. 6, the blue line represents TDA is used, while the red line represents without TDA. Triangle represent model evaluation on the CPU, while pentagrams represent evaluation on the GPU. When the model attacks the captcha on the CPU, the average time with TDA is around 10.6 ms, without TDA it is around 8.2 ms, showing a difference of 2.4 ms. When the model attacks the captcha on the GPU, the average time with TDA is around 5.6 ms, without TDA it is around 4.8 ms, showing a difference of 0.8 ms. For captcha attack, this is an acceptable range.

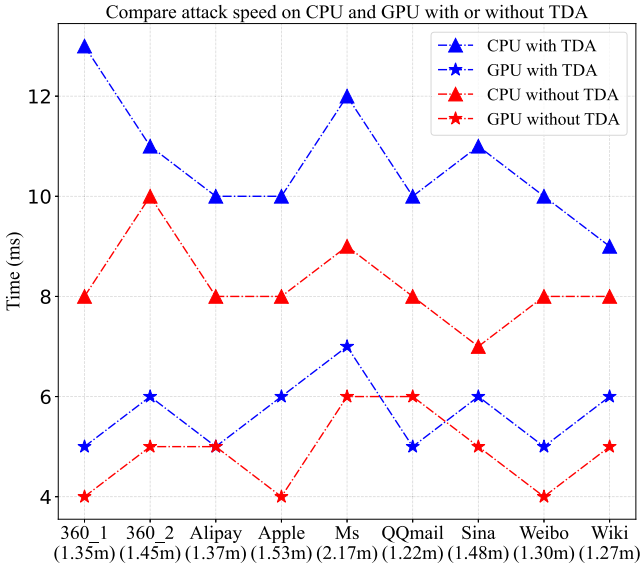

Fig. 6. Compare attack speed on CPU and GPU with and without TDA. Blue line represents TDA is used. Red line represents without TDA. Triangle represent model evaluation on the CPU. Pentagrams represent model evaluation on the GPU. The letter "m" represents for minutes of per epoch.

4.6. Transfer learning strategy

End-to-end attack methods, through clever design, fully utilized the model's performance on independent and sufficiently large datasets, demonstrating excellent experimental results. However, when the number of real datasets is limited, the results of training specifically for a particular dataset are often unsatisfactory. Therefore, researchers are exploring transfer learning strategies aimed at proposing a universal recognition model capable of effectively handling low-sample images. Transfer learning strategy pipeline and result are shown in Fig. 5 and Table 9. First, train the model using a synthetic dataset to obtain a pre-training model with pre-trained weights. Subsequently, fine-tune the model using real-world data.

More specifically, training a pre-trained model on a large amount of synthetic datasets for downstream tasks. Subsequently, fine-tuning is performed on real datasets to adapt to specific data distributions. We conducted 40 epochs of pre-training on the model using SynCAPTCHA1

Table 9
Transfer learning strategy result.

Scheme	Pre-trained model		Fine-tuned model							
	100,000		500		1000		2000		8500	
	CA(%)	WA(%)	CA(%)	WA(%)	CA(%)	WA(%)	CA(%)	WA(%)	CA(%)	WA(%)
360_1	2.30	0.00	88.62	61.20	93.73	77.10	96.10	84.70	**97.59**	**89.80**
360_2	4.10	0.00	85.78	54.30	90.29	65.50	93.55	75.10	**95.60**	**81.90**
Alipay	7.40	0.00	98.45	94.40	99.05	96.20	99.22	96.90	**99.40**	**97.60**
Apple	15.20	0.00	90.53	70.10	92.14	75.40	94.62	80.90	**96.13**	**87.10**
Microsoft	23.30	0.70	78.09	38.70	85.03	52.80	90.34	65.80	**95.13**	**81.10**
QQmail	4.50	0.00	84.72	58.30	92.23	76.60	94.87	84.90	**96.60**	**90.10**
Sina	18.00	0.20	96.04	87.40	96.18	86.80	96.98	90.30	**97.36**	**91.90**
Weibo	10.40	0.00	95.57	84.40	96.47	87.20	97.12	89.70	**97.98**	**92.50**
Wiki	13.00	0.00	97.62	82.60	98.09	86.60	98.35	88.20	**98.96**	**92.50**
Avg	10.91	0.10	90.60	70.15	93.69	78.24	95.68	84.05	**97.19**	**89.38**

as the training dataset. Following this, we fine-tuned the schemes based on Roman characters by employing 500, 1000, 2000, and 8500 real-world samples in 40 epochs.

Without fine-tuning, the word accuracy is only 0.10%. After fine-tuning with 500, 1000, 2000, and 8500 real samples for 40 epochs, the word accuracy are 70.15%, 78.24%, 84.05%, and 89.38%. The results obtained through the transfer learning strategy show an increase in word accuracy by 0.5%, 0.3%, and 0.3% on the 360_2, Alipay, and Wiki.

We adopted transfer learning strategy, utilizing only 5.8%, 8.5%, and 23.5% of the original dataset. This resulted in word accuracy reaching 76.9%, 85.8%, and 92.2% of the end-to-end training result. By fine-tuning with the complete dataset, we can elevate the word accuracy to 98% of the end-to-end training result. When facing a new dataset, simply fine-tuning a pre-trained model with the new dataset can achieve word accuracy close to that of fully trained models. This strategy saves retraining time and improves the model's generalization.

4.7. Visualization analysis

As shown in Fig. 8, by visualizing the feature maps of the model, we compare the roles of different attentions on various character components. We select 14 images for illustration, covering samples from 9 different datasets. Each image consists of two columns, where the first row of the first column represents the original image, and the first row of the second column indicates the attention positions of all characters. The remaining sub-images display the attention positions of individual characters, while the black and white images are used for a clearer contrast of attention ranges.

The last row displays samples of recognition errors. In the first example of the last row, due to the overlapping of the letters 'H' and 'B', the model incorrectly identifies 'B' as '3'. The same error occurs in the image 'MFCE,' where 'C' is mistakenly identified as 'Q'. Based on the above content, researchers can contemplate how to design captcha images with character adhesion to effectively prevent automated attacks by computer recognition programs.

Based on the visualization results, our method demonstrates accurate character localization and successful recognition even in the presence of occlusion, interference from non-character elements, and text rotation. However, the performance decreases when two characters overlap significantly.

4.8. Effects of lack of illumination

From the visualization results, it is evident that our method can recognize captcha images containing features such as occlusion, interference from non-character elements, and text rotation. However, most images have a white background, giving them a brighter appearance. When the background brightness is reduced, it is necessary to discuss whether our method can still successfully attack the captcha image under lack of illumination conditions.

We reduced the brightness of the training and test sets to half of the original images, while keeping the other training setting unchanged. The experimental results are shown in Table 10. WA_O represents the original word accuracy, which is the same as the value in the last row of Table 4. WA_L represents the word accuracy lack of illumination. The blue and red fonts indicate how much the word accuracy decreased and increased.

From the results in Table 10, we notice that reducing the brightness by half causes a decrease in word accuracy for all schemes except 360_2, Sina and Wiki. The largest decrease is observed in 360_1, which drops by 1.80%, while the smallest decrease is in Apple, which drops by only 0.30%. On average, the word accuracy decreases by 0.49%. However, in the 360_2, Sina and Wiki schemes, word accuracy increases by 0.20%, 0.50% and 0.40%. This indicates that reducing the brightness of the images decreases some specially designed interfering features under lack of illumination conditions. Overall, our method can still successfully attack most of the captcha images even when the brightness is reduced.

4.9. Result of adding Gaussian noise to the image

Based on word accuracy, 360_2 has the lowest word accuracy. From a defense perspective, adding background noise to the image can reduce the accuracy of attacks. Therefore, we added Gaussian noise (Ho et al., 2020) to the images and conducted experiments on them. The noise formula $g(x)$ is as follows, where $\mu = 0$ and $\sigma = 30$. By adding the noise to the original image, the noisy image is obtained. The experimental results are shown in Table 11.

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right), \mu = 0, \sigma = 30 \tag{20}$$

Based on the experimental results in Table 11, we find that the average word accuracy decreases by 2.18% after adding Gaussian noise. The 360_1 scheme experiences the largest drop, with a decrease of 4.30%, while the Sina scheme sees the smallest decrease, at 0.50%. In the 360_1 scheme, the Gaussian noise blurs the main features of the characters, reducing the attack accuracy. The accuracy of both the 360_2 and Microsoft schemes decreased by 4%. The 360_2 scheme already has significant background noise, and the added Gaussian noise makes some characters indistinguishable even to humans. For other schemes, Gaussian noise reduced the success rate of attacks to varying degrees.

4.10. Attacks on Chinese text captcha images

Chinese websites typically use Chinese text captchas to distinguish between users and computer programs, ensuring that only real human users can access the site. Experiments were conducted to explore the performance of method in handling Chinese text captcha images. In attacks on Chinese text captcha experiments, the same experimental environment and setting as the Roman character dataset are adopted.

Table 10
Word accuracy under lack of illumination.

Scheme	Example	WA_O(%)	WA_L(%)
360_1		92.30	90.50 −1.80
360_2		81.40	81.60 +0.20
Alipay		97.30	96.90 −0.40
Apple		90.30	90.00 −0.30
Microsoft		90.80	89.40 −0.40
QQmail		90.30	89.40 −0.90
Sina		91.90	92.40 +0.50
Weibo		93.80	93.10 −0.70
Wiki		92.20	92.60 +0.40
Avg	-	91.14	90.65 −0.49

WA_O represents the original word accuracy. WA_L represents the word accuracy lack of illumination. The blue and red fonts indicate how much the word accuracy decreased and increased.

Table 11
Word accuracy after adding Gaussian noise to the image.

Scheme	Example	WA_O(%)	WA_G(%)
360_1		92.30	88.00 −4.30
360_2		81.40	77.40 −4.00
Alipay		97.30	96.50 −0.80
Apple		90.30	88.20 −2.10
Microsoft		90.80	86.80 −4.00
QQmail		90.30	88.50 −1.80
Sina		91.90	91.40 −0.50
Weibo		93.80	93.00 −0.80
Wiki		92.20	90.90 −1.30
Avg	-	91.14	88.96 −2.18

WA_O represents the original word accuracy. WA_G represents the word accuracy after adding Gaussian noise. The blue fonts indicate how much the word accuracy decreased.

Table 12
Attacks on Chinese character-based dataset result.

Scheme	Wojna et al. (2017)[a]	Wang et al. (2020c)[a]	Ours
Baidu	76.80	91.80	**98.30**
Douban	97.40	98.00	**99.90**
Dajie	94.60	98.60	**99.90**
It168	98.80	97.80	**100.00**
Remin	96.20	99.80	**99.90**
Avg	92.76	97.20	**99.60**

Wang et al. (2023a) reproduced the experiments for capthca attack using the original methods.
[a] Denote the original method of reproduction.

The English character set has been changed to the Chinese character set, which includes 6280 different Chinese characters. The results are shown in Table 12. The average accuracy of words in results is 99.60%.

4.11. Performance in scene text recognition

To evaluate the generalization of our method, we conduct experiments on the scene text recognition task. The training and test datasets are provided by Baek et al. (2021). Examples of images from the training and test sets are shown in Fig. 7. The training set includes SVT, IC13, IC15, IIIT, COCO, RCTW17, Uber, ArT, LSVT, MLT19, and ReCTS, totaling 278,388 images. The test set includes IC13, IC15, IIIT, and CUTE80, with 1015, 2077, 3000, and 288 images. The experiment trained for a total of 8 epochs, and the other experimental settings remain the same as before.

Fig. 7. Some examples from the scene text recognition test datasets.

Table 13
Recognition results on the scene text datasets.

Scheme	CRNN[a] (Shi et al., 2016)	TRBA[a] (Baek et al., 2019)	Ours
IC13	86.3	92.6	**94.5**
IC15	62.2	**76.0**	73.1
IIIT	83.5	93.5	**93.9**
CUTE	64.7	86.1	**88.5**
Avg	74.2	87.1	**87.5**

Baek et al. (2021) reproduced the experiments using real-world training datasets.
[a] Denote the original method of reproduction.

The recognition results on the scene text datasets are shown in Table 13. Our word accuracy on IC13, IIIT, and CUTE datasets is higher than TRBA (Baek et al., 2019) by 1.9%, 0.4%, and 2.4%, respectively. However, our results on the IC15 dataset are 2.9% lower than TRBA. From the experimental results, we can conclude that our method has better recognition performance for rotated text, but it does not perform well in situations where background features are inconsistent.

5. Countermeasures

To counter our attacks, we provide some implementation suggestions to ensure system safety.

For capthca images, most text-based capthca images currently have relatively clean backgrounds. To reduce word accuracy, researchers can consider decreasing the image brightness or adding Gaussian noise.

Our attack method operates at a faster speed. In contrast, human verification of captcha requires considerably more time spent typing on a keyboard, a process that far exceeds our attack speed. Researchers leverage this behavioral characteristic to impose constraints during captcha verification. For instance, recognition speeds surpassing a certain threshold could indicate automated processes. Furthermore, keystrokes on the keyboard can be detected by the system, enabling the addition of keystroke detection (Wang et al., 2023a) to distinguish human users.

Increasing the difficulty of obtaining captcha. Our method requires a sufficient number of images for training, so researchers can consider making it harder to obtain captcha images, thereby reducing the number of images acquired through mechanisms like web crawlers. For example, each IP address could be limited to a certain number of captcha image requests within a given time frame. This would not only decrease the likelihood of web crawlers acquiring captcha images in bulk but also prevent malicious users from abusing system resources.

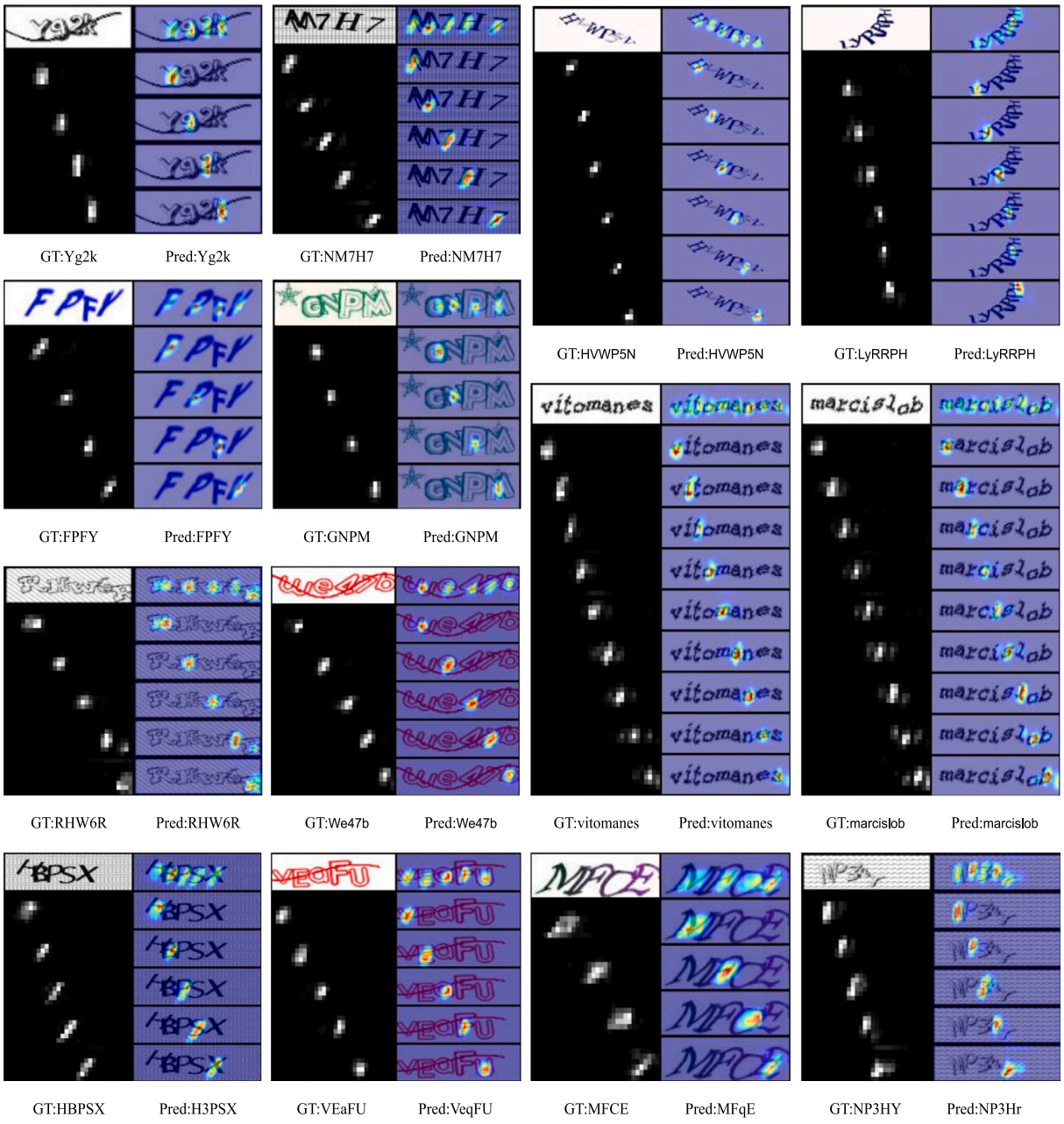

Fig. 8. Selected various samples from 9 Roman character schemes and visualized their feature maps. The last row displays the incorrect results of the attack.

6. Conclusion and future work

This study extensively investigates the limitations in feature extraction and character localization of current text-based captcha recognition methods. We propose a triple deep attention module designed to efficiently extract character features by extracting cross-dimensional features in both spatial and channel dimensions. In our method, we adopt query enhancement module to enhance the positional information of the characters and reduce the attentional drift. With these improvements, the model is able to accurately locate the detailed features of characters, resulting in a significant increase in recognition accuracy and more satisfactory results. We reduced training samples by employing transfer learning strategy. Additionally, our method exhibited satisfactory performance when utilized on Chinese text-based captcha dataset. We further explored the method's effectiveness under conditions of reduced brightness and added Gaussian noise in images. Since manually annotating captcha images is time-consuming, we will explore how to apply our method using semi-supervised approach in the future.

CRediT authorship contribution statement

Bo Zhang: Writing – review & editing, Writing – original draft, Visualization. **Yu-Jie Xiong:** Supervision. **Chunming Xia:** Resources. **Yongbin Gao:** Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62006150); Science and Technology Commission of Shanghai Municipality, China (21DZ2203100).

References

Aggarwal, A., Mittal, M., Battineni, G., 2021. Generative adversarial network: An overview of theory and applications. Int. J. Inf. Manage. Data Insights 1 (1), 100004.

Baek, J., Kim, G., Lee, J., Park, S., Han, D., Yun, S., Oh, S.J., Lee, H., 2019. What is wrong with scene text recognition model comparisons? dataset and model analysis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4715–4723.

Baek, J., Matsui, Y., Aizawa, K., 2021. What if we only use real datasets for scene text recognition? toward scene text recognition with fewer labels. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3113–3122.

Cahuantzi, R., Chen, X., Güttel, S., 2023. A comparison of LSTM and GRU networks for learning symbolic sequences. In: Science and Information Conference. Springer, pp. 771–785.

Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H., 2019. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S., 2020. End-to-end object detection with transformers. In: European Conference on Computer Vision. Springer, pp. 213–229.

Chandra, M.A., Bedi, S., 2021. Survey on SVM and their application in image classification. Int. J. Inf. Technol. 13, 1–11.

Chen, J., Luo, X., Hu, J., Ye, D., Gong, D., 2018. An attack on hollow captcha using accurate filling and nonredundant merging. IETE Tech. Rev. 35 (sup1), 106–118.

Chen, J., Luo, X., Liu, Y., Wang, J., Ma, Y., 2019. Selective learning confusion class for text-based CAPTCHA recognition. IEEE Access 7, 22246–22259. http://dx.doi.org/10.1109/access.2019.2899044.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Dou, Z., 2021. The text captcha solver: A convolutional recurrent neural network-based approach. In: 2021 International Conference on Big Data Analysis and Computer Science. BDACS, IEEE, pp. 273–283.

Fan, Q., Zhuo, W., Tang, C.-K., Tai, Y.-W., 2020. Few-shot object detection with attention-RPN and multi-relation detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4013–4022.

Fang, S., Xie, H., Wang, Y., Mao, Z., Zhang, Y., 2021. Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7098–7107.

Finn, C., Abbeel, P., Levine, S., 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In: International Conference on Machine Learning. PMLR, pp. 1126–1135.

Girshick, R., 2015. Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1440–1448.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2961–2969.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.

Ho, J., Jain, A., Abbeel, P., 2020. Denoising diffusion probabilistic models. arXiv preprint arXiv:2006.11239.

Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7132–7141.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25.

Li, C., Chen, X., Wang, H., Wang, P., Zhang, Y., Wang, W., 2021. End-to-end attack on text-based CAPTCHAs based on cycle-consistent generative adversarial network. Neurocomputing 433, 223–236.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022.

Ma, Y., Zhong, G., Liu, W., Sun, J., Huang, K., 2020. Neural CAPTCHA networks. Appl. Soft Comput. 97, 106769.

Misra, D., Nalamada, T., Arasanipalai, A.U., Hou, Q., 2021. Rotate to attend: Convolutional triplet attention module. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3139–3148.

Nian, J., Wang, P., Gao, H., Guo, X., 2022. A deep learning-based attack on text CAPTCHAs by using object detection techniques. IET Inf. Secur. 16 (2), 97–110.

Oord, A.v.d., Li, Y., Vinyals, O., 2018. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

Shi, B., Bai, X., Yao, C., 2016. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Trans. Pattern Anal. Mach. Intell. 39 (11), 2298–2304.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Snell, J., Swersky, K., Zemel, R., 2017. Prototypical networks for few-shot learning. Adv. Neural Inf. Process. Syst. 30.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826.

Tang, M., Gao, H., Zhang, Y., Liu, Y., Zhang, P., Wang, P., 2018. Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. IEEE Trans. Inf. Forensics Secur. 13 (10), 2522–2537.

Tian, S., Xiong, T., 2020. A generic solver combining unsupervised learning and representation learning for breaking text-based captchas. In: Proceedings of the Web Conference 2020. pp. 860–871.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.

Von Ahn, L., Blum, M., Hopper, N.J., Langford, J., 2003. CAPTCHA: Using hard AI problems for security. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22. Springer, pp. 294–311.

Wang, P., Gao, H., Guo, X., Xiao, C., Qi, F., Yan, Z., 2023a. An experimental investigation of text-based CAPTCHA attacks and their robustness. ACM Comput. Surv. 55 (9), 1–38.

Wang, P., Gao, H., Rao, Q., Luo, S., Yuan, Z., Shi, Z., 2021a. A security analysis of captchas with large character sets. IEEE Trans. Dependable Secure Comput. 18 (6), 2953–2968. http://dx.doi.org/10.1109/TDSC.2020.2971477.

Wang, P., Gao, H., Shi, Z., Yuan, Z., Hu, J., 2020a. Simple and easy: Transfer learning-based attacks to text CAPTCHA. IEEE Access 8, 59044–59058.

Wang, J., Li, X., Li, J., Sun, Q., Wang, H., 2022. NGCU: A new RNN model for time-series data prediction. Big Data Res. 27, 100296.

Wang, Y., Wei, Y., Zhang, Y., Jin, C., Xin, G., Wang, B., 2023b. Few-shot learning in realistic settings for text CAPTCHA recognition. Neural Comput. Appl. 35 (15), 10751–10764.

Wang, Q., Wu, B., Zhu, P.e., Li, P., Zuo, W., Hu, Q., 2020b. ECA-Net: Efficient channel attention for deep convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11534–11542.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L., 2021b. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 568–578.

Wang, T., Zhu, Y., Jin, L., Luo, C., Chen, X., Wu, Y., Wang, Q., Cai, M., 2020c. Decoupled attention network for text recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 12216–12224. http://dx.doi.org/10.1609/aaai.v34i07.6903.

Wojna, Z., Gorban, A.N., Lee, D.-S., Murphy, K., Yu, Q., Li, Y., Ibarz, J., 2017. Attention-based extraction of structured information from street view imagery. In: 2017 14th IAPR International Conference on Document Analysis and Recognition. ICDAR, http://dx.doi.org/10.1109/icdar.2017.143.

Woo, S., Park, J., Lee, J.-Y., Kweon, I.S., 2018. CBAM: Convolutional block attention module. In: Computer Vision – ECCV 2018. In: Lecture Notes in Computer Science, pp. 3–19. http://dx.doi.org/10.1007/978-3-030-01234-2_1.

Xu, X., Liu, L., Li, B., 2020. A survey of CAPTCHA technologies to distinguish between human and computer. Neurocomputing 408, 292–307.

Yu, Y., Si, X., Hu, C., Zhang, J., 2019. A review of recurrent neural networks: LSTM cells and network architectures. Neural Comput. 31 (7), 1235–1270.

Yusuf, M.O., Srivastava, D., Singh, D., Rathor, V.S., 2023. Multiview deep learning-based attack to break text-CAPTCHAs. Int. J. Mach. Learn. Cybern. 14 (3), 959–972.

Zhang, N., Ebrahimi, M., Li, W., Chen, H., 2022. Counteracting dark web text-based CAPTCHA with generative adversarial learning for proactive cyber threat intelligence. ACM Trans. Manage. Inf. Syst. (TMIS) 13 (2), 1–21.

Zhang, K., Wu, F., Sun, H., Cai, M., 2023. Monocular vehicle speed detection based on improved YOLOX and DeepSORT. Neural Comput. Appl. 1–18.

Zheng, T., Chen, Z., Fang, S., Xie, H., Jiang, Y.-G., 2024. CDistNet: Perceiving multi-domain character distance for robust text recognition. Int. J. Comput. Vis. 132 (2), 300–318.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J., 2020. Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159.

Zi, Y., Gao, H., Cheng, Z., Liu, Y., 2019. An end-to-end attack on text captchas. IEEE Trans. Inf. Forensics Secur. 15, 753–766.