International Journal of Engineering and Technology Innovation, vol. 15, no. 2, 2025, pp. 136-151

FEGAN: A High-Performance Font Enhancement Network for Text CAPTCHA Preprocessing

Xing Wan[1,2], Fazlina Ahmat Ruslan[2], Juliana Johari[2,*]

[1]School of Intelligent Manufacturing, Leshan Vocational and Technical College, Leshan, Sichuan, China

[2]School of Electrical Engineering, Universiti Teknologi MARA (UiTM), Shah Alam, Selangor, Malaysia

Received 07 July 2024; received in revised form 24 October 2024; accepted 29 October 2024

DOI: https://doi.org/10.46604/ijeti.2024.13977

Abstract

This study aims to address performance deficiencies in CAPTCHA preprocessing methods that impede the accurate recognition of text CAPTCHAs, which are crucial for identifying security vulnerabilities. To improve CAPTCHA preprocessing methods, a similar font is initially searched and acquired by manually removing obstructing pixels from a target CAPTCHA and retaining the font part. Using the found font, a pseudo-dataset is generated containing a large number of clean and dirty pairs to train to the proposed supervised Font Enhancement Generative Adversarial Network (FEGAN), which is designed to effectively eliminate non-font-related interferences and preserve the font outlines. Test results show that FEGAN can improve the recognizer's accuracy by approximately 16% to 50% on the M-CAPTCHA dataset (a publicly available dataset on Kaggle) and 5% to 35% on the P-CAPTCHA dataset (generated using the Python ImageCaptcha package), substantially outperforming the Multiview-filtering-based preprocessing approach.

Keywords: CAPTCHA, recognition, font enhancement, GAN

1. Introduction

Text CAPTCHA is still the most widely used CAPTCHA security scheme and is deployed on websites worldwide. The security of text CAPTCHAs is being increasingly challenged due to the development of deep learning techniques. CAPTCHA breaking is an effective means to study security mechanisms and help researchers and companies develop more secure CAPTCHAs. These CAPTCHAs are resistant to various attacks [1]. Besides, the CAPTCHA recognition algorithm is also a vital tool for system security assessment and vulnerability discovery. Therefore, highly efficient recognition algorithms based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are generally used to break text CAPTCHAs for security evaluation. However, CAPTCHA recognition differs from general image recognition tasks because its image contains many specially designed anti-attack security mechanisms. Consequently, a substantial volume of data is required to train a recognizer to attain a high recognition rate.

However, due to the existence of anti-crawling mechanisms, it is difficult to obtain enough images to train a model [2]. To enhance the accuracy of CAPTCHA identification when there are only a few images available, data preprocessing approaches are employed to combat the anti-attack measures present in the CAPTCHAs. Text CAPTCHA recognition differs from object detection in that many commonly used data preprocessing approaches are not applicable. As an illustration, mixup combines distinct images [3], PatchUp combines intermediate feature layers directly [4], and Random Erasing randomly

* Corresponding author. E-mail address: julia893@uitm.edu.my

English language proofreader: Chih-Wen Teng

removes certain regions of an image [5]. These preprocessing approaches either distort the font outlines or alter the semantic information by rearranging the character order in text CAPTCHAs. In addition, many filtering-based methods, on the other hand, are usually effective only for Gaussian and pretzel noise, which are not very effective for CAPTCHA preprocessing.

A text CAPTCHA typically comprises three layers: a background layer, a foreground layer, and a font layer [6]. Only the font provides legitimate information for a text CAPTCHA, while other patterns like background, texture, color, and interferences are deceptive designs that impede the model's effective learning. It is crucial to keep only the font component, remove all other pixels from the CAPTCHA image, and teach the model to distinguish between fonts and other anti-attack designs in the CAPTCHA.

2. Related Works

The CAPTCHA recognizer is responsible for identifying the text within a text CAPTCHA using a classification network. Currently, the prevailing approach for recognizing CAPTCHAs is based on deep learning models. These models often employ CNNs and RNNs due to the small size of the CAPTCHA image and the need for mobile deployment.

In 2020, Noury and Rezaei [7] introduced DeepCAPTCHA, employing a CNN for feature extraction and a fully connected layer with numerous Softmax functions for final character predictions, as illustrated in Fig. 1, achieving a recognition accuracy of over 98% on evaluated datasets. It comprises several convolutional layers, each succeeded by a MaxPooling layer. After the layers, a dense layer with a dropout rate of 30% is implemented, followed by parallel Softmax layers that encode outputs as predictions, each corresponding to a distinct character in the CAPTCHA. The model employs a binary cross-entropy loss function and is trained using the Adam optimizer, which has been shown to achieve faster convergence and better performance.

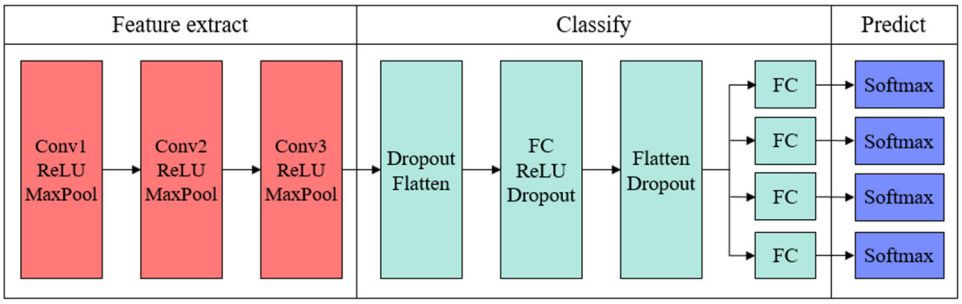

Fig. 1 The architecture of DeepCAPTCHA

The AdaptiveCAPTCHA model incorporates a CNN combined with an RNN (CRNN) module and Adaptive Fusion Filtering Networks (AFFN) for enhanced text CAPTCHA recognition, which was proposed by Wan et al. [8] in 2024. The CNN extracts features from the image, while the CRNN module captures character dependencies, improving recognition accuracy, as shown in Fig. 2. The AFFN module, composed of nested autoencoder filter units, effectively mitigates noise and interference. Notably, AdaptiveCAPTCHA incorporates residual connections to enhance gradient propagation and further enhance model robustness. This architecture minimizes the number of parameters and improves training speed while achieving high accuracy, making AdaptiveCAPTCHA a more efficient and robust text CAPTCHA solver.

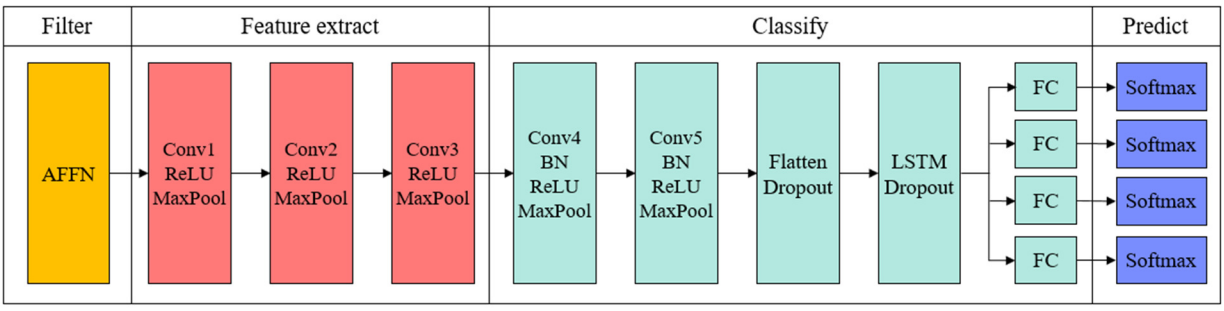

Fig. 2 The architecture of AdaptiveCAPTCHA

Qing and Zhang [9] used ConvNet to solve the text recognition problem, achieving over 90% accuracy on all the datasets they evaluated. The model utilizes a multi-kernel convolutional layer, which extracts multi-scale spatial features, followed by a series of 3 × 3 convolutional blocks and a predictive convolutional layer to extract features and predict the characters, as shown in Fig. 3.

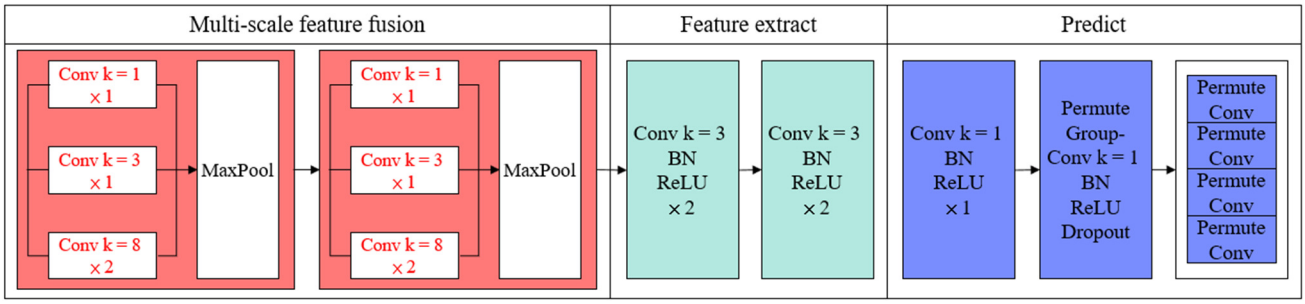

Fig. 3 The architecture of ConvNet

Mocanu et al. [10] introduced Capsule Networks (CapsNet), each representing a vector that encodes spatial properties and the presence probability of a visual entity. CapsNet utilizes a dynamic routing algorithm to route the output vectors from lower-level capsules to higher-level capsules, allowing for the learning of spatial relationships between features. This routing algorithm, which iteratively updates the coupling coefficients between capsules, enables the model to perform well on tasks involving spatial transformations, such as CAPTCHA recognition.

Ding et al. [11] proposed a novel method for image-based verification code generation and recognition, named VeriBypasser. It utilizes a CNN-based deep learning model to bypass three types of CAPTCHAs: visual reasoning, slider, and inference puzzle CAPTCHAs. VeriBypasser demonstrates high accuracy (85.20%) in recognizing and bypassing these codes, demonstrating its effectiveness in challenging scenarios. Table 1 summarizes the breaking models mentioned above.

Table 1 Summary of text CAPTCHA recognition networks

Model	Authors	Year	Description
DeepCAPTCHA	Noury and Rezaei [7]	2020	Using multiple Softmax layers to predict
CapsNet	Mocanu et al. [10]	2022	Introducing capsule networks for better geometric deformations
ConvNet	Qing and Zhang [9]	2022	Employing ConvNet and group convolution to extract neighboring information
VeriBypasser	Ding et al. [11]	2024	Adopting a CNN-based deep learning model to bypass verification codes
AdaptiveCAPTCHA	Wan et al. [8]	2024	Utilizing adaptive filters and RCNN to reduce interference

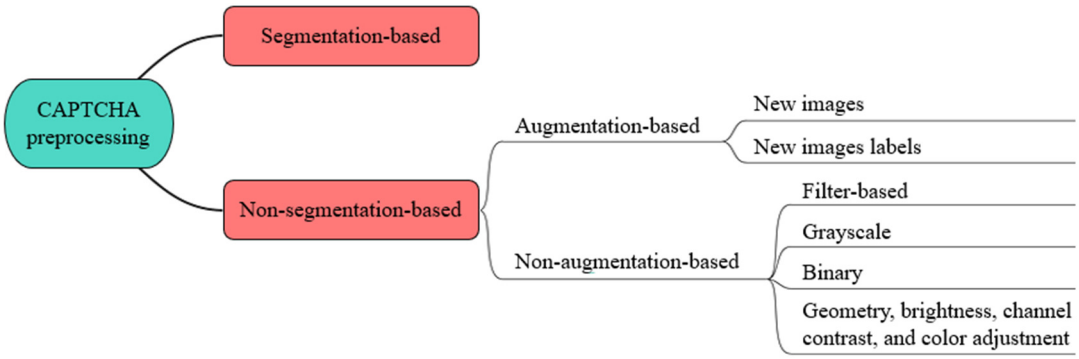

Fig. 4 Text CAPTCHA preprocessing methods

Most recognizers struggle to effectively navigate the intricate anti-attack measures incorporated in complicated CAPTCHA because the intricate nature of these traits can often result in the phenomenon of model overfitting. In addition to the CAPTCHA recognizer, text CAPTCHA preprocessing methods are widely used, which are classified into two categories: segmentation-based approaches and non-segmentation-based methods, as shown in Fig. 4. The excessive occurrence of font

sticking in CAPTCHA has limited the applicability of segmentation-based approaches, leading to their rare employment. Non-segmentation-based methods can be further categorized into augmentation-based methods and non-augmentation-based methods. The former generates new synthetic images based on real images to train the generator. Non-augmentation methods enhance the attacker's ability to identify the CAPTCHA without synthetic images. Two non-augmentation methods frequently employed for picture denoising are filter-based methods and grayscale and binarization techniques.

In 2018, Liu et al. [12] adopted conditional deep convolutional generative adversarial networks (cDCGAN) and CNN to solve small sample problems, which made tremendous progress in accuracy. Their method improves and fine-tunes the base solver using only a small number of labeled real CAPTCHAs. Thobhani et al. [13] utilized the provided binary as the data label for the text CAPTCHA, transforming recognizing several characters into a problem of recognizing a single character. Li et al. [14] proposed an improved model based on cycle-consistent generative adversarial networks (Cycle-GANs) in 2021, which has better transferability. Compared to other models, their method greatly reduces the cost of data labeling. By changing a few setup parameters, it can attack common text-based CAPTCHA schemes, making the assault simpler to carry out.

In the same year, Wang et al. [15] proposed a CAPTCHA solver that can effectively break text-based CAPTCHAs with complex security features using a small amount of labeled data. This method used Cycle-GAN to simplify the CAPTCHA images, resulting in 96%-character accuracy and 74% CAPTCHA accuracy for all evaluated schemes. Kimbrough et al. [16] enhanced the technique by positioning the affixed picture label at the lowermost part of the image. Some augmentation-based preprocessing methods for training CAPTCHA generation are summarized in Table 2.

Table 2 Augmentation-based preprocessing methods for text CAPTCHA

Method	Authors	Year	Description
cDCGAN	Liu et al. [12]	2018	Using a conditional GAN to generate new samples
Attached binary images	Thobhani et al. [13]	2020	Employing attached binary images as labels for every character
Cycle-GAN	Li et al. [14]	2021	Utilizing Cycle-GAN to generate similar training CAPTCHAs
Cycle-GAN	Wang et al. [15]	2021	Using Cycle-GAN to generate similar training CAPTCHAs
Encapsulated preprocessing	Kimbrough et al. [16]	2022	Adopting attached binary images as labels for every character

However, preprocessing methods based on data augmentation are time-consuming, labor-intensive, and difficult to implement, requiring extensive labeling of the generated data or constant tuning of the model based on the target CAPTCHA dataset to generate CAPTCHAs that approximate the real distribution.

In 2020, Ye et al. [17] used a GAN-based synthesizer to automatically generate training CAPTCHAs to construct a base solver and adopt another Pix2Pix model to remove noise. Zhang et al. [18] proposed Dark Web generative adversarial network (DW-GAN) in 2022, a GAN-based method that utilizes background denoising, character segmentation, and character recognition to automatically break dark web CAPTCHA with an over 92.08% success rate on dark web serval datasets. Yusuf et al. [19] employed a Multiview-filtering strategy to improve the performance of the recognizer. This approach involves using various filtering operators to increase the variety of the input images. Ishkov and Terekhov [20] employed the technique of appropriate color channel mixing to enhance the recognition efficacy of the model. Some non-augmentation-based preprocessing methods for training CAPTCHA generation are summarized in Table 3.

Table 3 Non-augmentation-based preprocessing methods for text CAPTCHA

Method	Authors	Year	Description
Pix2Pix	Ye et al. [17]	2020	Using the Pix2Pix to reduce the noise of CAPTCHAs
DW-GAN	Zhang et al. [18]	2022	Employing DW-GAN to denoise CAPTCHAs
Multiview-filtering	Yusuf et al. [19]	2022	Adopting multiple filtering to preprocess CAPTCHAs
Color channel combination	Ishkov and Terekhov [20]	2022	Leveraging a learnable linear combination of RGB channels
Pix2Pix	Ye et al. [17]	2020	Using the Pix2Pix to reduce the noise of CAPTCHAs

Non-data augmentation methods are more efficient to implement and are suitable for the vast majority of scenarios, yet existing methods have their limitations. Filtering-based methods are mainly for Gaussian noise and pretzel noise, while traditional GAN-based methods, which are too simple in loss function design, generally use adversarial loss and time-domain reconstruction loss.

Apart from preprocessing methods, evaluating the impact is a crucial aspect. However, there is no specific evaluation index for CAPTCHA preprocessing. For image quality evaluation, there are subjective and objective methods [21]. The subjective method involves directly observing the pictures before and after preprocessing with the naked eye to determine the effective removal of noise and interference. Objective methods encompass the Peak Signal-to-Noise Ratio (PSNR), where a higher value indicates superior image quality and reduced noise. This method is solely for noise preprocessing, as CAPTCHA contains not only noise but also various interferences and background patterns. Structural Similarity (SSIM) quantifies the extent of similarity between two images; a value approaching one indicates greater similarity between the images. But this method is also not applicable to CAPTCHA because the similarity between the image after removing the background and the clean image is usually low. In general, the efficacy of CAPTCHA preprocessing is evaluated based on the accuracy of recognition.

This paper presents a font enhancement method for text CAPTCHA recognition, comprising three primary contributions. First, a novel approach called font enhancement for text CAPTCHA recognition is introduced. Second, a supervised font enhancement network, termed Font Enhancement Generative Adversarial Network (FEGAN), is proposed. Finally, the accuracy of contemporary attack models integrated with FEGAN was assessed.

3. Methodology

The objective of text CAPTCHA breaking is to accurately identify the characters within each CAPTCHA. Any elements of the image design, such as background, texture, style, color, etc., that are not related to the font, serve as anti-attack measures that impede the correct recognition of the content by models. This paper introduces a novel FEGAN network aimed at enhancing the font contour and eliminating unnecessary distractions in CAPTCHA images. This is achieved by selectively removing all unrelated pixels from a single image in the target dataset while preserving the letter outlines. The implementation of font improvement significantly reduces the effectiveness of anti-attack systems in images, hence enhancing the accuracy of subsequent model identification. The entire process is segmented into three phases: producing training CAPTCHA pairs, training FEGAN, and training the recognizer.

3.1. Generating pseudo-target dataset

The first stage is to generate a certain number of training CAPTCHA pairs based on the similar fonts of the images in the target dataset. Due to the presence of many interference lines and other anti-attack designs in target CAPTCHAs, it is not possible to detect the font in a CAPTCHA image directly without any treatment. Too many distractions in the CAPTCHA make the fonts indistinguishable. Therefore, the process of creating training CAPTCHA pairs begins with the selection of only one image from the target dataset and the subsequent manual removal of the background and other interference, leaving only the font outline preserved. This step is of great importance, as it is only through the complete removal of non-outline interfering pixels that the font can be successfully searched and highly similar to the target font. The process is realized manually and may take about a few minutes.

The search procedure is illustrated in Fig. 5. Initially, an image from the target CAPTCHA dataset is selected. Subsequently, extraneous pixels are erased, thereby retaining only the contour of the font. Subsequently, the image is uploaded to a font recognition website, where the font used in the image is identified. To create the training dataset, clean images

comprising random characters are generated based on the obtained font, with no background or other interference. Dirty CAPTCHAs that correspond one-to-one with the clean images are also created. The dirty images incorporate a variety of elements, including random lines, noisy points, and different backgrounds, to simulate the target CAPTCHAs. The background colors for the dirty CAPTCHAs may be randomly selected, provided that they encompass the red, green, and blue (RGB) channels, thus generating a wide range of colors that cover the full-color space.

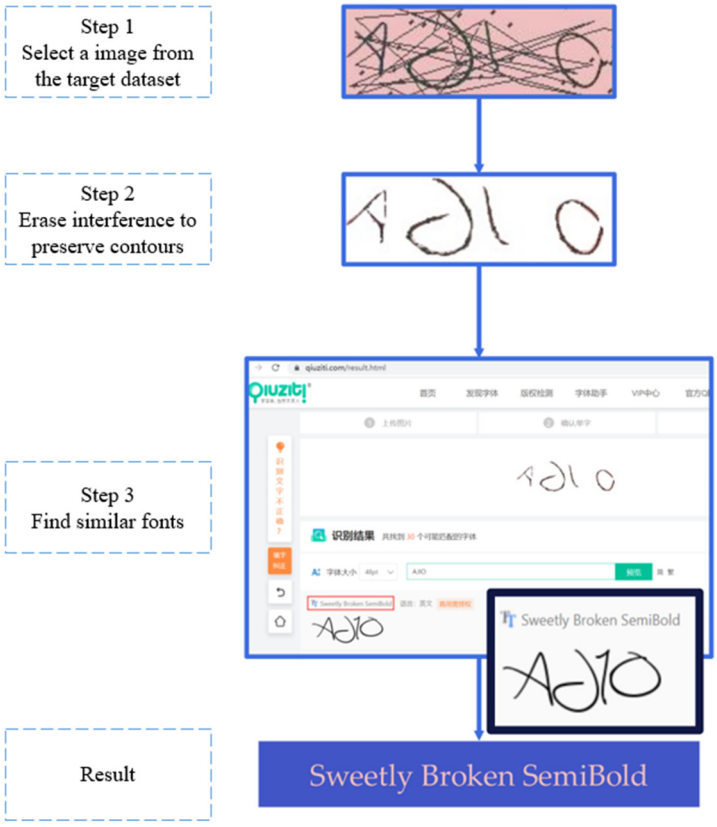

Fig. 5 The procedure of font searching

The significance of constructing such CAPTCHA pairs is that the valid information in the images only comes from their fonts. By comparing clean and dirty images, font information in the CAPTCHAs could be learned by FEGAN, which is useful for the effective removal of interfering pixels from other target CAPTCHAs. The clean, dirty, and target CAPTCHAs are shown in Fig. 6, where the first two appear in pairs and the target CAPTCHAs are unpaired, which are selected from the M-CAPTCHA dataset. From the figure, the font of the clean dataset is close to that of the target dataset. However, their background colors are different.

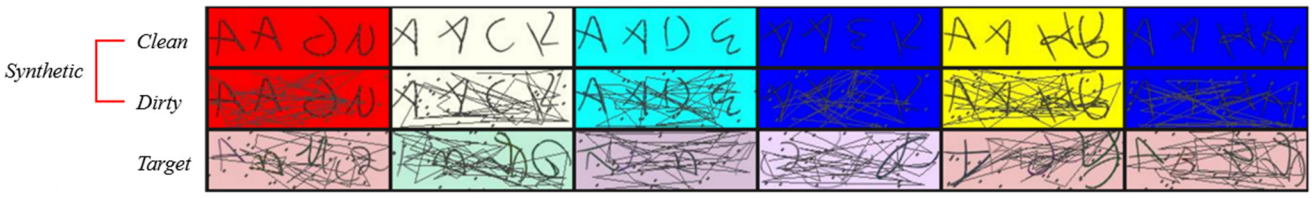

Fig. 6 Synthetic and target CAPTCHAs from M-CAPTCHA dataset

3.2. FEGAN

The second step is to train FEGAN using the synthetic CAPTCHA pairs. The architecture of FEGAN is shown in Fig. 7. In this network, a generator for font enhancement is first introduced, complemented by two discriminators. The design of this architecture is inspired by image denoising with frequency domain (UID-FDK), an unsupervised learning model [22]. UID-FDK removes noise and smooths pictures while preserving as much content and background. However, the objective of

FEGAN is to remove all non-relevant pixels from the font using a supervised learning model. Therefore, one generator was reduced from UID-FDK, and some changes were made to the loss functions while the same structures of the generator and discriminators were kept.

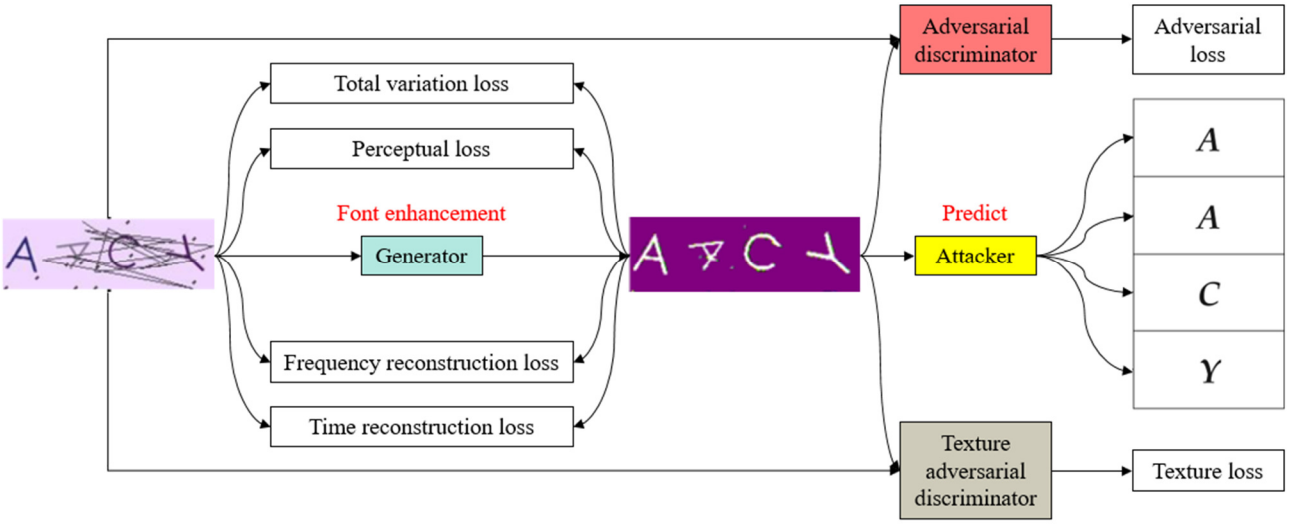

Fig. 7 The architecture of FEGAN

The adversarial discriminator must accurately differentiate between clean and dirty images. To this end, two loss functions improve its discriminative ability. The first one is adversarial loss, which helps the generator's output distribution to better fit the real distribution of the target dataset. The function takes two low-dimensional feature tensors instead of one scalar as input, facilitating the presentation of better details. Assuming that x are from the dirty data domain $X \sim P_d$, and y are from the clean data domain $Y \sim P_c$, the goal of the adversarial discriminator D_{adv} is to distinguish between them to the greatest extent possible. Adversarial loss values can be obtained by calculating the least squares of the two inputs and their labels, which was first proposed by Least Square GAN [23].

$$L_{adv} = E_{y \sim P_c}\left[\left\|D_{adv}(y)\right\|_2\right] + E_{x \sim P_d}\left[\left\|D_{adv}\left(G(x)\right) - \mathrm{I}\right\|_2\right] \tag{1}$$

The other is adversarial texture loss, which focuses on comparing the difference in texture details between clean and dirty images by excluding the effect of the absolute value of the color and only considering the difference between adjacent pixels. Grayscale conversion transforms a color image into its intensity values, removing chromatic information while preserving luminance, resulting in a single-channel image with varying shades of gray. By training pairs of CAPTCHA images, the texture discriminator learns the association information between fonts and non-font pixels to produce a cleaner texture that contains only fonts. The texture loss also follows an adversarial approach.

$$L_{adv}^{Texture} = E_{y \sim P_c}\left[\left\|D_{texture}\left(gretscale(y)\right)\right\|_2\right] + E_{x \sim P_d}\left\{\left\|D_{texture}\left[greyscale\left(G(x)\right)\right] - \mathrm{I}\right\|_2\right\} \tag{2}$$

The generator's objective function for the adversarial loss and texture adversarial loss mirrors that of the discriminator, with the distinction that their optimization goals are contrary. The texture adversarial discriminator differs from the adversarial discriminator in that the former processes a single-channel image, whereas the latter handles a three-channel image.

Additionally, a two-dimensional max-min normalized frequency loss is introduced as follows.

$$L_{reconst}^{Freq} = E_{x,y \sim P_{d,c}}\left\{normalize_{Max-Min}\left[\log_{10}\left\|FFT_{2D}\left(G(x)\right) - FFT_{2D}(y)\right\|_1\right]\right\} \tag{3}$$

The network often finds it challenging to directly remove the interfering lines and dots from the time domain due to their close intertwinement with the CAPTCHA fonts. Interference like Gaussian noise is easier to handle in the frequency domain, so the

de-interference effect is enhanced with the signal spectrum from a macro perspective, ensuring the overall consistency of the generated images. The frequency spectrum is obtained by the two-dimensional Fourier transform, and the normalization guarantees that the range of its values is comparable to other losses.

To highlight the font pixels, the total variation loss is added, which reduces the degree of variation between neighboring regions. This loss is especially vital for font enhancement because the ideal result is that the total variance of the font and background internal regions after font enhancement should be zero. In addition, image gradients are minimized in horizontal and vertical directions to refine the font's appearance further. The corresponding formulation is given below.

$$L_{TV} = E_{x\sim P_d}\left[\left\|\nabla_w G(x)\right\|_2 + \left\|\nabla_h G(x)\right\|_2\right] \tag{4}$$

To obtain a pixel-level reconstruction, the pixel-wise difference between the dirty and clean images is evaluated. This reconstruction loss, while helping the model learn font information, introduces unnecessary background learning. It is important to choose an appropriate weight to balance the background and fonts.

$$L_{reconst}^{Time} = E_{x,y\sim P_{d,c}}\left[\left\|G(x)-y\right\|_2\right] \tag{5}$$

Additionally, the generator introduces a perceptual loss by computing certain output layers of the Visual Geometry Group (VGG). This loss, in contrast to pixel loss, makes the output images perceptually more intuitively similar to the target image, rather than forcing them to match pixels exactly [24]. The higher layers of VGG minimize their output to maintain the semantic information. The corresponding formulation is given below.

$$L_{percetual} = E_{x,y\sim P_{d,c}}\left[\left\|VGG\left(G(x)\right)-VGG(y)\right\|_2\right] \tag{6}$$

Ultimately, the optimization objective comprises six losses, with corresponding weights assigned to each part of the loss. The total loss function is formulated as follows, and the discriminator and generator need to be trained alternately to obtain the optimal model parameters.

$$\min_{G}\max_{D_{adv},D_{texture}} \lambda_{adv}L_{adv} + \lambda_{texture}L_{adv}^{Texture} + \lambda_{freq}L_{reconst}^{Freq} + \lambda_{tv}L_{TV} + \lambda_{time}L_{reconst}^{Time} + \lambda_{percetual}L_{percetual} \tag{7}$$

3.3. Recognition procedure using FEGAN

The trained FEGAN should be incorporated into recognizers as a preprocessing network for their training and testing, as depicted in Fig. 8. The recognition models can be AdaptiveCAPTCHA, DeepCAPTCHA, ConvNet, and others that were introduced earlier. Before recognition, the CAPTCHA dataset needs to undergo preprocessing by FEGAN to form the FE dataset. Ultimately, the recognition model recognizes the images in the FE dataset.

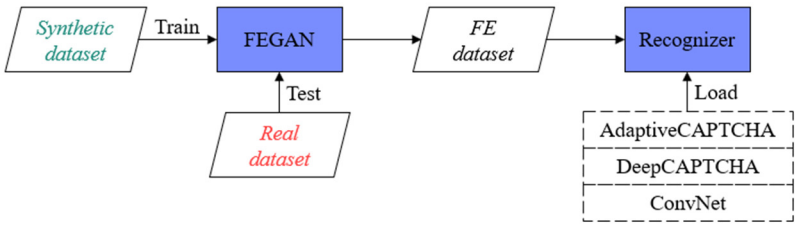

Fig. 8 Training and testing flow of FEGAN

4. Results and Discussion

In this study, two datasets are employed for evaluation. The first dataset is the publicly accessible M-CAPTCHA dataset available on Kaggle (https://www.kaggle.com/datasets/sanluo/mcaptcha). The second dataset, P-CAPTCHA, is generated using the Python ImageCaptcha package with its default settings. From the M-CAPTCHA dataset, 8,000 samples were

randomly selected, and an additional 3,000 images were randomly generated for the P-CAPTCHA dataset. Every CAPTCHA image consists of four characters, with each character being selected from the set of 26 uppercase English letters. 10,000 pairs of clean and dirty CAPTCHAs are generated for FEGAN training. The performance of FEGAN was evaluated by combining five different recognition models. In addition, FEGAN is compared with another preprocessing method, Multiview-filtering. The average attacking success rate (AASR) is employed as a metric to evaluate the effectiveness of the approach, as the goal of FEGAN is to enhance the recognition accuracy of attackers. The training experiments† were all run on the NVIDIA A100 platform, while the inference tests were performed on the Intel(R) Core(TM) i5-8265U, as shown in Table 4.

Table 4 Experimental parameter settings

Parameter	Value
Epoch	300
Learning	0.001
Optimizer	Adam
Batch size	64
GPU	NVIDIA A100
CPU	Intel (R) Core (TM) i5-8265U
Dataset	M-CAPTCHA (8,000 images), P-CAPTCHA (3,000 images)
Training pairs	10,000
Recognizer	DeepCAPTCHA, AdaptiveCAPTCHA, CapsNet, ConvNet, VeriBypasser
Preprocess method	FEGAN, Multiview-filtering

4.1. Visual results of FEGAN on different datasets

Fig. 9 depicts the visual outcomes of font enhancement using FEGAN on the M-CAPTCHA dataset. Discerning CAPTCHAs is difficult since they contain a large number of linear and non-linear points and lines. After undergoing font enhancement with FEGAN, the font's outline becomes more prominent, resulting in a significant reduction in background pattern and interference. All the backgrounds assume the same color, indicating that FEGAN recognizes that the fonts are unaffected by the backgrounds. Fig. 10 shows that when the synthetic dataset and the target dataset have similar fonts and interference styles, FEGAN still achieves comparable results. Fig. 11 shows the results of FEGAN when testing CAPTCHAs with the same font but without any background, while eliminating interference.

Fig. 9 Training and testing on both the font of M-CAPTCHA

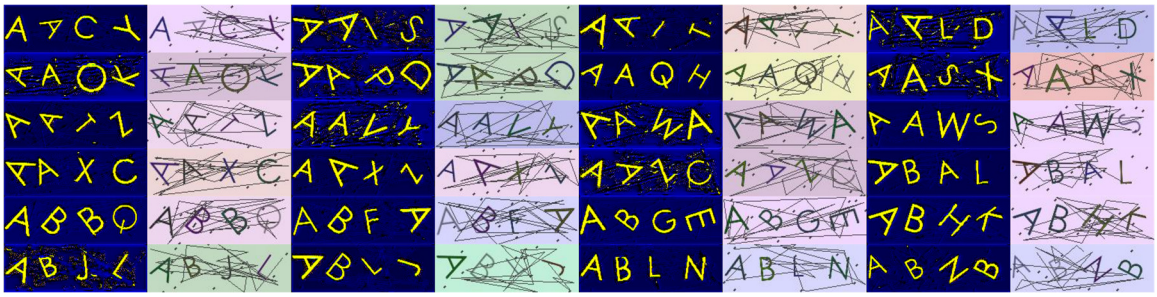

Fig. 10 Training and testing on the similar fonts

† FEGAN code: https://github.com/krantson/FEGAN; Recognizer code: https://github.com/krantson/Recognizer

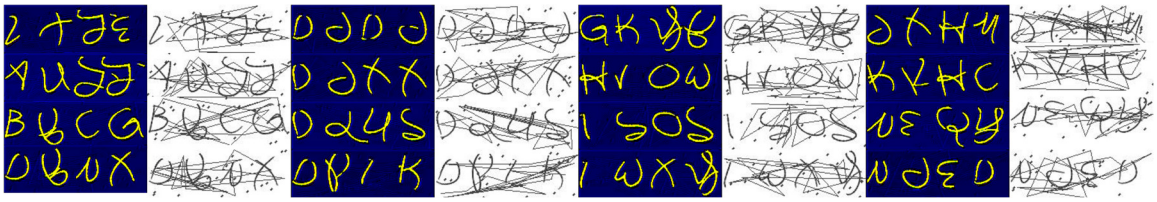

Fig. 11 Testing on the same font but no background

The findings on P-CAPTCHA, as depicted in Fig. 12, are unsatisfactory due to significant disparities in typefaces and styles between the testing dataset P-CAPTCHA and the fake M-CAPTCHA dataset. Consequently, Comparable fonts and produced synthetic pairs were searched based on the style that closely resembled that of P-CAPTCHA, as depicted in Fig. 13. It demonstrates that, although comparable fonts are used, there are noticeable distinctions. On the other hand, the synthetic dataset utilizes a randomly generated background color.

Furthermore, the fake dataset excludes random interference lines because P-CAPTCHA has a limited supply. Fig. 14 demonstrates the impact of FE through training with 10,000 synthetic CAPTCHAs. The results indicate a substantial reduction in interference and a significant suppression of background colors. The preservation of the font outlines in their original form is evident, and the image displays a heightened contrast.

Fig. 12 Testing on P-CAPTCHA after training on synthetic M-CAPTCHA

Fig. 13 Generating the training pairs with a similar font of P-CAPTCHA

Fig. 14 Testing on the P-CAPTCHA after training on synthetic pairs

Fig. 15 illustrates FEGAN's efficacy when integrated with DeepCAPTCHA for recognition. The model without FEGAN integration achieves a final training accuracy of approximately 83%, while the model with FEGAN integration achieves an accuracy of approximately 99%, indicating a 16% improvement over the original model. Fig. 16 shows that the test accuracy with FEGAN is about 35% higher than the accuracy rate without it. Both experiments show that with FEGAN font enhancement, DeepCAPTCHA can extract key features more efficiently to improve accuracy.

The reason that the improvement in testing accuracy is larger than the training accuracy is that the dataset becomes simpler after FEGAN preprocessing, which prevents overfitting of the model and thus improves the generalization performance. Fig. 17 demonstrates that by employing AdaptiveCAPTCHA with FEGAN on M-CAPTCHA for font enhancement, the accuracy significantly increases from approximately 50% to almost 99%, resulting in a nearly twofold improvement in performance.

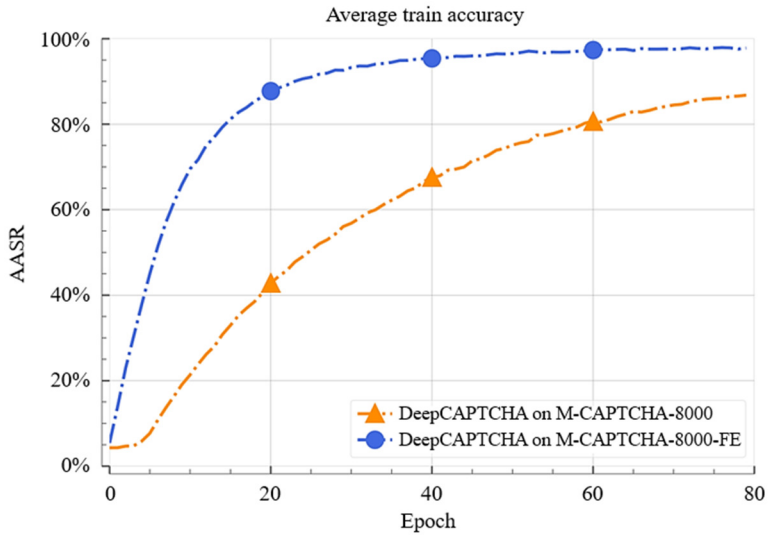

Fig. 15 Training accuracy of DeepCAPTCHA on the M-CAPTCHA with FEGAN

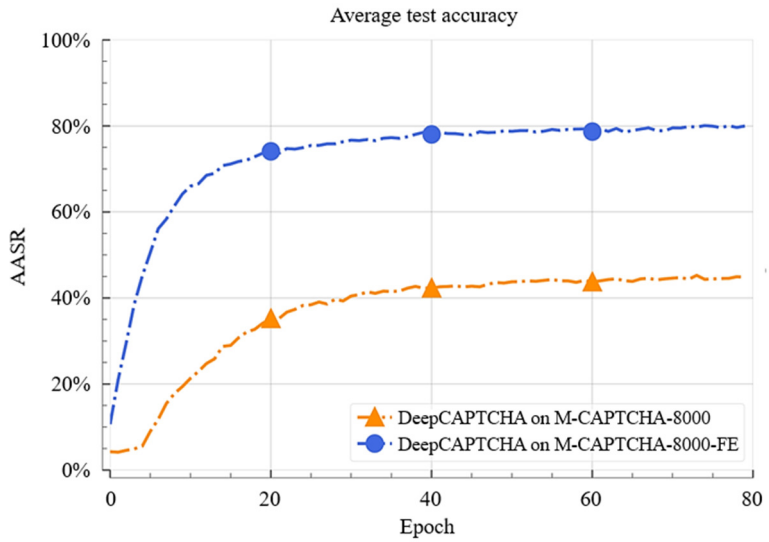

Fig. 16 Testing accuracy on the M-CAPTCHA with DeepCAPTCHA

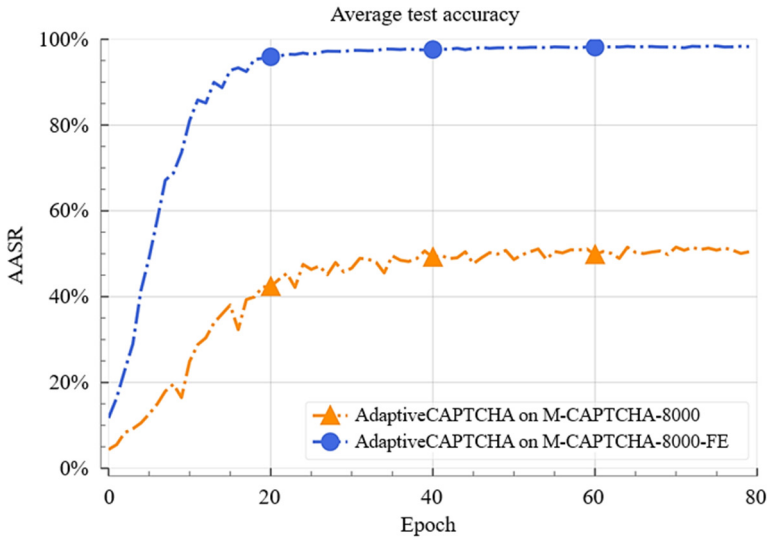

Fig. 17 Testing accuracy on the M-CAPTCHA with AdaptiveCAPTCHA

In addition, the FEGAN improves the ability of other attack models to recognize characters more effectively on M-CAPTCHA, as seen in Fig. 18. The recognition accuracies of DeepCAPTCHA, ConvNet, CapsNet, and VeriBypasser models are improved greatly after undergoing FEGAN processing, resulting in an increase of around 36%, 70%, 51%, and 40%,

respectively. Fig. 19 shows the accuracies of different recognizers on the P-CAPTCHA. All models gained at least 7 percentage points or more. Due to the limited number of anti-attack measures used by P-CAPTCHA, the benefits are less significant compared to M-CAPTCHA.

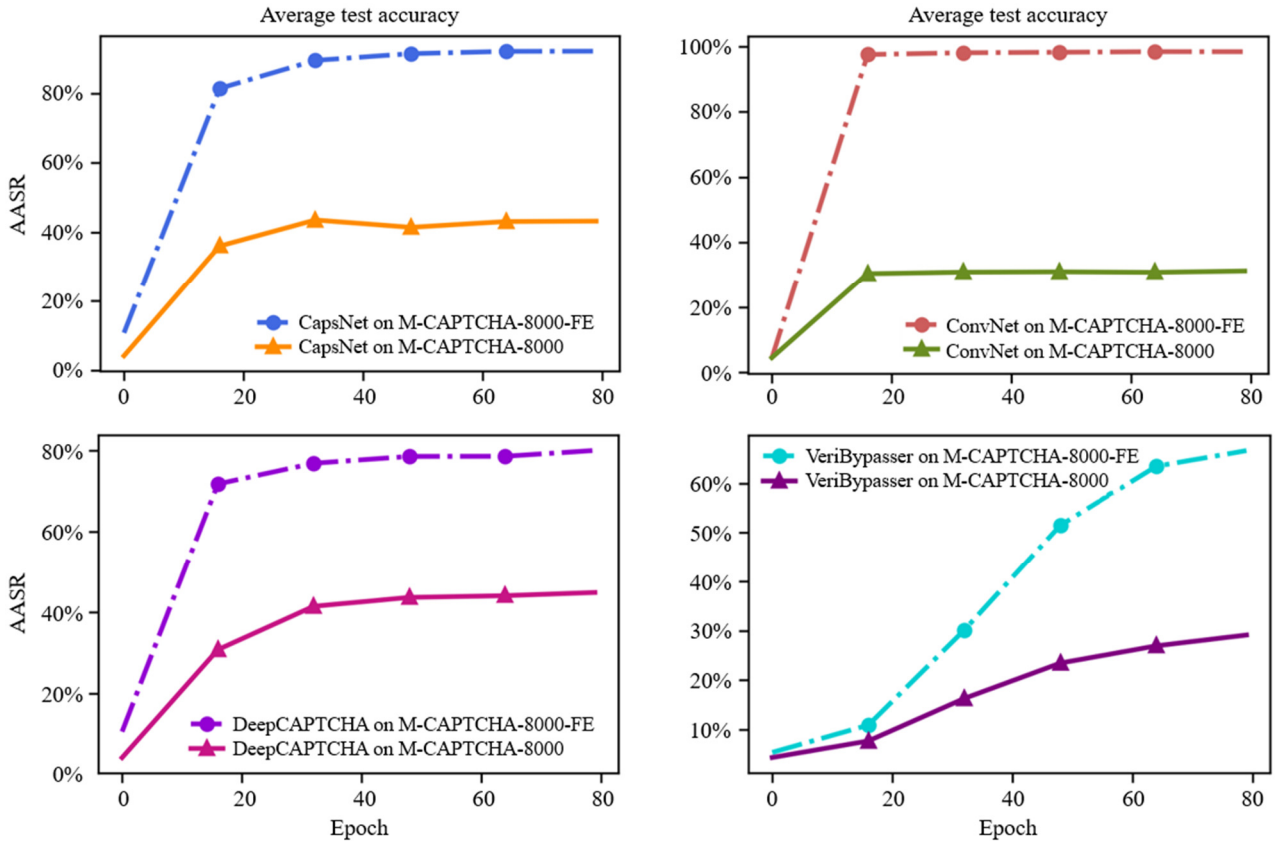

Fig. 18 Testing accuracy for different models using FEGAN testing on M-CAPTCHA

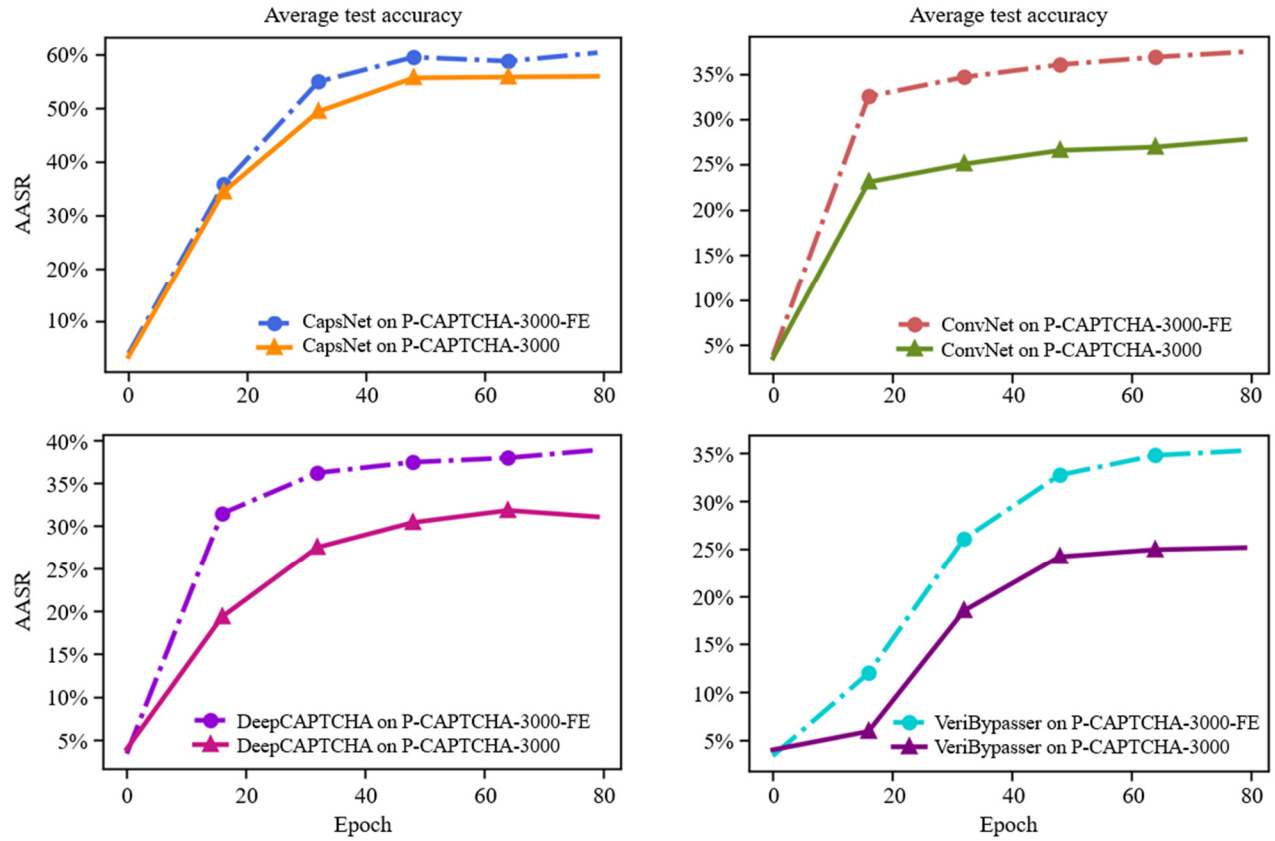

Fig. 19 Testing accuracy for different models using FEGAN testing on P-CAPTCHA

Fig. 20 shows the results of the comparison between the FEGAN-based method and the Multiview-filtering-based method on P-CAPTCHA, demonstrating that the FEGAN method has the highest recognition accuracy. For Multiview-filtering, two or three combinations of Gaussian filtering, median filtering, and bilateral filtering were utilized to preprocess the input images and predict them using DeepCAPTCHA. For M-CAPTCHA, the effect of FEGAN is more striking, as shown in Fig. 21. Since M-CAPTCHA has more jamming information, the DeepCAPTCHA can gain more from the font enhancement, about 50 percentage points higher than the best filter-based methods. It is evident that FEGAN, when applied to both M-CAPTCHA and P-CAPTCHA, significantly enhances the recognizer's accuracy compared to the Multiview-filtering method.

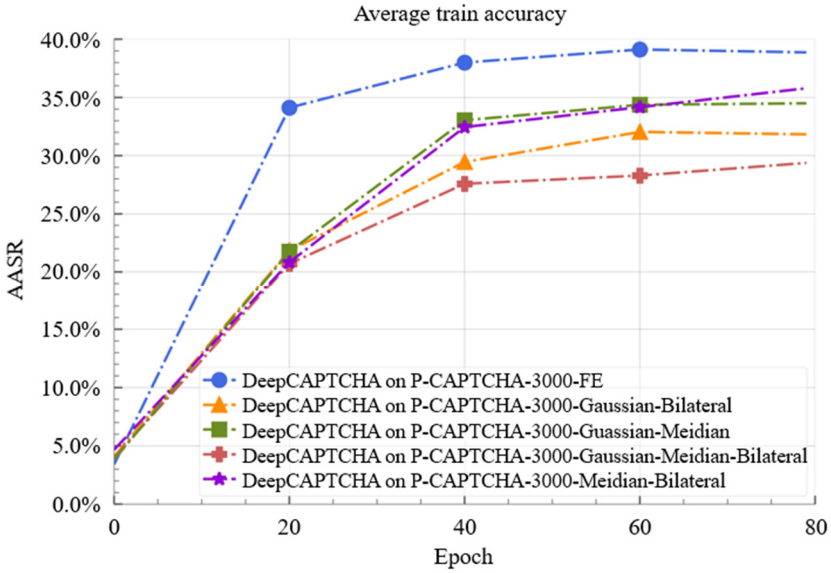

Fig. 20 Accuracy comparison of DeepCAPTCHA on P-CAPTCHA using Multiview-filtering

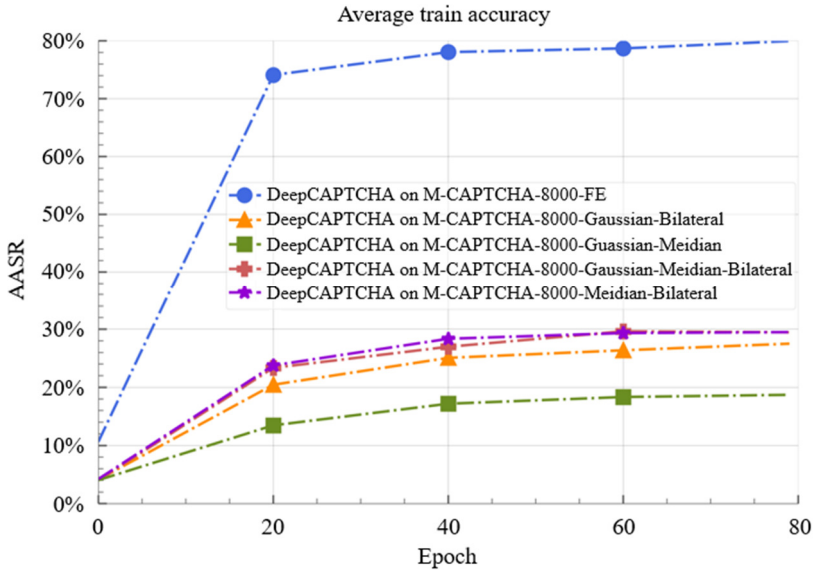

Fig. 21 Accuracy comparison of DeepCAPTCHA on M-CAPTCHA using Multiview-filtering

Grad-CAM is an interpretability technique that visualizes model decisions by highlighting important regions through gradients [25]. Fig. 22 shows the gradient heat map of the CAPTCHA from the M-CAPTCHA dataset before and after FE processing, where the output features of the 5th activation function of the AdaptiveCAPTCHA model are used as the class activation map. The colored highlights show the regions that have more influence on the gradient of the middle two characters. The characters are more affected by the bottom region and the neighboring characters after FE processing. When the gradient highlights of the original CAPTCHA are dispersed and impacted by interference, the application of FE demonstrates an ability to make the gradient more continuous and stable, effectively reducing the interference.

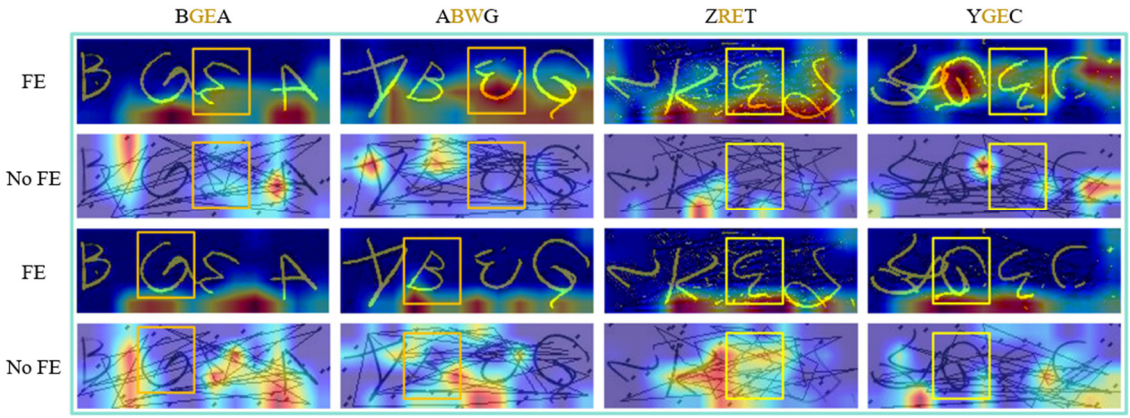

Fig. 22 Pre-FE and post-FE Grad-CAM analysis on M-CAPTCHA-8000

Fig. 23 illustrates the alteration in the image when the weights of the weights remain constant while one term is substantially increased. Increasing total variation loss smooths the image and diminishes interference, yet it also blurs the typeface; conversely, assigning greater weight to texture loss accentuates the font but introduces additional shadows. As previously mentioned, augmenting the time-domain loss reinstates the font but introduces background interference, whereas emphasizing the frequency-domain loss mitigates Gaussian noise and is ineffective against line jamming. While augmenting the adversarial loss does not enhance the image to the same extent as other losses, it remains an integral component of the GAN and contributes to the network's generalization capability.

Ultimately, the perceptual loss preserves the higher-level features of the image to the greatest extent possible, thereby indicating some interference. These losses interact with each other and need optimal weights in network training based on different datasets.

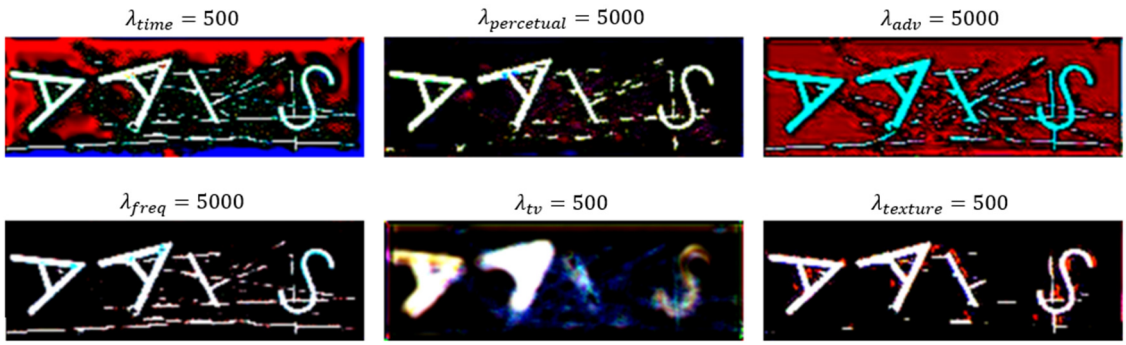

Fig. 23 The effect on the image when increasing only one loss weight

The time required for the training phase is related to the size of the dataset and the hardware platform and can be done in a single training session. For a dataset of about 10,000 images, FEGAN needs to train about 300 epochs to converge on the A100 platform. Typically, inference time needs to be evaluated by neural network models, and here the time required was tested with different models on the two pipelines, preprocessing and recognition, on the Intel (R) Core (TM) i5-8265U platform.

Table 5 shows that in the preprocessing stage, FE takes around 240 ms, much more than the time required for filtering and higher than the recognition time of all recognizers. In addition, ConvNet has the highest recognition rate of 98.4%, but it takes about 66 ms, whereas other models usually have recognition times of around 10 ms. Notably, the PSNR after FE was only 4.57 and the SSIM was only 0.094, suggesting that these metrics are not effective for evaluating font enhancement.

It should be noted that the font enhancement effect of FEGAN depends on the fake dataset, which is constructed by mimicking the target CAPTCHA font and the main form of interference. The entire recognition process proposed in this paper is time-consuming, including three phases: generation of a pseudo-CAPTCHA dataset, FEGAN training, and training and testing of FEGAN with recognizers. Moreover, the overall loss function is intricate, and achieving a balance among the weights of different loss components poses a considerable challenge.

Future research should be able to further integrate FEGAN and CAPTCHA recognizers into one model to further reduce model complexity and training time. This can be achieved by adding the binary cross-entropy loss of the recognizer to the objective function of FEGAN. This new loss function can give font enhancement a clearer training goal, which makes the CAPTCHA better recognized by models. Future studies should focus on balancing the effects of different losses that interact with each other.

Table 5 Inference time and performance of recognizers and Pre-processing on M-CAPTCHA8000-FE

Model	Pipeline	Inference time per image (ms)	Performance
AdaptiveCAPTCHA	Recognition	14.5	98.3% (AASR)
DeepCAPTCHA	Recognition	9.2	80.0% (AASR)
ConvNet	Recognition	66.5	98.4% (AASR)
CapsNet	Recognition	9.3	92.2% (AASR)
VeriBypasser	Recognition	13.6	72.0% (AASR)
Gaussian-Bilateral filtering	Pre-processing	5.4	0.44 (PSNR), 0.12 (SSIM)
Guassian-Meidian filtering	Pre-processing	1.1	0.44 (PSNR), 0.11 (SSIM)
Font enhancement	Pre-processing	240.3	4.57 (PSNR), 0.094 (SSIM)

5. Conclusion

This study introduces the concept of CAPTCHA font enhancement and develops a font enhancement network called FEGAN. This network is designed to eliminate anti-attack measures in text CAPTCHA by leveraging the similarity of target typefaces and interference. By eliminating extraneous pixels in the target CAPTCHA, fonts that closely resemble the target CAPTCHA can be identified, which allows for the creation of a synthetic dataset for FEGAN training.

The experimental results demonstrate the following conclusions:

(1) The proposed FEGAN is highly effective in eliminating anti-attack mechanisms in text CAPTCHAs while preserving the original typeface.
(2) The combination of FEGAN and Multiview-filtering techniques, along with recognizers such as DeepCAPTCHA, AdaptiveCAPTCHA, ConvNet, and Capsule, enables accurate evaluation of M-CAPTCHAs and P-CAPTCHAs.
(3) Compared to Multiview-filtering alone, integrating FEGAN with all recognizers further improves recognition accuracy, providing valuable insights for font optimization and text CAPTCHA security research.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] S. Sharma and D. Singh, “CAPTCHA in Web Security and Deep-Captcha Configuration Based on Machine Learning,” 3rd International Conference for Innovation in Technology, pp. 1-6, 2024.
[2] P. Wang, H. Gao, X. Guo, C. Xiao, F. Qi, and Z. Yan, “An Experimental Investigation of Text-Based CAPTCHA Attacks and Their Robustness,” vol. 55, no. 9, ACM Computing Surveys, article no. 196, 2023.
[3] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond Empirical Risk Minimization,” https://doi.org/10.48550/arXiv.1710.09412, 2018.
[4] M. Faramarzi, M. Amini, A. Badrinaaraayanan, V. Verma, and S. Chandar, “PatchUp: A Feature-Space Block-Level Regularization Technique for Convolutional Neural Networks,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 1, pp. 589-597, 2022.
[5] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random Erasing Data Augmentation,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 07, pp. 13001-13008, 2020.

[6] W. Xing, M. R. S. Mohd, J. Johari, and F. A. Ruslan, "A Review on Text-Based CAPTCHA Breaking Based on Deep Learning Methods," International Conference on Computer Engineering and Distance Learning, pp. 171-175, 2023.
[7] Z. Noury and M. Rezaei, "Deep-CAPTCHA: A Deep Learning Based CAPTCHA Solver for Vulnerability Assessment," https://doi.org/10.48550/arXiv.2006.08296, 2020.
[8] X. Wan, J. Johari, and F. A. Ruslan, "Adaptive CAPTCHA: A CRNN-Based Text CAPTCHA Solver with Adaptive Fusion Filter Networks," Applied Sciences, vol. 14, no. 12, article no. 5016, 2024.
[9] K. Qing and R. Zhang, "An Efficient ConvNet for Text-Based CAPTCHA Recognition," International Symposium on Intelligent Signal Processing and Communication Systems, pp. 1-4, 2022.
[10] I. G. Mocanu, Z. Yang, and V. Belle, "Breaking CAPTCHA with Capsule Networks," Neural Networks, vol. 154, pp. 246-254, 2022.
[11] W. Ding, Y. Luo, Y. Lin, Y. Yang, and S. Lian, "VeriBypasser: An Automatic Image Verification Code Recognition System Based on CNN," Computer Communications, vol. 217, pp. 246-258, 2024.
[12] F. Liu, Z. Li, X. Li, and T. Lv, "A Text-Based CAPTCHA Cracking System with Generative Adversarial Networks," IEEE International Symposium on Multimedia, pp. 192-193, 2018.
[13] A. Thobhani, M. Gao, A. Hawbani, S. T. M. Ali, and A. Abdussalam, "CAPTCHA Recognition Using Deep Learning with Attached Binary Images," Electronics, vol. 9, no. 9, article no. 1522, 2020.
[14] C. Li, X. Chen, H. Wang, P. Wang, Y. Zhang, and W. Wang, "End-to-End Attack on Text-Based CAPTCHAs Based on Cycle-Consistent Generative Adversarial Network," Neurocomputing, vol. 433, pp. 223-236, 2021.
[15] Y. Wang, Y. Wei, M. Zhang, Y. Liu, and B. Wang, "Make Complex CAPTCHAs Simple: A Fast Text Captcha Solver Based on a Small Number of Samples," Information Sciences, vol. 578, pp. 181-194, 2021.
[16] T. Kimbrough, P. Tian, W. Liao, E. Blasch, and W. Yu, "Deep CAPTCHA Recognition Using Encapsulated Preprocessing and Heterogeneous Datasets," IEEE Conference on Computer Communications Workshops, pp. 1-6, 2022.
[17] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, et al., "Using Generative Adversarial Networks to Break and Protect Text Captchas," ACM Transactions on Privacy and Security, vol. 23, no. 2, article no. 7, 2020.
[18] N. Zhang, M. Ebrahimi, W. Li, and H. Chen, "Counteracting Dark Web Text-Based CAPTCHA with Generative Adversarial Learning for Proactive Cyber Threat Intelligence," ACM Transactions on Management Information Systems, vol. 13, no. 2, article no. 21, 2022.
[19] M. O. Yusuf, D. Srivastava, D. Singh, and V. S. Rathor, "Multiview Deep Learning-Based Attack to Break Text-CAPTCHAs," International Journal of Machine Learning and Cybernetics, vol. 14, no. 3, pp. 959-972, 2023.
[20] D. O. Ishkov and V. I. Terekhov, "Text CAPTCHA Traversal with ConvNets: Impact of Color Channels," 4th International Youth Conference on Radio Electronics, Electrical and Power Engineering, pp. 1-5. 2022.
[21] A. Koshy, N. B. MJ, S. A, and A. John, "Preprocessing Techniques for High Quality Text Extraction from Text Images," 1st International Conference on Innovations in Information and Communication Technology, pp. 1-4, 2019.
[22] N. Kim, D. Jang, S. Lee, B. Kim, and D. S. Kim, "Unsupervised Image Denoising with Frequency Domain Knowledge," https://doi.org/10.48550/arXiv.2111.14362, 2021.
[23] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least Squares Generative Adversarial Networks," Proceedings of the IEEE International Conference on Computer Vision, pp. 2794-2802, 2017.
[24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution." Computer Vision – ECCV 2016: 14th European Conference on Computer Vision, pp. 694-711, 2016.
[25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," International Journal of Computer Vision, vol. 128, no. 2, pp. 336-359, 2020.