Article

Not peer-reviewed version

Adaptive-CAPTCHA: A Text CAPTCHA Solver based on CRNN and Configurable Filter Networks

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

Posted Date: 16 April 2024

doi: 10.20944/preprints202404.1050.v1

Keywords: CAPTCHA recognition; noise, interference, filter, LSTM, Resistance Mechanisms

Article

Adaptive-CAPTCHA: A Text CAPTCHA Solver based on CRNN and Configurable Filter Networks

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

[1] School of Electrical Engineering, Universiti Teknologi MARA (UiTM) Shah Alam 40450, Malaysia
[2] School of Intelligent Manufacturing, Leshan Vocational &Technical College, Leshan 614000, China
* Correspondence: 2022995467@student.utim.edu.my (X.W.); fazlina419@uitm.edu.my (F.A.R.)

Abstract: Text-based CAPTCHA remains the most widely adopted security scheme, which is the first barrier to securing websites. Deep learning methods, especially Convolutional Neural Networks (CNNs) are the mainstream approach for text-CAPTCHA recognition, which are widely used in CAPTCHA vulnerability assessment and data collection. However, verification code recognizers are mostly deployed on the CPU platform as part of a web crawler and security assessment, they are required to have both low complexity and high recognition accuracy. Due to the specifically designed anti-attack mechanisms like noise, interference, geometric deformation, twisting, rotation, and character adhesion in text CAPTCHAs, some characters are difficult to efficiently identify with high accuracy in these complex CAPTCHA images. This paper proposed a recognition model named Adaptive-CAPTCHA with a CRNN module and trainable and configurable filtering networks, which effectively handle the interference and learn the correlation between characters in CAPTCHAs to enhance recognition accuracy. Experimental results on two datasets of different complexity show that compared with the baseline model Deep-CAPTCHA, the number of parameters of our proposed model is reduced by about 70% and the recognition accuracy is improved by more than 10 percentage points in the two datasets. In addition, the proposed model has a faster training convergence speed.

Keywords: CAPTCHA recognition; noise; interference; filter; LSTM; Resistance Mechanisms

1. Introduction

The Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA), which is recognized as Human Interactive Proofs (HIP), represents a widely utilized mechanism designed to autonomously differentiate between human users and machines [1]. Introduced by von Ahn and colleagues as a text-oriented CAPTCHA variant known as reCAPTCHA, this CAPTCHA emerged from a research initiative at Carnegie Mellon University in 2003 [2]. Due to its low cost, high reliability, and easy deployment, text-based CAPTCHA is widely used as a security product on the Internet, and is applied to many websites and information systems around the world [3]. However, it is difficult for websites to distinguish whether the CAPTCHA reader is a human or a machine. These text-CAPTCHAs are increasingly vulnerable to malicious attacks based on deep learning, such as popular Optical Character Recognition (OCR) [4]. The pervasive incorporation of digital and Roman characters in text-based CAPTCHAs could render them susceptible to interpretation by OCR [5]. For CAPTCHAs engendered with elevated levels of noise, the efficacy of models in deciphering the content is significantly impeded owing to the persistence of interference even after the images are binarized [6]. Breaking algorithms based on traditional machine learning, including Decision Trees (DT), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM), also suffer from insufficient recognition ability [7].

For text-based CAPTCHA designers, it is necessary to balance both the resistance mechanisms against web attacks and the usability of website users [8]. To increase anti-attack ability without reducing the user's recognition time, Connecting Characters Together (CCT) and deformation are used in the CAPTCHA design [9]. As a result, a few consecutive letters may be difficult to distinguish by the naked eye, such as two back-to-back letters V, as shown in Figure 1. To avoid confusion, designers try to reduce confusing characters when designing CAPTCHAs, resulting in Markov Transition Probabilities (MTP) between adjacent characters. For this reason, some datasets present the character sequence dependence between two consecutive characters, such as the public dataset M-CAPTCHA [10]. For these CAPTCHAs, break models must consider how to process this correlation between adjacent characters.

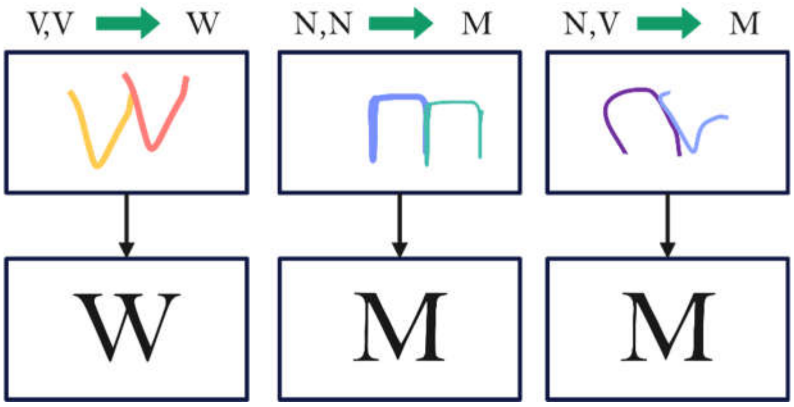


Figure 1. Some confusing adjacent characters in CAPTCHAs.

In recent years, the field of text-based CAPTCHA recognition has seen several advancements, particularly in the realm of deep learning, which has become the mainstream direction for text-based CAPTCHA recognition [11]. CNNs are the most used networks as they excel at learning spatial hierarchies in an image, thereby providing a robust way for CAPTCHA recognition [12]. Recurrent Neural Network (RNN) or Long Short-Term Memory (LSTM) is another neural network, which has shown impressive recognition accuracy in CAPTCHAs with CCT as it can remember patterns over lengthy sequences. [13]. Combined with CNN and RNN, the attention mechanisms allow the neural network to focus on certain important features, spatial positions, and channels, thereby improving the recognition effect [14]. Additionally, some content generation techniques have shown potential in enhancing models by creating believable CAPTCHA-like images to train a more robust network [15]. It is important to note that a single model may not always be the best choice. Depending on the complexity of the CAPTCHAs, a hybrid approach could be employed to decode CAPTCHAs [16].

In addition to designing robust network models, evaluation of model performance is critical. Among all evaluation metrics, the Attack Success Rate (ASR) and the number of Parameters (PARAMs) are the key metrics for CAPTCHA recognition. The definition of ASR is the proportion of the number of correctly recognized characters to the total number of characters in a dataset, shown in the following formula 1. Average Attack Success Rate (AASR) is another index, which is used to measure the overall recognition accuracy of CAPTCHAs regardless of the specific character position.

$$\text{ASR} = \text{the number of characters recognized/the number of all characters} \tag{1}$$

The CAPTCHA recognition model is commonly utilized as a module of web scrawlers running on personal CPU platforms in data collection tasks. Therefore, it is necessary to reduce PARAMs while maintaining AASR. Accordingly, it is important to discern as many CAPTCHAs as possible per second. Frames Per Second (FPS) is therefore employed as a metric to gauge the real-time performance of CAPTCHA solvers, which reflects the number of CAPTCHAs that a solver can process each second, making it a critical measure of efficiency in some scenarios. Multiply-Accumulate Operations Per Second (MACs) and Floating-Point Operations Per Second (FLOPs) are another two metrics, which are employed to evaluate computational performance and estimate the complexity of models or algorithms.

To train the model, an appropriate loss function is selected to better update the model parameters. For classification models, Cross-Entropy (CE) and Binary Cross-Entropy (BCE) are the most used. Focal Loss, an improved version of the BCE loss function, is often used in object detection and image classification tasks [15]. It is worth mentioning that performance evaluation is generally tested on public datasets.

The primary contribution of the research is to propose a new text-based CAPTCHA recognition model, named Adaptive-CAPTCHA based on Deep-CAPTCHA, which has the characteristics of small model parameters and high recognition accuracy. The improvements are in three aspects:

- Trainable and configurable multi-layer filtering networks are added to combat the noise and interference presented in CAPTCHAs.
- A CNN combined with RNN (CRNN) component is adopted to replace the global Fully Connected (FC) layers to increase the ability to identify correlation between characters, which also greatly reduces the number of parameters.
- By introducing residual connections, the model has a faster training convergence speed compared with the baseline.

2. Related Works

In the past, CAPTCHA-breaking methods are mostly using machine learning, which has limited performance and poor algorithm robustness. As image recognition enters the era of deep learning, the main research directions have focused on CNN, RNN, Generative Adversarial Networks (GAN), and object detection networks. These models both require data preprocessing to better recognize characters. Preprocessing acts as a preliminary stage, incorporating tasks such as image grayscale and binarization, thinning, and filtering [17]. Grayscale transformation is a process that converts a colored image into a single-channel image, while binarization is the conversion of a grayscale image to an image composed solely of black and white pixels. In some tasks, thinning is employed to represent the shape of a character as a skeleton, removing specific points from the original image while retaining the overall structure. For CAPTCHA-breaking models that generally run on CPU platforms, the number of parameters cannot be too large, so the grayscale is necessary to convert a three-channel color image into a single-channel image. This can effectively reduce the width of the model without significantly affecting the recognition accuracy. In addition, some cracking methods will first segment the characters before recognizing them, but these segmentation methods require complex designs for different deformations and are not robust[18].

There is usually a filtering module after preprocessing, which is extremely prominent for CAPTCHA recognition with noisy backgrounds. The result of filtering directly affects the character recognition accuracy of the subsequent network. After filtering, predicted results are outputted through neural networks, which consist of a combination of different technologies such as CNN, RNN, and GAN.

2.1. CAPTCHA Recognition with CNN and RNN

CNN and RNN are the most powerful classification networks, and many CAPTCHA recognition networks are constructed based on them. Leveraging the Dense Convolutional Network (DenseNet) with cross-layer connections, Wang et al. introduced modified networks for CAPTCHA recognition [19]. They reduce the number of convolutional blocks and develop specific classifiers for different CAPTCHA image types. A CAPTCHA recognition method with a focal loss function is presented by Wang et al. Their method enhances the traditional VGG network structure and incorporates the focal loss function [20]. Lu et al. proposed a skip-connection CNN model using two publicly available datasets of text-based CAPTCHA images, which yields a promising result compared to previous studies [21]. More recently, capsule networks have been used due to their capability of preserving detailed information about the input [22]. Shi et al. proposed an RCNN network based on the Connectionist Temporal Classification (CTC) loss function and RNN, which improved the detection ability of variable-length characters [23].

Deep-CAPTCHA is a state-of-the-art model that is used in the field of text-based CAPTCHA recognition proposed by Zahra Noury et al. in 2020 [4]. The model consists of a series of convolutional layers and pooling layers, which are responsible for extracting features from the input, as shown in Figure 2. These features are then fed into FC layers for the classification task. The probabilities of each character in the CAPTCHA are finally outputted by several SoftMax functions individually. However, this model does not adequately suppress noise and interference by filtering, considering the large amount of noise spots and interference lines that are specially designed for CAPTCHAs. In addition, the model lacks a module for modeling sequence relationships, whereas there are correlations between many neighboring characters due to interference and statistical distributions. In response to the above two problems, a new text-based CAPTCHA cracker, named Adaptive-CAPTCHA based on the Deep-CAPTCHA model is proposed, which will be explained in detail in the next section.

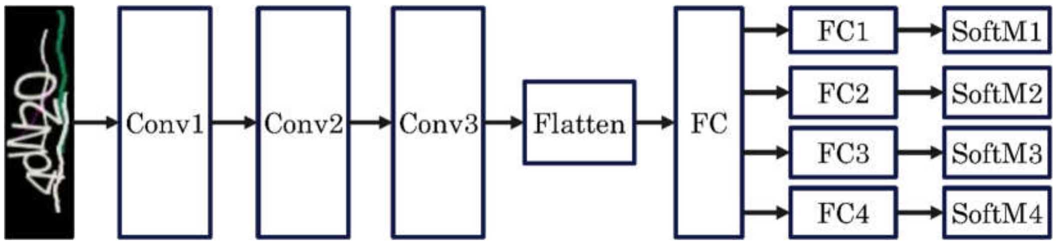


Figure 2. The Network of Deep-CAPTCHA.

2.2. CAPTCHA Recognition with Object Detection Networks

The objective of an object detection algorithm is developed to recognize and spatially locate objects in images [24]. Because each character in the text CAPTCHA can be detected and classified as an object box, object detection networks can identify the content of CAPTCHAs. There are many excellent models in the field of object detection, such as SSD, YOLO3, Faster R-CNN, and YOLOv7 [25–28]. Du et al. demonstrate that Faster R-CNN can effectively extracts feature maps, enabling accurate recognition of the characters and their locations in CAPTCHA images [29]. Experimental results indicate that Faster R-CNN achieves high accuracy in the recognition of CAPTCHAs. Nian et al. proposed a network based on Mask R-CNN, which comprises a feature extraction module, a character location and recognition module, and a coordinate matching module [30]. In 2020, N. Carion et al. adopted end-to-end transformer-based detectors, which have great application potential in the field of CAPTCHA Breaking [31]. However, text-based CAPTCHA recognition based on object detection requires a large amount of image annotation work, resulting in high cost and low efficiency.

2.3. CAPTCHA Recognition with GAN-based Synthetic CAPTCHAs

As transfer learning based on synthetic CAPTCHAs can greatly improve the training effectiveness of the model, GAN methods have become a research hotspot in CAPTCHA cracking in recent years [32]. These methods have shown considerable promise in CAPTCHA recognition, where models pre-trained on specific synthetic CAPTCHA datasets are fine-tuned on the target dataset. In 2018, Ye et al. proposed a model based on GANs to crack text-based CAPTCHAs [14]. In 2020, the authors used GAN-based cracking methods to evaluate 33 different verification code datasets and achieved high accuracy [33]. In 2021, Li et al. introduced a CAPTCHA recognizer with the Cycle-GAN approach that drew inspiration from the method devised by Ye et al [34]. The approach involves training a GAN to produce new CAPTCHAs, which are then used to train a solver network. Their model reported a high accuracy on multiple CAPTCHA datasets. In addition, Wang et al. proposed a fast CAPTCHA solver that effectively breaks complex text CAPTCHAs while using minimal labeled data [35]. It employs a GAN for simplifying image processing, resulting in a character accuracy rate of over 96%. GAN-based models usually utilize public datasets for pre-training, but the training cost greatly increases, making it not cost-effective for small tasks such as CAPTCHA Breaking [41].

2.4. CAPTCHA Recognition with Attention Mechanisms

The attention mechanisms allow networks to give more attention to certain channels and regions of the CAPTCHA. The Convolutional Block Attention Module (CBAM) conceived by Woo et al., is set to be integrated into any classification networks, which introduced channel attention and spatial attention mechanisms [36]. Zheng et al. focus on enhancing CAPTCHA recognition by integrating the CBAM with a baseline network [15]. Zi et al. proposed a model based on encoder-decoder architecture that employs an attention mechanism in conjunction with LSTM [37]. A novel transformer-based method is used for CAPTCHA identification by Shi et al [38]. The method involves character segmentation through optional image pre-processing to enhance accuracy, followed by reconstruction. As a spatial attention mechanism, Spatial Transformer Networks (STN) can correct the deformation and distortion of characters in CAPTCHAs [39].

In summary, in the preprocessing stage, the grayscale method without segmentation is a better option. For the recognition stage, GAN is inefficient in terms of training, while object detection methods require a large amount of manual annotation. Therefore, a model based on CNN and RNN architecture is the optimal combination with high accuracy and low complexity. However, ASR, FPS, and PARAMs must be fully considered when evaluating models running on CPU platforms.

3. Methods

As analyzed previously, every CAPTCHA image contains noise, interference, deformations, overlapping, and CCT. To address these problems for text-based CAPTCHA breaking, a model called Adaptive-CAPTCHA is raised that utilizes filters and RCNN, which borrow part of the structure from Deep-CAPTCHA, as shown in Figure 3. Compared with Deep-CAPTCHA, the proposed model has fewer model parameters, fewer convergence epochs, and higher recognition accuracy. Pos T, Pos 0, Pos 1, Pos2, and Pos3 respectively represent different positions of the model. These different positions can be short-circuited using residual connections, potentially improving the training speed [40].

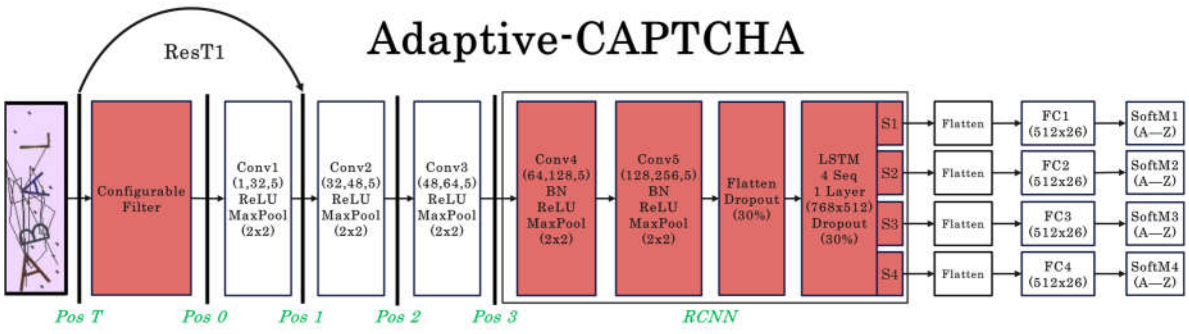


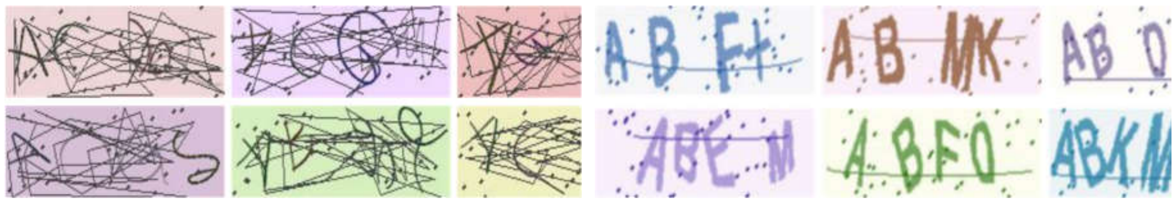
Figure 3. The networks of Adaptive-CAPTCHA.

In Adaptive-CAPTCHA, the parameter settings of the first three convolutional layers and FC layers are completely consistent with the Deep-CAPTCHA. Unlike Deep-CAPTCHA which uses fully connected layers with a huge number of parameters after conv3, our model introduces RCNN based on two convolutional layers and one LSTM layer to replace the FC layers. The subsequent experiments show that replacing can significantly reduce the number of parameters. Additionally, LSTM can model the correlation between CAPTCHA characters, thereby improving ASR.

The verification code is subject to interference and noise., and the filter module is specially designed to address the lack of processing in the original Deep-CAPTCHA. However, the inclusion of the filter layers must be adjusted according to the level of image noise. Otherwise, it may negatively impact the recognition of the characters.

3.1. Data Collection and Preprocessing

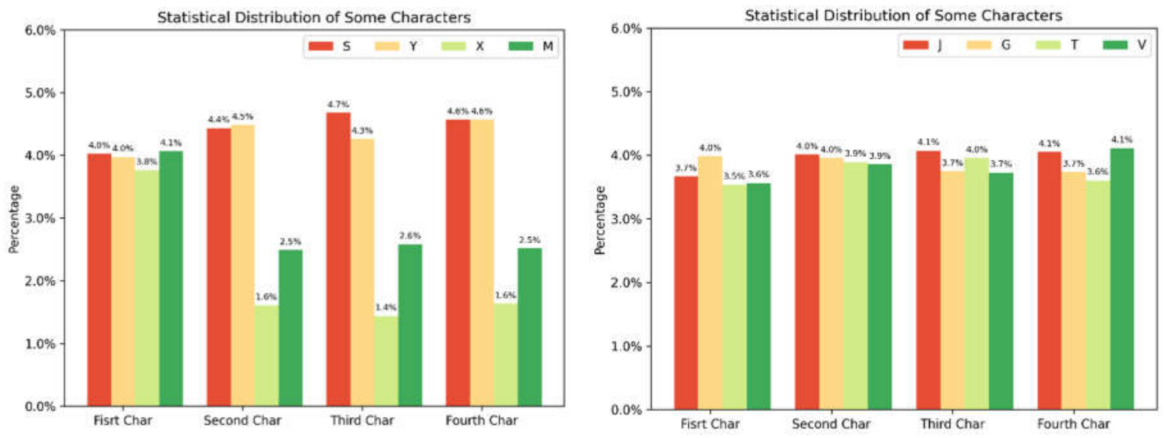
In this study, two datasets are adopted for performance analysis: the first is the public dataset M-CAPTCHA on the Kaggle (https://www.kaggle.com/datasets/sanluo/mcaptcha), and the second is a dataset generated based on the Python ImageCaptcha library, called P-CAPTCHA. M-CAPTCAH contains 25,000 images, while P-CAPTCHA has 20,000 CAPTCHAs. Every CAPTCHA image contains four characters from position one to four and each character is taken from the 26 uppercase English letters. Figure 4a,b show some samples in the M-CAPTCHA and P-CAPTCHA datasets respectively. As can be seen in the figures, images in P-CAPTCHA have little interference, noise, character distortion, and deformation, while the M-CAPTCHA dataset has a much stronger anti-attack mechanism, which includes more complex background interference, greater spatial deformation, and nonlinear distortion, making it very difficult to recognize.



(**a**) (**b**)

Figure 4. Samples of dataset: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Figure 5a,b show that M-CAPTCHA characters present a uniform distribution, while P-CAPTCHA characters are relatively uniformly distributed, indicating that there are correlations between the characters in the CAPTCHAs of M-CAPTCHA.



(**a**) (**b**)

Figure 5. Character statistical distribution: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Further analysis found that the M-dataset also has Markov Transition Probabilities (MTP) between characters, and the MTPs of some characters (from R to Z) in the training set are shown in Figure 6. The presence of MTPs leads to their first-order statistical distributions being closely related to the transition probabilities. Letters in the vertical coordinate represent a character, and letters in the horizontal coordinate represent the next adjacent character. The numerical values of their intersection are the MTPs of these two characters, which are zero in some grids, such as the MTP between the two consecutive letters W. The presence of MTPs indicates a strong correlation between characters, which can be better modeled by RNN compared with FC to improve recognition accuracy.

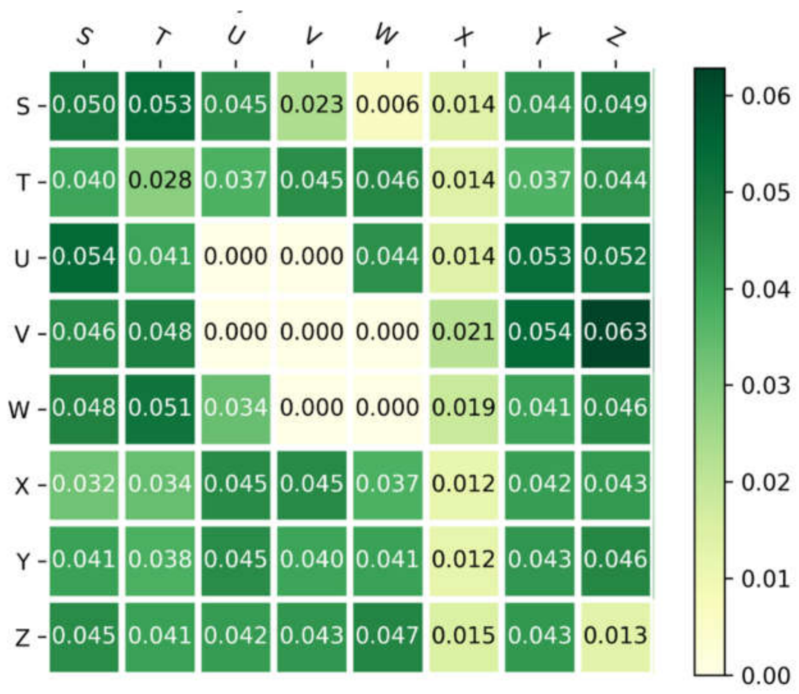


Figure 6. Markov transition probabilities between characters on the M-CAPTCHA.

Based on the above data analysis, it is necessary to model the inter-character correlation, so a CRNN module was proposed in this study. The images in both datasets are of the same size, with a width of 64 and a height of 192 pixels, thus requiring no scaling. However, grayscale processing of the images is necessary, which can greatly reduce the complexity of the model while maintaining the accuracy of CAPTCHA recognition. In addition, all the images need to be normalized for better training.

3.1. Filter Networks

CAPTCHA recognition is different from general image recognition tasks. To prevent web attacks, there are many artificially designed noise points and interference lines in the image. A robust filter network can help the subsequent networks to better extract the features of the CAPTCHA, thereby improving the recognition accuracy. In this study, filter networks based on encoder-decoder architecture with a configurable number of layers are added. Figure 7 shows an example of filter networks with layers $\mathrm{N} = 4$. Since the encoder and decoder appear in pairs, parameter $\mathrm{N} = 2\mathrm{K}$ takes an even number, where $\mathrm{K} \in \boldsymbol{Z}$. In filter networks, the output image size of each layer remains unchanged, which contains a convolutional layer, a Batch Normalization (BN) layer, and an activation function. The number of layers depends on the noise power of the target dataset, the more noise the larger the number of layers is set, and vice versa the fewer the number. The experiments in the next section will show that a reasonable number of filter network layers is required to improve the ASR of the model.

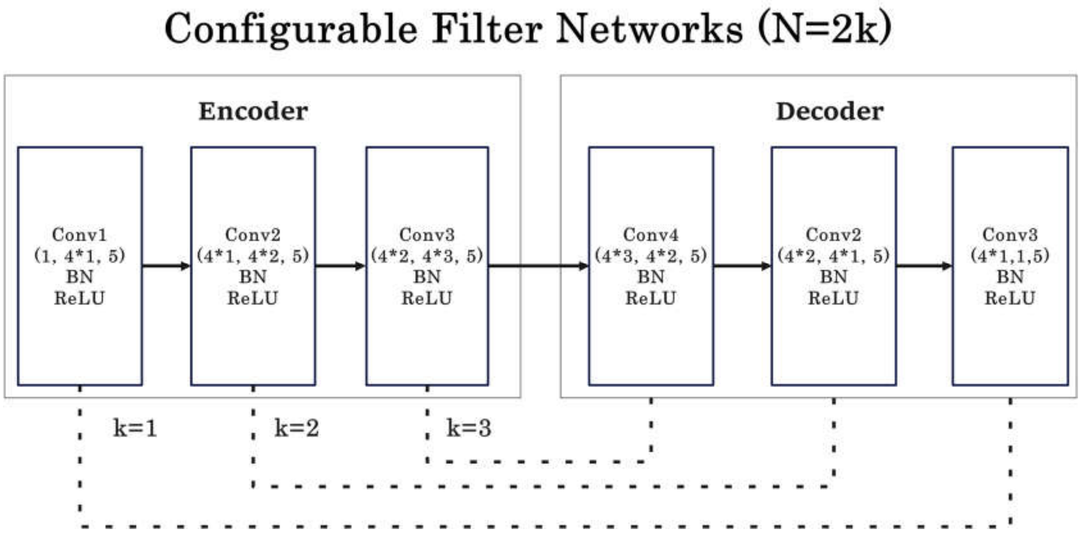

Configurable Filter Networks (N=2k)
Encoder
Decoder
Conv1 (1, 4*1, 5) BN ReLU
Conv2 (4*1, 4*2, 5) BN ReLU
Conv3 (4*2, 4*3, 5) BN ReLU
Conv4 (4*3, 4*2, 5) BN ReLU
Conv2 (4*2, 4*1, 5) BN ReLU
Conv3 (4*1,1,5) BN ReLU
k=1
k=2
k=3

Figure 7. Filter networks with the configurable number of layers.

3.2. Residual Connections

Different layers output feature maps of different sizes, which can be connected through residual connections to favor the gradient backpropagation. However, when using residuals, too many connections may lead to performance degradation for models with few layers like the text CAPTCHA recognizer. Multiple residual connections can confuse training signals during backpropagation, affecting gradient optimization. Secondly, an excessive number of residual connections make the model overfit the training data, which can negatively impact its ability to generalize to the test set. Additionally, excessive residuals harm the weights and learning rate, which may reduce the efficiency of the network parameters. Fourth, unnecessary residuals between specific layers can disrupt the original flow of information and introduce extra disturbances, especially if these layers transfer information well on their own.

In our model, the research experimentally selected some residual connections from Pos T to 3. Because our model has a small number of layers, it is important to carefully verify the impact of residuals on overall performance through experiments to find the optimal configuration.

3.3. CRNN Module

The RCNN module contains two convolutional layers, each followed by a BN layer and an activation function, as shown in Figure 3. These two convolutional layers are used to further downsample the features and extract higher-level semantic contents. The addition of the BN layer can accelerate the convergence of the model, which will be shown in the subsequent experimental section. The LSTM module adopts a single-layer structure, which divides the output of the convolutional layer into four parts as the input, each sequence represents a character. In this way, the dependencies between characters can be learned to combat interference problems between characters such as CCT, geometric deformation, and MTPs. A dropout layer is also added before and after the LSTM layer to prevent overfitting. This RCNN module is the key to model improvement, which not only greatly reduces the number of parameters, but also improves the ASR.

3.3. Loss Functions

In the image classification, the Cross-Entropy (CE) and Binary Cross-Entropy (BCE) are currently mostly used loss functions, and these functions are very suitable for backpropagation to update network weight parameters with the probability distribution input. If x_i represents the predicted probability and y_i represents the true label, Equation 1 shows the weighted BCE calculation formula for a multi-classification task. w_i represents the weight. If w_i is equal to 1, the above formula degenerates into a general BCE function.

$$L = -w_i\,[y_i log x_i + (1 - y_i) log(1 - x_i)] \tag{1}$$

Another loss function is focal loss, as shown in Equation 2. Among them, α is used to control the ratio of positive and negative samples, and γ is used to control the weight of difficult and easy samples. By setting different parameter values, different samples were given different weights.

$$L = -\alpha y_i (1 - x_i)^{\gamma} log x_i - (1 - \alpha)(1 - y_i) x_i^{\gamma} log(1 - x_i) \tag{2}$$

Mean Squared Error (MSE) loss is another loss function that calculates the squared difference between predicted probabilities and real labels. Its advantages include smooth gradients, which benefit optimization. Different from BCE, MSE is less sensitive to differences between probabilities for true categories due to its quadratic nature. However, BCE provides a stronger gradient signal for low-probability events, which is often desirable in classification tasks where the focus is on the probability of the correct class.

4. Results and Discussion

An ablation study was performed to demonstrate the performance by incorporating different modules into Deep-CAPTCHA. All the experiments were tested on M-CAPTCHA and P-CAPTCHA, each being divided into two subsets, with 80% devoted to training and 20% to testing. An Adam optimizer with a learning rate of 0.0001 was employed in the experiment. 130 epochs were enough for the model's convergence, so this value was adopted as the training epochs. Most experiments were primarily conducted on the Tesla T4 platform using the CUDA 12.1 environment, with some experiments performed on an Intel (R) Core (TM) i5-8265U CPU, such as FPS.

4.1. Visual Analysis of Filter Networks

Figure 8 shows that the filter networks had improved the recognition accuracy for all four positional characters on the M-dataset. There was an increase in AASR from around 85% to about 95%, with filter networks contributing approximately 10 percentage points on M-dataset. The first and fourth characters had a relatively high ASR for the CCT from only one adjacent character, while the second and third characters are affected by both sides. Therefore, the filter networks filtered out more noise and interference, resulting in a greater boost to ASR in the 2nd and 3rd positions.

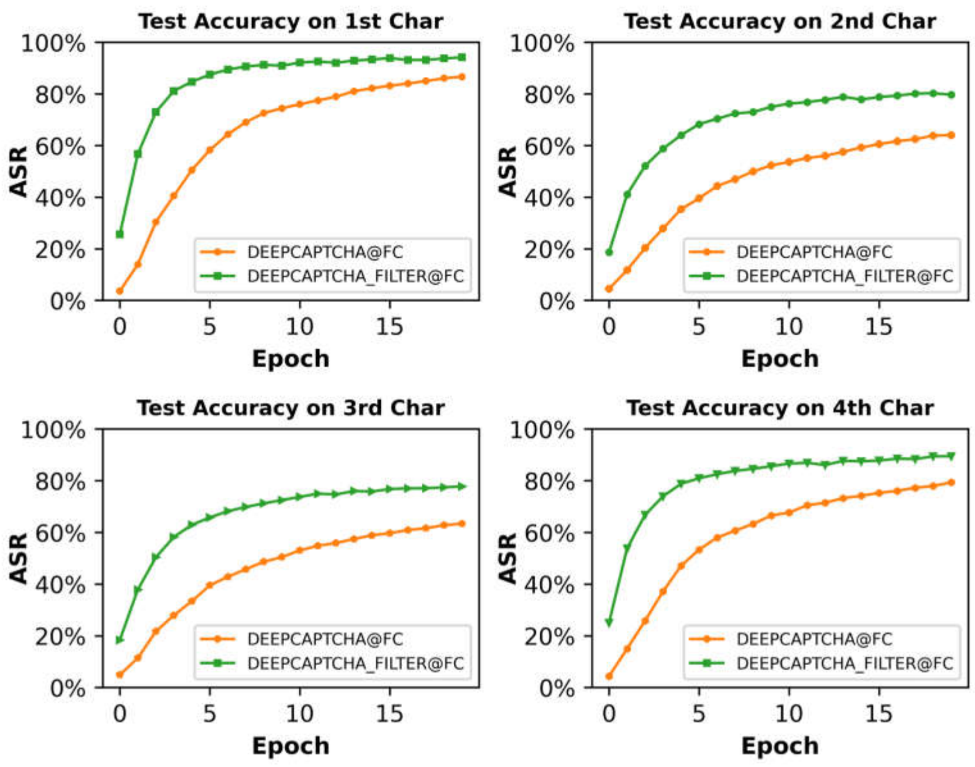


Figure 8. ASR with and without filter networks on the M-CAPTCHA.

Unlike the M-dataset, where ASR improves significantly with the addition of the filter networks, there was no change in ASR on the P-dataset, and a slight decrease was found on the second character, as shown in Figure 9.

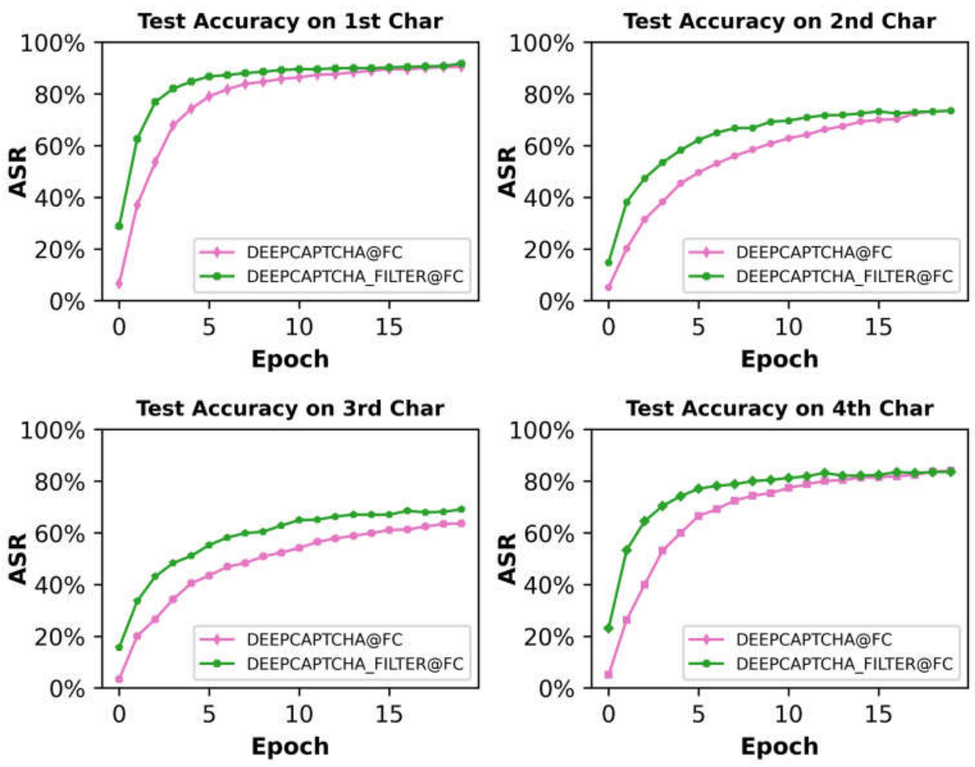
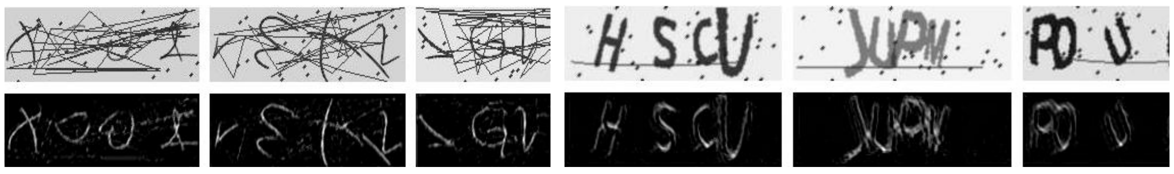


Figure 9. ASR with and without filter networks on the P-CAPTCHA.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 16 April 2024 doi:10.20944/preprints202404.1050.v1

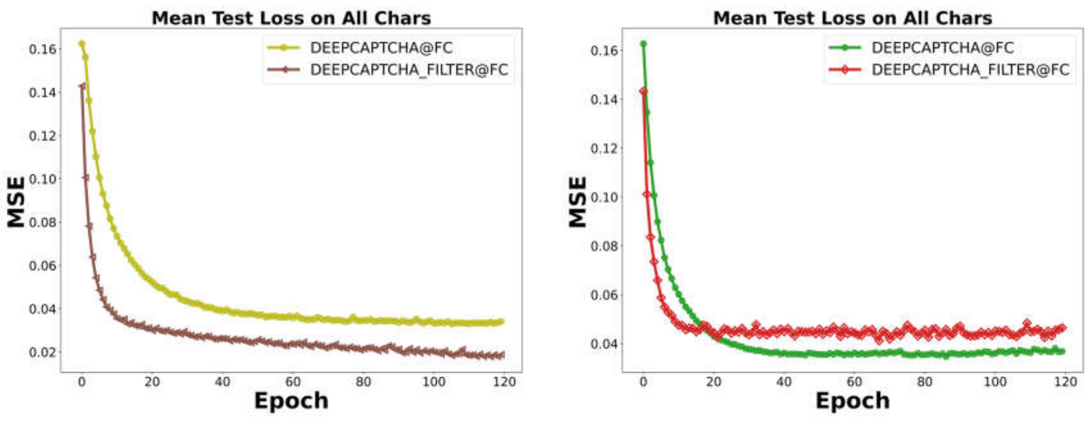
After undergoing the filtering process illustrated in Figure 10 (a), it is evident that the noise level in M-CAPTCHA images had markedly diminished. Compared with M-CAPTCHA, the reason for the lack of improvement in ASR is that the P-dataset contains less background noise and interference, the strong filter networks did not only filter out the noise but also did some damage to the characters in the original image as shown in Figure 10 (b). The design of filter networks is a double-edged sword, which must be balanced between the noise and characters. For images that contain a lot of interference and noise, it is necessary to increase the number of layers of the filter networks. On the contrary, less noise should reduce the number.



(**a**) (**b**)

Figure 10. Comparison of images before and after filtering: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

The loss with and without the filter networks on M-CAPTCHA and P-CAPTCHA are shown in Figures 11a,b, respectively. The curves in the two figures indicate that the convergence speeds increased after adding the filter networks, suggesting that the interference of two images had been reduced, which made it easier for the model to learn the features. However, the filter networks caused overfitting to the interference on the P-dataset, which affected the characters and led to an increase in MSE error.



a) (**b**)

Figure 11. Loss comparison before and after filtering: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Therefore, the number of filter network layers in Adaptive-CAPTCHA should be considered a hyperparameter. It is recommended to configure more filter layers for noisier CAPTCHAs and fewer for less noisy ones. Figure 12 shows the AASR when the filter network is configured with different numbers of layers, and the best AASR on the P-dataset was obtained when the layer was set to two.

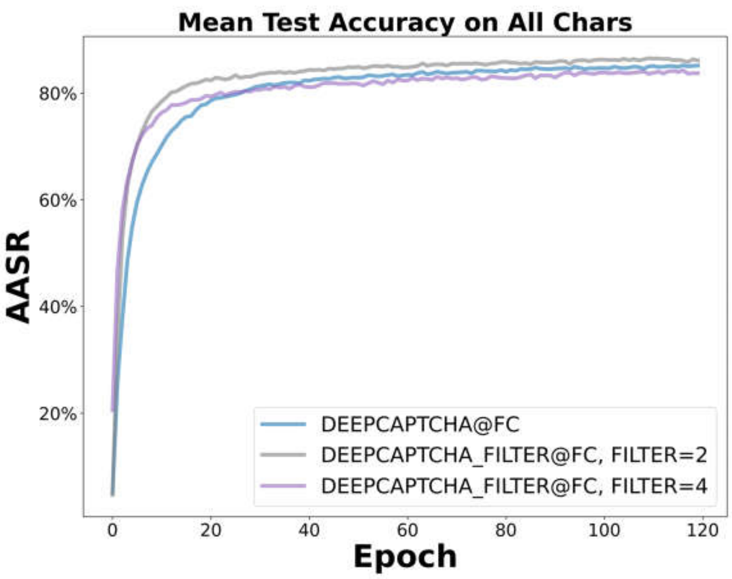
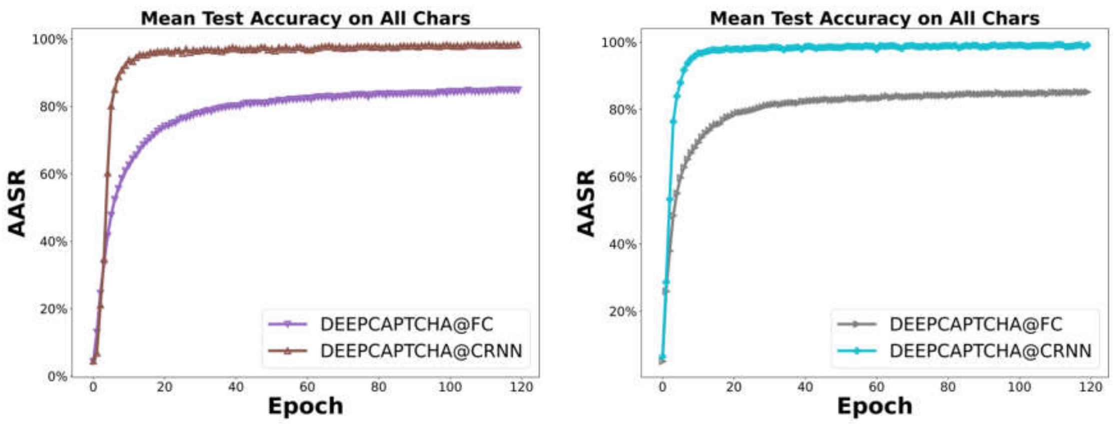


Figure 12. AASR on P-dataset with different layers of the filter networks.

4.2. Visual Analysis of CRNN Networks

To compare under the same benchmark, this experiment did not use the filter networks but only replaced the FC layers of Deep-CAPTCHA with CRNN networks, which contain two convolutional layers, a flatten layer, and an LSTM layer. The settings of all other layers and parameters were the same as Deep-CAPTCHA. The AASR on the M-CAPTCHA was increased to over 98%, as shown in Figure 13 (a). The CRNN layers improved the AASR on the P-dataset from about 85% to almost 99%, as shown in Figure 13 (b). These two results indicated that CRNN can more effectively extract CAPTCHA features and learn the correlation between characters caused by CCT and prior distribution regardless of whether the dataset has a lot of noise or not.



(**a**) (**b**)

Figure 13. ASR comparison using FC and CRNN: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

In CRNN, the CCN layers extract high-level semantic features, while the LSTM layers learn character relationships. In addition, the BN layers accelerate model learning, so removing them affects the convergence speed of the model. It can be found from Figure 14 that after using LSTM without BN layers, the AASR curve has a plateau area between about zero to five epoch intervals. Adding the BN layer speeds up gradient propagation and makes training more stable.

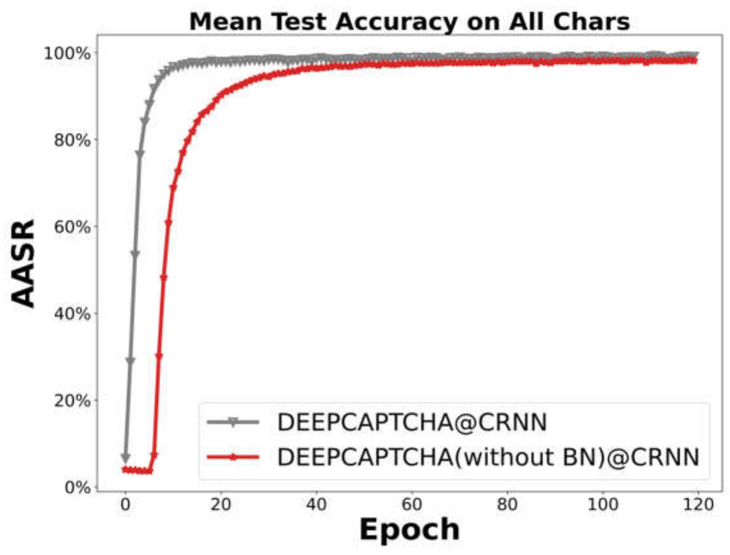


Figure 14. AASR comparison with and without BN in CRNN.

For the CRNN, increasing the number of layers of LSTM cannot improve the AASR since one layer of LSTM is enough to learn the sequence relations of CAPTCHA images, as shown in Figure 15. On the curve, one-layer LSTM even slightly outperformed two and three layers, which also had a minimum number of parameters.

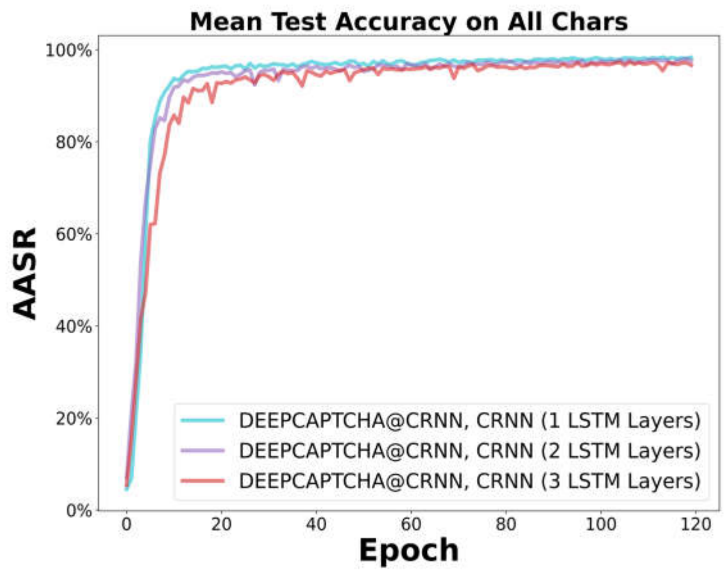
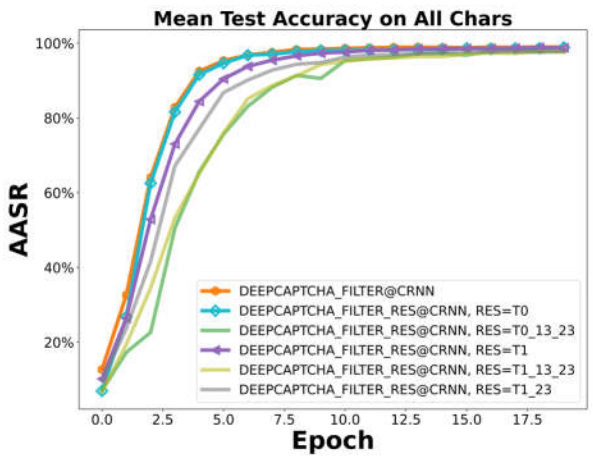


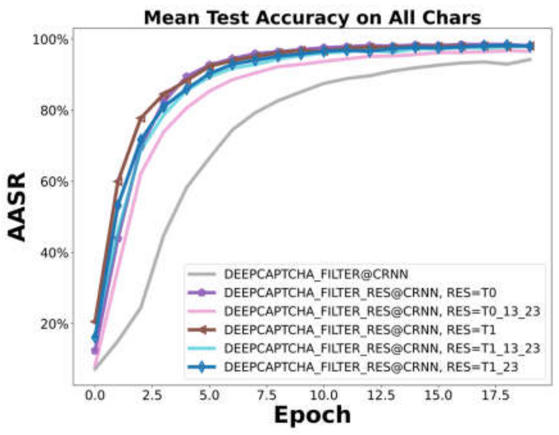
Figure 15. AASR with different layers of LSTM on P-dataset.

4.3. Visual Analysis of Residual Connections

Residual connections are used in neural networks to help address the vanishing gradient problem, particularly in networks with many layers. This situation occurs during training when the gradients become very small, making it difficult for the network to learn effectively. By adding residual connections, the network can bypass certain layers, allowing the gradients to flow more easily through the network. Overall, residual connections can help improve the training and convergence of deep neural networks, particularly in cases where the network has many layers and may encounter the vanishing gradient problem. Since the proposed networks add the filter, CNN, LSTM, and BN layers based on Deep-CAPTCHA, which increases the number of layers in the entire network, it is necessary to add residual connections. Figure 16(a) shows that on the M-CAPTHA when one residual T0 or T1 is added compared to no residuals, the convergence rate is comparable, while once more than two residuals are added it leads to overfitting. This is because the model can be trained relatively well, adding too many residuals instead hinders the gradient propagation. However, the residuals compensate for the corruption of the filter networks on the characters of the dataset, allowing for a smoother flow of information and therefore improving the results, as shown in Figure 16 (b).



(**a**) M-dataset



(**b**) P-dataset

Figure 16. AASR with different residual connections: (**a**) M-CAPTCHA (**b**) P-CAPTCHA.

4.4. Visual Analysis of Loss Functions

The empirical observations reveal that among the various loss functions examined, including BCE, MSE, CE, and focal loss, there were no significant performance discrepancies, as shown in Figure 17. Surprisingly, the BCE function exhibits a slight performance edge over its counterparts. Given the lack of pronounced performance benefits and the increased computational expense of focal loss, the original BCE function emerges as the more prudent choice for the model.

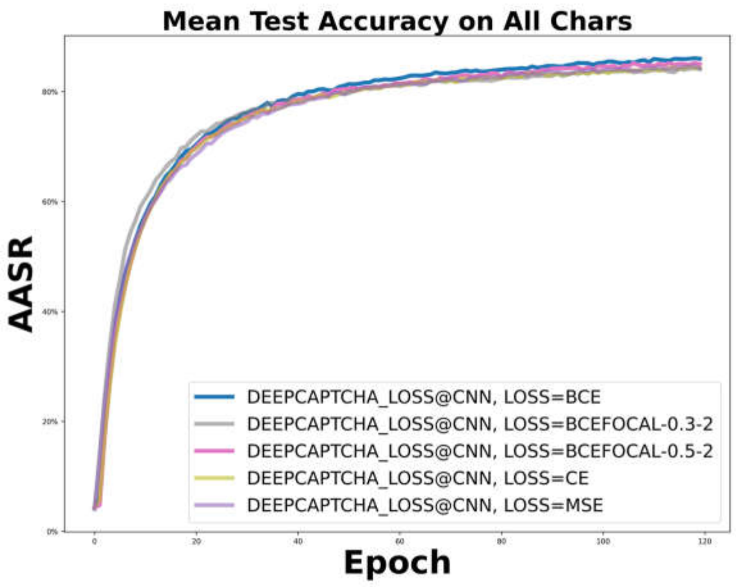


Figure 17. AASR with different loss functions.

To better visualize the results of this study, a confusion matrix was employed. Figure 18 depicts the AASR confusion matrix of Adaptive-CAPTCHA on the M-dataset. The results suggested a better degree of precision, as evidenced by the misclassification rate of the proposed method for characters being less than 1%. The figure shows that the letter O is easily recognized as Q, and Z is often misidentified as S. Therefore, by examining the confusion matrix, the model could be allowed to further improve the AASR by using active learning for some characters that are easily misidentified.

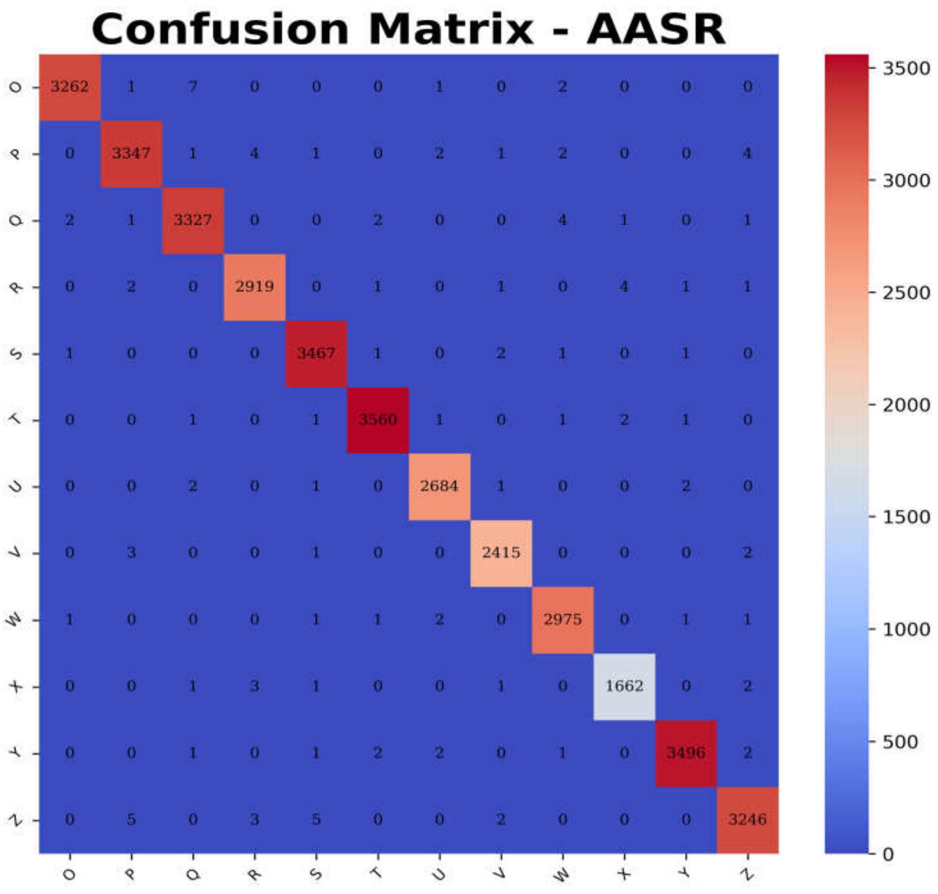


Figure 18. AASR confusion matrix of Adaptive-CAPTCHA on M-dataset.

4.4. Ablation Study

In this study, ablation experiments were conducted on STN, filter networks, and residual connections. A comprehensive performance was compared in terms of FPS, AASR, MACs, Params, and convergence speed, as shown in Table 1. Considering that CAPTCHA recognizers as part of a crawler usually run on personal computers, all FPS were measured on a CPU platform. In both datasets, the number of model parameters using CRNN was reduced by almost 41% compared to the PARAMs using FC, while the recognition accuracy was improved by more than 12%. This shows that both on M-CAPTCHA with a priori statistical distributions and complex backgrounds and on the simple and low noisy P-CAPTCHA, CRNN has a clear advantage in terms of accuracy and complexity.

Although the STN can theoretically be used to correct the geometric deformation of characters, the experimental results are not ideal by only promoting a little AASR on M-CAPTCHA but increasing the number of parameters. The reason is that there are many irregular and non-linear deformations that are not uniform across a single CAPTCHA making STN difficult to calibrate. The performance improvement of STN for the P-CAPTCHA is noticeable from around 85% to 95%, which is due to the low noise and the regular geometrical deformation of character strokes. However, the 95% ASR of STN on simple datasets is still not as good as the 99% of CRNN, not to mention the fact that STN is ineffective on complex datasets, and therefore STN is not the best choice for text CAPTCHA breaking.

The Adaptive-CAPTCHA converges much faster than Deep-CAPTCHA on both datasets. Among all possible residual connections in experimental results, the model with residual connection T1 had a convergence speed of 40 epochs on the M-dataset and 20 epochs on the P-dataset while maintaining the highest accuracy and the minimum number of parameters. From the table, for a low-complexity network such as Adaptive-CAPTCHA, too many residual connections may destroy the gradient propagation instead of decreasing the convergence speed.

The table shows that the Deep-CAPTCHA had advantages in FPS and MACs as shown in the table, which indicates there was a lower computational complexity. However, for real-time data collection tasks, the FPS value of 72 for Adaptive-CAPTCHA was sufficient. For the Deep-CAPTCHA, there was an obvious shortcoming in PARAMs, which was about 70% higher than Adaptive-CAPTCHA. The proposed model Adaptive-CAPTCHA performed well in terms of AASR, PARAMs, and convergence speed while keeping proper performance in FPS and MACs, which can meet the requirements of web crawlers and security vulnerability assessment.

Table 1. Ablation study.

No.	Models	FPS	MACs	Params	AASR M-dataset	AASR P-dataset	CEPOCH M-dataset	CEPOCH P-dataset
O	Deep-CAPTCHA (Baseline)	126	193.1M	6.46M	85%	85%	120	120
A	Baseline + CRNN	109	276.1M	3.82M	98%	99%	110	110
B	A + Filters (4 layers)	77	319.1M	3.82M	99%	98%	30	100
E	B + R(T0)	77	319.1M	3.82M	99%	99%	50	30
F	B + R(T0_13_23)	71	320.2M	3.82M	99%	99%	70	20
G	B + R(T0_T1)	67	319.6M	3.82M	99%	99%	70	70
H	B + R(T0_T1_13_23)	66	320.7M	3.82M	99%	99%	60	20
I	B + R(T1) (Adaptive-CAPTCHA)	72	319.6M	3.82M	99%	99%	40	20
J	B + R(T1_13_23)	70	320.7M	3.82M	99%	98%	80	40
L	B + R(T1_23)	70	320.3M	3.82M	99%	99%	70	80
M	O + Filters	81	236.1M	6.46M	93%	84%	100	120
N	O + STN	67	226.2M	6.53M	62%	95%	120	60
O	O + Filters + STN	60	269.2M	6.53M	89%	97%	110	60

4. Conclusions

In conclusion, a novel text-CAPTCHA model was proposed in this research based on a comprehensive enhancement of the Deep-CAPTCHA through means of experiments. The integration of additional filtering networks suppresses background noise and interference in CAPTCHA images while overpowered filter networks can destroy character strokes in an image. Based on the experiments, configurable filtering networks based on encoder-decoder architecture were proposed whose number of layers was set according to the noise level of the target CAPTCHA dataset. A notable reduction in the model's parameter was achieved by substituting the original model's FC layers with a novel CRNN, which significantly increased the model's capability to capture inter-character dependencies, thus elevating the AASR on both datasets with different complexity. In addition, a series of ablation studies have been carried out to examine the impact of different components such as STN, layers of filter networks, LSTM, and residual connections. We also have provided an extensive comparison among various models, assessing pivotal performance metrics including the PARAMs, FPS, MACs, and AASR. Empirical evidence demonstrates that our proposed Adaptive-CAPTCHA model achieves high accuracy with a substantially reduced number of parameters. As a direction for future work, we suggest exploration into variable-length character CAPTCHAs, which present challenges in CAPTCHA recognition tasks. We also advocate for the application of Bayesian estimation to the realm of CAPTCHA recognition.

References

1. von Ahn, L.; Blum, M.; Langford, J. Telling humans and computers apart automatically. *Commun. ACM* **2004**, *47*, 56–60. 10.1145/966389.966390.
2. von Ahn, L.; Blum, M.; Hopper, N.J.; Langford, J. CAPTCHA: Using Hard AI Problems for Security. In Proceedings of the Advances in Cryptology — EUROCRYPT 2003; Biham, E., Ed.; Springer: Berlin, Heidelberg, 2003; pp. 294–311. 10.1007/3-540-39200-9_18.
3. Che, A.; Liu, Y.; Xiao, H.; Wang, H.; Zhang, K.; Dai, H.-N. Augmented Data Selector to Initiate Text-Based CAPTCHA Attack. *Secur. Commun. Netw.* **2021**, *2021*, e9930608. 10.1155/2021/9930608.
4. Noury, Z.; Rezaei, M. Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment 2020. 10.48550/arXiv.2006.08296.
5. Baird, H.S.; Popat, K. Human Interactive Proofs and Document Image Analysis. In Proceedings of the Document Analysis Systems V; Lopresti, D., Hu, J., Kashi, R., Eds.; Springer: Berlin, Heidelberg, 2002; pp. 507–518. 10.1007/3-540-45869-7_54.
6. Baykara, M.; Alnıak, F.; Çınar, K. Review and comparison of captcha approaches and a new captcha model. In Proceedings of the 2018 6th International Symposium on Digital Forensic and Security (ISDFS); 2018; pp. 1–6. 10.1109/ISDFS.2018.8355316.
7. Bostik, O.; Klecka, J. Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms. *IFAC-Pap.* **2018**, *51*, 208–213. 10.1016/j.ifacol.2018.07.155.
8. Wang, P.; Gao, H.; Guo, X.; Xiao, C.; Qi, F.; Yan, Z. An Experimental Investigation of Text-based CAPTCHA Attacks and Their Robustness. *ACM Comput. Surv.* **2022**. 10.1145/3559754.
9. Alsuhibany, S.A. Optimising CAPTCHA Generation. In Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security; 2011; pp. 740–745. 10.1109/ARES.2011.114.
10. San Luo M-CAPTCHA 2023. https://doi.org/10.34740/KAGGLE/DS/3991186.
11. Weng, H.; Zhao, B.; Ji, S.; Chen, J.; Wang, T.; He, Q.; Beyah, R. Towards understanding the security of modern image captchas and underground captcha-solving services. *Big Data Min. Anal.* **2019**, *2*, 118–144. 10.26599/BDMA.2019.9020001.
12. O, I.E.; A, A.A.; A, I.F.; O, A.M.; Oludayo, O.O. Research trends on CAPTCHA: A systematic literature. *Int. J. Electr. Comput. Eng. IJECE* **2021**, *11*, 4300–4312. 10.11591/ijece.v11i5.pp4300-4312.
13. UmaMaheswari, P.; Ezhilarasi, S.; Harish, P.; Gowrishankar, B.; Sanjiv, S. Designing a Text-based CAPTCHA Breaker and Solver by using Deep Learning Techniques. In Proceedings of the 2020 IEEE International Conference on Advances and Developments in Electrical and Electronics Engineering (ICADEE); 2020; pp. 1–6. 10.1109/ICADEE51157.2020.9368949.
14. Ye, G.; Tang, Z.; Fang, D.; Zhu, Z.; Feng, Y.; Xu, P.; Chen, X.; Wang, Z. Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In Proceedings of the Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security; Association for Computing Machinery: New York, NY, USA, 2018; pp. 332–348. 10.1145/3243734.3243754.
15. Zheng, Y. Captcha Recognition Based on Attention Mechanism. In Proceedings of the The 6th International Conference on Control Engineering and Artificial Intelligence; Association for Computing Machinery: New York, NY, USA, 2022; pp. 119–124. 10.1145/3522749.3523077.
16. Chen, Y.; Luo, X.; Xu, S.; Chen, R. CaptchaGG: A linear graphical CAPTCHA recognition model based on CNN and RNN. In Proceedings of the 2022 9th International Conference on Digital Home (ICDH); 2022; pp. 175–180. 10.1109/ICDH57206.2022.00034.
17. Xing, W.; Mohd, M.R.S.; Johari, J.; Ruslan, F.A. A Review on Text-based CAPTCHA Breaking Based on Deep Learning Methods. In Proceedings of the 2023 International Conference on Computer Engineering and Distance Learning (CEDL); 2023; pp. 171–175. 10.1109/CEDL60560.2023.00040.
18. Zhang, Y.; Zhang, C. A new algorithm for character segmentation of license plate. In Proceedings of the IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683); 2003; pp. 106–109. 10.1109/IVS.2003.1212892.
19. Wang, J.; Qin, J.; Xiang, X.; Tan, Y.; Pan, N.; College of Computer Science and Information Technology, Central South University of Forestry and Technology, 498 shaoshan S Rd, Changsha, 410004, China CAPTCHA recognition based on deep convolutional neural network. *Math. Biosci. Eng.* **2019**, *16*, 5851–5861. 10.3934/mbe.2019292.
20. Wang, Z.; Shi, P. CAPTCHA Recognition Method Based on CNN with Focal Loss. *Complexity* **2021**, *2021*, e6641329. 10.1155/2021/6641329.
21. Lu, S.; Huang, K.; Meraj, T.; Rauf, H.T. A novel CAPTCHA solver framework using deep skipping Convolutional Neural Networks. *PeerJ Comput. Sci.* **2022**, *8*, e879. 10.7717/peerj-cs.879.
22. Mocanu, I.G.; Yang, Z.; Belle, V. Breaking CAPTCHA with Capsule Networks. *Neural Netw.* **2022**, *154*, 246–254. 10.1016/j.neunet.2022.06.041.
23. Shi, B.; Bai, X.; Yao, C. An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition 2015.
24. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey 2019.

25. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the Computer Vision – ECCV 2016; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, 2016; pp. 21–37. 10.1007/978-3-319-46448-0_2.
26. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement 2018. 10.48550/arXiv.1804.02767.
27. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. 10.1109/TPAMI.2016.2577031.
28. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors 2022. 10.48550/arXiv.2207.02696.
29. Du, F.-L.; Li, J.-X.; Yang, Z.; Chen, P.; Wang, B.; Zhang, J. Captcha recognition based on faster R-CNN. *Lect. Notes Comput. Sci. Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinforma.* **2017**, *10362 LNCS*, 597–605. 10.1007/978-3-319-63312-1_52.
30. Nian, J.; Wang, P.; Gao, H.; Guo, X. A deep learning-based attack on text CAPTCHAs by using object detection techniques. *IET Inf. Secur.* **2022**, *16*, 97–110. 10.1049/ise2.12047.
31. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers 2020. 10.48550/arXiv.2005.12872.
32. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein Generative Adversarial Networks. In Proceedings of the Proceedings of the 34th International Conference on Machine Learning; PMLR, 2017; pp. 214–223.
33. Ye, G.; Tang, Z.; Fang, D.; Zhu, Z.; Feng, Y.; Xu, P.; Chen, X.; Han, J.; Wang, Z. Using Generative Adversarial Networks to Break and Protect Text Captchas. *ACM Trans. Priv. Secur.* **2020**, *23*, 7:1-7:29. 10.1145/3378446.
34. Zhu, J.-Y.; Park, T.; Isola, P.; Efros, A.A. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV); 2017; pp. 2242–2251. 10.1109/ICCV.2017.244.
35. Wang, Y.; Wei, Y.; Zhang, M.; Liu, Y.; Wang, B. Make complex CAPTCHAs simple: A fast text captcha solver based on a small number of samples. *Inf. Sci.* **2021**, *578*, 181–194. 10.1016/j.ins.2021.07.040.
36. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module 2018. 10.48550/arXiv.1807.06521.
37. Li, C.; Chen, X.; Wang, H.; Wang, P.; Zhang, Y.; Wang, W. End-to-end attack on text-based CAPTCHAs based on cycle-consistent generative adversarial network. *Neurocomputing* **2021**, *433*, 223–236. 10.1016/j.neucom.2020.11.057.
38. Shi, Y.; Liu, X.; Han, S.; Lu, Y.; Zhang, X. A Transformer Network for CAPTCHA Recognition. In Proceedings of the 2021 2nd International Conference on Artificial Intelligence and Information Systems; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1–5. 10.1145/3469213.3470366.
39. Chan, K.-H.; Im, S.-K.; Ian, V.-K.; Chan, K.-M.; Ke, W. Enhancement Spatial Transformer Networks for Text Classification. In Proceedings of the Proceedings of the 4th International Conference on Graphics and Signal Processing; Association for Computing Machinery: New York, NY, USA, 2020; pp. 5–10. 10.1145/3406971.3406981.
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2016; pp. 770–778. 10.1109/CVPR.2016.90.