applied sciences

MDPI

Article

Adaptive CAPTCHA: A CRNN-Based Text CAPTCHA Solver with Adaptive Fusion 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 Selangor 40450, Malaysia; julia893@uitm.edu.my

[2] School of Intelligent Manufacturing, Leshan Vocational and Technical College, Leshan 614000, China

* Correspondence: 2022995467@student.utim.edu.my (X.W.); fazlina419@uitm.edu.my (F.A.R.)

Abstract: Text-based CAPTCHAs remain 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 and 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 CNN combined with an RNN (CRNN) module and trainable Adaptive Fusion Filtering Networks (AFFN), which effectively handle the interference and learn the correlation between characters in CAPTCHAs to enhance recognition accuracy. Experimental results on two datasets of different complexities 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. Compared with several of the latest models, the model proposed by the study also has better comprehensive performance.

Keywords: CAPTCHA recognition; noise; interference; filter; LSTM; resistance mechanisms

check for updates

Citation: Wan, X.; Johari, J.; Ruslan, F.A. Adaptive CAPTCHA: A CRNN-Based Text CAPTCHA Solver with Adaptive Fusion Filter Networks. *Appl. Sci.* **2024**, *14*, 5016. https://doi.org/10.3390/app14125016

Academic Editors: George Drosatos, Konstantinos Rantos, Konstantinos Demertzis and Keun Ho Ryu

Received: 24 April 2024
Revised: 18 May 2024
Accepted: 6 June 2024
Published: 8 June 2024

1. Introduction

The Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA), which is recognized as a Human Interactive Proof (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 the Deep CAPTCHA [4]. The pervasive incorporation of digital and Roman characters in text-based CAPTCHAs could render them susceptible to interpretation by Optical Character Recogniton (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].

The research on the CAPTCHA recognition algorithm can help websites and CAPTCHA designers find the vulnerabilities of CAPTCHAs. By using the proposed CAPTCHA recognition engines, companies and developers may evaluate the security of the website's CAPTCHAs and block dangerous vulnerabilities in time to avoid being attacked. From a commercial point of view, the CAPTCHA recognition algorithms can be embedded into network security assessment toolkits to evaluate the security of websites. In addition, the research on CAPTCHA recognition may be used for reference in scene text recognition, object detection, and OCR.

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 [8]. CNNs are the most used networks as they excel at learning spatial hierarchies in an image, thereby providing a robust way for CAPTCHA recognition [9]. Recurrent Neural Network (RNN) or Long Short-Term Memory (LSTM) is another neural network that has shown impressive recognition accuracy in CAPTCHAs with Connecting Characters Together (CCT), as it can remember patterns over lengthy sequences [10]. Combined with CNNs and RNNs, the attention mechanisms allow the neural network to focus on certain important features, spatial positions, and channels, thereby improving the recognition effect [11]. Additionally, some content generation techniques have shown potential for enhancing models by creating believable CAPTCHA–like images to train a more robust network [12]. 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 [13].

The CAPTCHA recognition model is commonly utilized as a module by web scrawlers running on personal CPU platforms for data collection tasks. Therefore, it is necessary to reduce Parameters (PARAMs) while maintaining the Average Attack Success Rate (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. It is worth mentioning that performance evaluation is generally tested on public datasets.

The primary contribution of this 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 adaptive fusion filter networks are integrated into the model to combat the noise and interference presented in CAPTCHAs.
- A CRNN 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.
- Compared with other works, the method proposed in this study achieves a compact real-time CAPTCHA breaker with extremely low complexity and high AASR.

2. Related Works

In the past, CAPTCHA–breaking methods were mostly used in 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 CNNs, RNNs, Generative Adversarial Networks (GANs), 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 [14]. 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 its 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 [15].

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, the predicted results are outputted through neural networks, which consist of a combination of different technologies such as CNNs, RNNs, and GANs.

2.1. CAPTCHA Recognition with CNN and RNN

The most powerful classification networks, CNNs and RNNs, serve as the foundation for many CAPTCHA recognition networks. Leveraging the Dense Convolutional Network (DenseNet) with cross-layer connections, Wang et al. introduced modified networks for CAPTCHA recognition [16]. They reduced the number of convolutional blocks and developed 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 [17]. 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 [18]. A drawback of this technique is the requirement to first segment characters, with the process relying on manually configured operators, which poses a challenge when dealing with CAPTCHAs that contain overlapping characters. Shi et al. proposed a CRNN based on the Connectionist Temporal Classification (CTC) loss function and an RNN, which improved the detection ability of variable-length characters [19].

More recently, capsule networks have been used due to their capability of preserving detailed information about the input [20]. Nevertheless, this approach is extremely computationally intensive, and later trials indicate that the Attack Success Rate (ASR) is not good for CAPTCHAs with significant levels of noise. Ke Qing et al. introduced a network called PosConv that utilizes the positional information in the character sequence without using an RNN. This network employs a unique padding method and modified convolution to directly incorporate the relative position into the local features of letters [21]. In 2022, Aditya Atri and his colleagues employed the Depth First Search (DFS) method to extract characters from CAPTCHAs, called DeCAPTCHA [22]. They then utilized a CNN to recognize these extracted characters. However, the process of segmentation is significantly impacted by the presence of noise and interference, as well as the complexity of the calculations involved. Ke Qing et al. introduced a ConvNet that incorporates a unique group convolution operation across the width of the image [23]. This operation effectively reduces unnecessary calculations, resulting in enhanced performance. In 2023, Rajat Subhra Bhowmick and his colleagues assessed the weaknesses in the CAPTCHA systems of government websites. The authors provide an innovative neural network structure designed to effectively solve CAPTCHAs by incorporating textual instructions into the image [24]. Mukhtar Opeyemi Yusuf et al. introduced a multiview deep learning method for cracking CAPTCHAs. This approach utilizes correlational characteristics from many perspectives to enhance the model's ability to generalize and accurately classify [25]. A method was proposed by Zaid Derea et al. for recognizing CAPTCHAs that involves making several copies of the original CAPTCHA pictures and producing individual binary images with the precise positions of each set of CAPTCHA letters [26]. Soumen Sinha et al. developed a technique to bypass CAPTCHAs using a sequential CNN model known as CAP-SECURE, which is employed to assess the flaws and susceptibilities of websites [27].

Deep CAPTCHA is a state-of-the-art model that is used in the field of text-based CAPTCHA recognition, as proposed by Zahra Noury et al. in 2020. 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 1. 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 number 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.

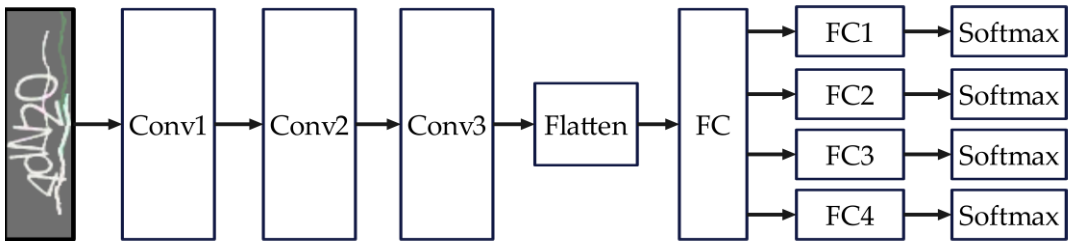


Figure 1. The Network of Deep CAPTCHA.

2.2. CAPTCHA Recognition with Object Detection Networks

The objective of an object detection algorithm is to recognize and spatially locate objects in images [28]. 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 [29–32]. Du et al. demonstrate that Faster R-CNN can effectively extract feature maps, enabling accurate recognition of the characters and their locations in CAPTCHA images [33]. 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 [34]. In 2020, N. Carion et al. adopted end-to-end transformer-based detectors, which have great application potential in the field of CAPTCHA breaking [35]. 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 Based on 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 [36]. 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 [11]. In 2020, the authors used GAN-based cracking methods to evaluate 33 different verification code datasets and achieved high accuracy [37]. In 2021, Zhuet al. introduced a CAPTCHA recognizer with the Cycle–GAN approach that drew inspiration from the method devised by Ye et al. [38]. The approach involves training a GAN to produce new CAPTCHAs, which are then used to train a solver network. Their model reported 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 [39]. 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. In 2020, Haitian Chen and colleagues introduced a novel CAPTCHA known as StyleCAPTCHA, which uses neural style transfer to produce CAPTCHAs [40]. In 2022, the framework by Ning Zhang et al. employs a GAN to combat interference and background noise and incorporates an improved character segmentation method to handle CAPTCHA pictures with varying character lengths [41]. Turhan Kimbrough introduced a novel framework to enhance feature extraction in CAPTCHAs by assigning several labels to each CAPTCHA. This approach involved generating training datasets using various CAPTCHA methods and employing a pre-processing methodology [42].

2.4. CAPTCHA Recognition with Attention Mechanisms and Transformers

Attention mechanisms allow networks to give more attention to certain channels and regions of a CAPTCHA. The Convolutional Block Attention Module (CBAM), conceived by Woo et al., is set to be integrated into any classification network, and introduces channel attention and spatial attention mechanisms [43]. Zheng et al. focus on enhancing CAPTCHA recognition by integrating the CBAM with a baseline network [12]. Zi et al. proposed a model based on an encoder–decoder architecture that employs an attention mechanism in conjunction with LSTM [44]. A novel transformer-based method was used for CAPTCHA identification by Shi et al. [45]. The method involves character segmentation through optional image pre-processing to enhance accuracy, followed by reconstruction. As a spatial attention mechanism, Spatial Transformer Networks (STNs) can correct the deformation and distortion of characters in CAPTCHAs [46]. Zhao et al. developed GEE-SOLVER, a method for solving text-based captchas using self-supervised learning. They utilized masked autoencoders to learn the hidden features of unmasked pictures. However, this approach consists of a vision transformer encoder that requires significant computational resources [47]. In 2023, Li et al. developed a task for predicting and reconstructing CAPTCHA images, which enables unsupervised feature encoding that implicitly represents the spatial domain properties of the images [48]. Raghavendra Hallya et al. introduced global and local attention methods for both transfer learning and parameter search models. However, the parameter search needs a lot of computation and is time-consuming [49].

In summary, in the preprocessing stage, the grayscale method without segmentation is a better option. For the recognition stage, a 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 architectures is the optimal combination of high accuracy and low complexity. However, ASR, FPS, and PARAMs must be fully considered when evaluating models running on CPU platforms.

3. Methods

For text-based CAPTCHA designers, it is necessary to balance both the resistance mechanisms against web attacks and the usability of website users [50]. To increase anti-attack ability without reducing the user's recognition time, CCT and deformation are used in the CAPTCHA design [51]. 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 2. To avoid confusion, designers try to reduce confusing characters when designing CAPTCHAs, resulting in Markov Transition Probabilities (MTPs) between adjacent characters. For this reason, some datasets present the character sequence dependence between two consecutive characters, such as the public dataset M-CAPTCHA [52]. For these CAPTCHAs, break models must consider how to process this correlation between adjacent characters.

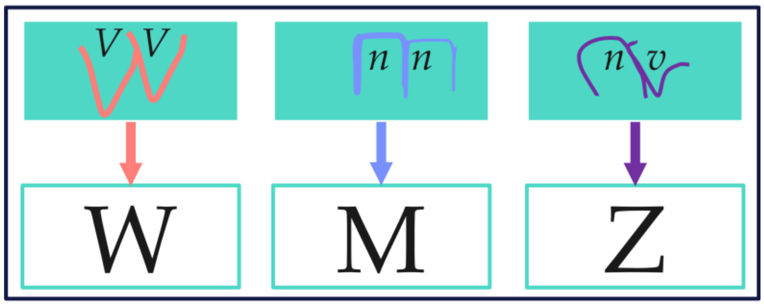


Figure 2. Some confusing adjacent characters in CAPTCHAs.

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 CRNN, 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, Pos 2, and Pos 3 represent different output positions of the model. These different positions can be short-circuited using residual connections, potentially improving the training speed [53].

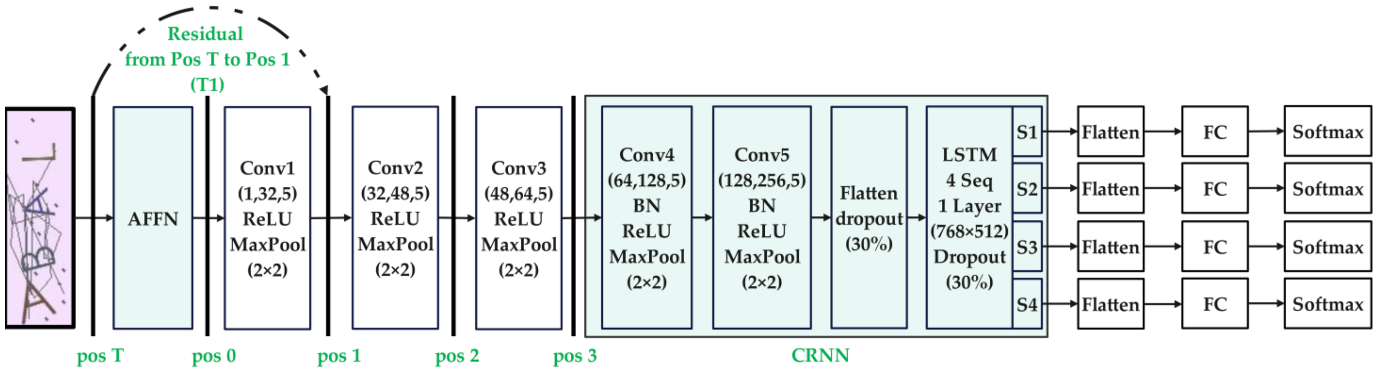

Residual
from Pos T to Pos 1
(T1)
AFFN
Conv1 (1,32,5) ReLU MaxPool (2×2)
Conv2 (32,48,5) ReLU MaxPool (2×2)
Conv3 (48,64,5) ReLU MaxPool (2×2)
Conv4 (64,128,5) BN ReLU MaxPool (2×2)
Conv5 (128,256,5) BN ReLU MaxPool (2×2)
Flatten dropout (30%)
LSTM 4 Seq 1 Layer (768×512) Dropout (30%)
S1
S2
S3
S4
Flatten
FC
Softmax
pos T
pos 0
pos 1
pos 2
pos 3
CRNN

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 Deep CAPTCHA. Unlike Deep CAPTCHA, which uses fully connected layers with a huge number of parameters after conv3, our model introduces CRNN based on two convolutional layers and one LSTM layer to replace the FC layers. The subsequent experiments show that replacing the full connected layers 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 evaluation: the first is the public dataset M-CAPTCHA on Kaggle "https://www.kaggle.com/datasets/sanluo/mcaptcha (accessed on 5 February 2024)", and the second is a dataset generated based on the Python ImageCaptcha library, called P-CAPTCHA. M-CAPTCHA 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.

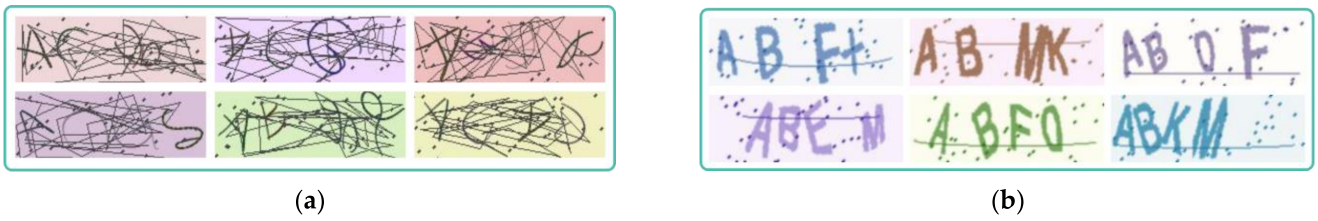


Figure 4. Samples of dataset: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

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

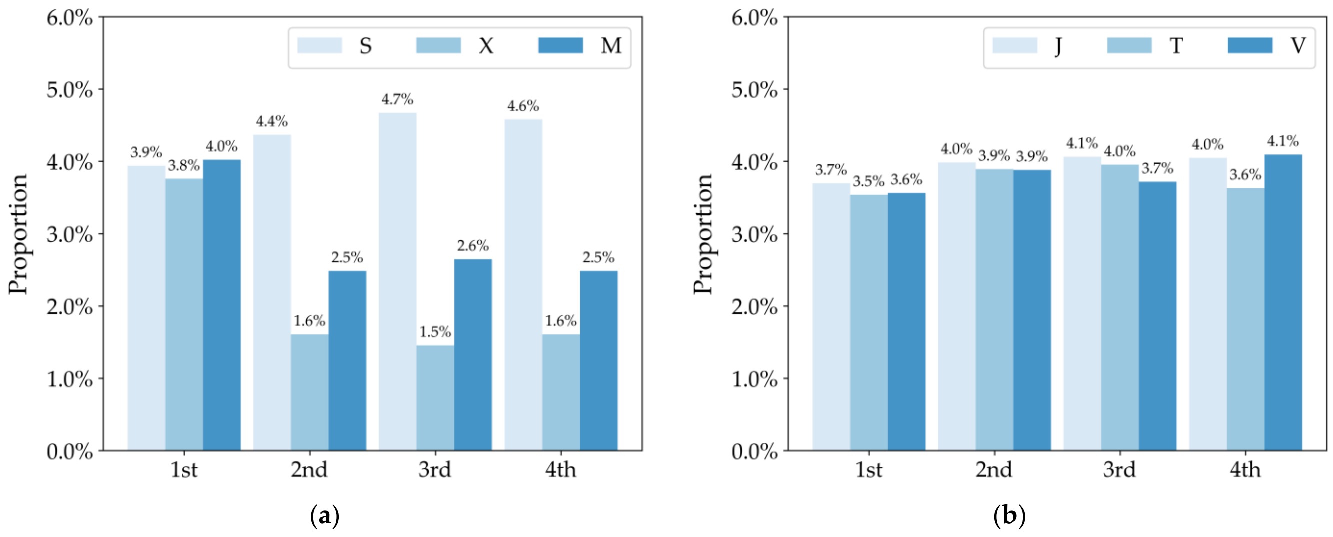


Figure 5. Character statistical distributions: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Further analysis found that the M–dataset also has MTPs 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.

Based on the above data analysis, it is necessary to model the inter-character correlation, so a CRNN module was adopted 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 of the images need to be normalized for better training.

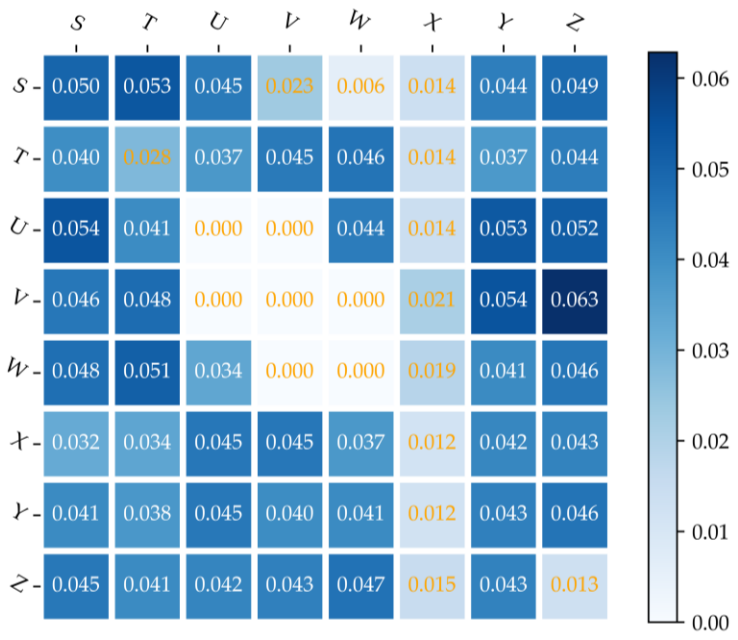


Figure 6. Markov transition probabilities between characters on the M-CAPTCHA.

3.2. AFFN Module

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 may help the subsequent CNN better extract the features of CAPTCHAs, thereby improving recognition accuracy. In this study, AFFN for combating anti-attack mechanisms presented in this research, as seen in Figure 7, comprises a sequence of nested autoencoder filter units. Both the encoder and decoder are composed of a convolutional layer, a Batch Normalization (BN) layer, and an activation layer.

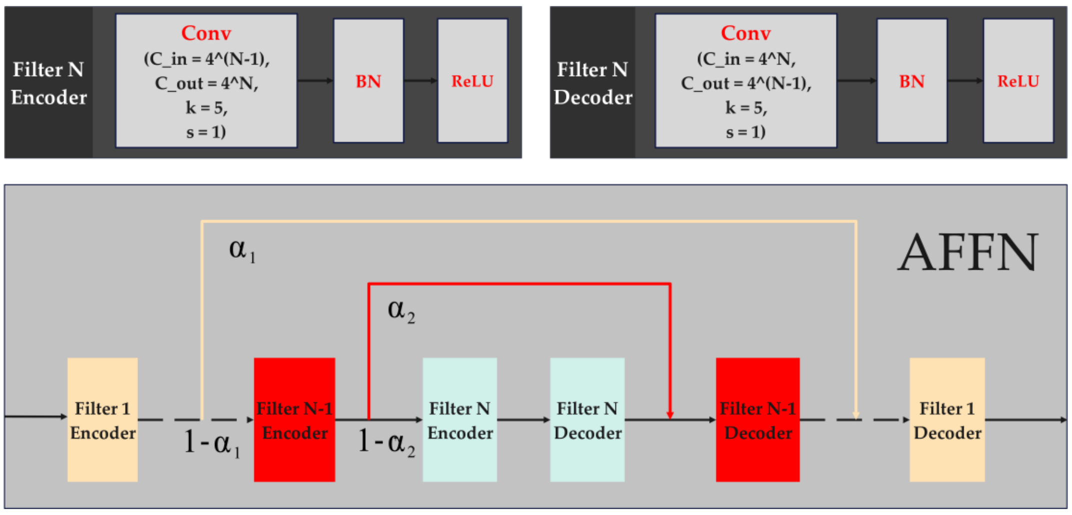


Figure 7. The structure of AFFN.

The number of layers depends on the noise power of the target dataset; the more noise, the larger the number of layers set, and vice versa. The experiments in the next section will show that a reasonable number of filter network layers is required to improve the AASR of the model. One approach is to manually set the number of filter units based on the noise level of the image, but this method requires experience and is not flexible. Another approach proposed in this study is to use adaptive filtering factor to adjust the weights of the filtering unit.

By introducing the fusion factor Alpha in AFFN, adaptive filtering of noise can be achieved. The adaptive fusion factor controls the weights of the next nested filtering unit participating in the filtering. Since this factor is trainable, the model can automatically adjust the filtering intensity based on the noise in the image. Each filtering unit in the filter bank adopts an encoder–decoder structure, and by using symmetric parameters, the shape of the input and output features can remain unchanged. The maximum number of filtering units *N* in the AFFN is a hyperparameter that needs to be set in advance.

Figure 8 shows the optimization process of the Alpha during training on the M–dataset and P–dataset when the maximum number of filtering units is two. The larger the Alpha value, the lower the filtering strength of the model, which also reflects the lower noise level of the data set. The Figure also shows that the P–dataset has a higher optimal Alpha than the M–dataset, which is consistent with the low noise characteristics of the P–dataset.

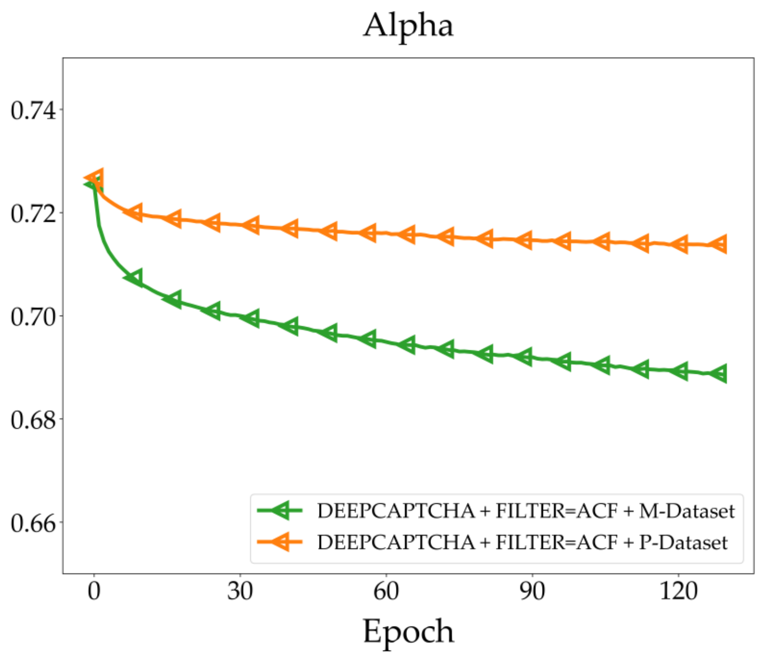


Figure 8. The training process of Alpha.

3.3. Residual Connections

Different layers output features of different sizes, which can be connected through residual connections to favor gradient backpropagation. However, when using residuals, too many connections may lead to performance degradation for models with few layers, such as the text CAPTCHA recognizer. Multiple residual connections can confuse training signals during backpropagation, affecting gradient optimization. Secondly, an excessive number of residual connections makes 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, we experimentally selected some residual connections from Pos T, Pos 0, Pos 1, Pos 2, and Pos 3, as shown in Figure 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.

3.4. CRNN Module

The CRNN module contains two convolutional layers, each followed by a BN layer and an activation function, as shown in Figure 9. These two convolutional layers are used to further down-sample 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 by flattening the channel, height, and width dimensions and dividing them evenly into four parts; each part 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. Based on testing experience, the probability of randomly discarding is set at 30%. This CRNN module is the key to model improvement, which not only greatly reduces the number of parameters but also improves the ASR.

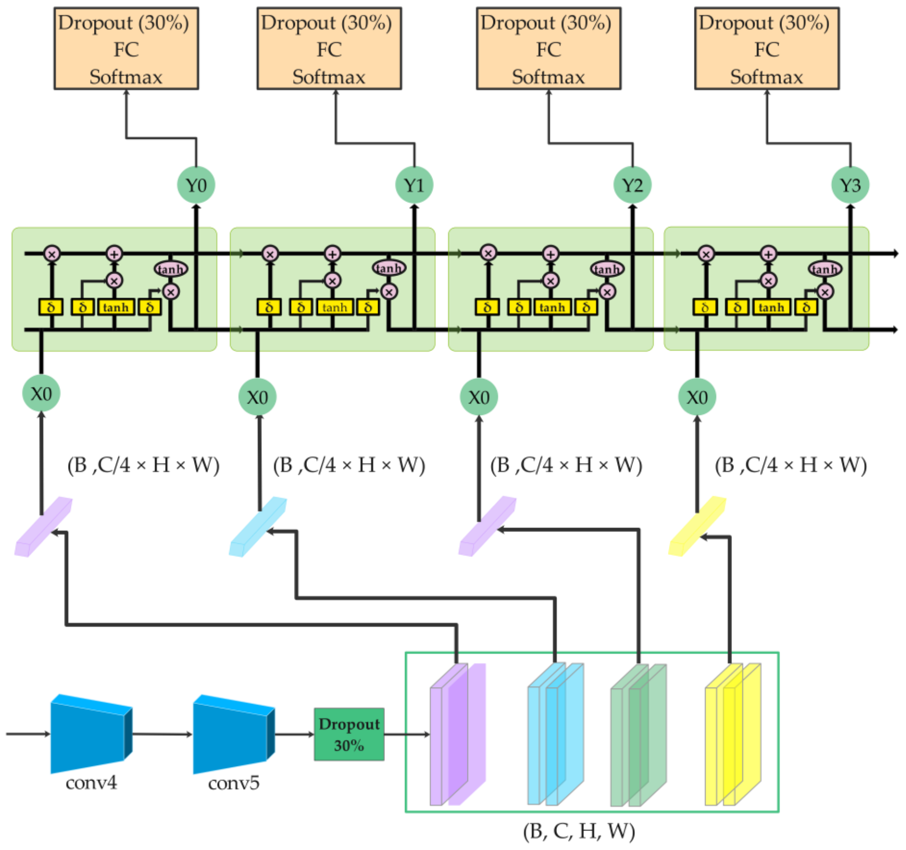


Figure 9. The structure of CRNN.

3.5. Metrics and Loss Functions

In addition to designing robust network models, the evaluation of model performance is critical. Among all evaluation metrics, the ASR and the 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, as shown in the following Equation (1). AASR is another index that is used to measure the overall recognition accuracy of CAPTCHAs regardless of the specific character position.

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

In the image classification, the CE and 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 (2) shows the weighted BCE formula for a multiclassification task, and w_i represents the weight. If w_i is equal to one, the above formula degenerates into a general BCE function.

$$L = -w_i[y_i \log x_i + (1 - y_i)\log(1 - x_i)] \tag{2}$$

Another loss function is focal loss, as shown in Equation (3). 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.

$$L = -\alpha y_i (1 - x_i)^{\gamma} \log x_i - (1 - \alpha)(1 - y_i) x_i^{\gamma} \log(1 - x_i) \tag{3}$$

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 aid 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 of 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. A total of 130 epochs were enough for the model's convergence, so this value was adopted as the training epoch. 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, manufactured at Chongqing, China, such as FPS.

4.1. Visual Analysis of Filter Networks

Figure 10 shows that the filter networks improved the recognition accuracy for all four positional characters on the M–dataset. There was an increase in ASR from around 85% to about 95%, with filter networks contributing approximately 10 percentage points on the 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 were affected by both sides. Therefore, the filter networks filtered out more noise and interference, resulting in a greater boost to ASR in the second and third positions.

As opposed to the M–dataset, where ASR improved significantly with the addition of the filter networks, there was no change in ASR in the P–dataset, and a slight decrease was found in the second character, as shown in Figure 11.

After undergoing the filtering process illustrated in Figure 12a, it is evident that the noise level in the M-CAPTCHA images has 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 12b. The design of filter networks is a double-edged sword that must be balanced between 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.

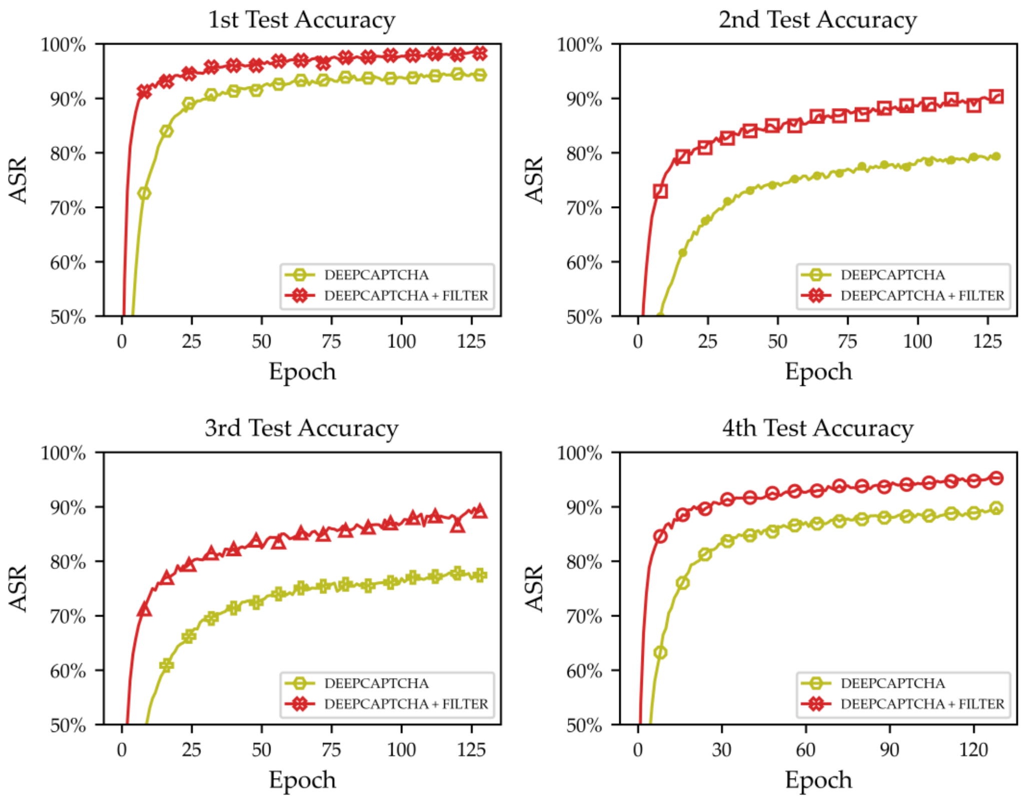


Figure 10. ASR with and without filter networks on the M-CAPTCHA.

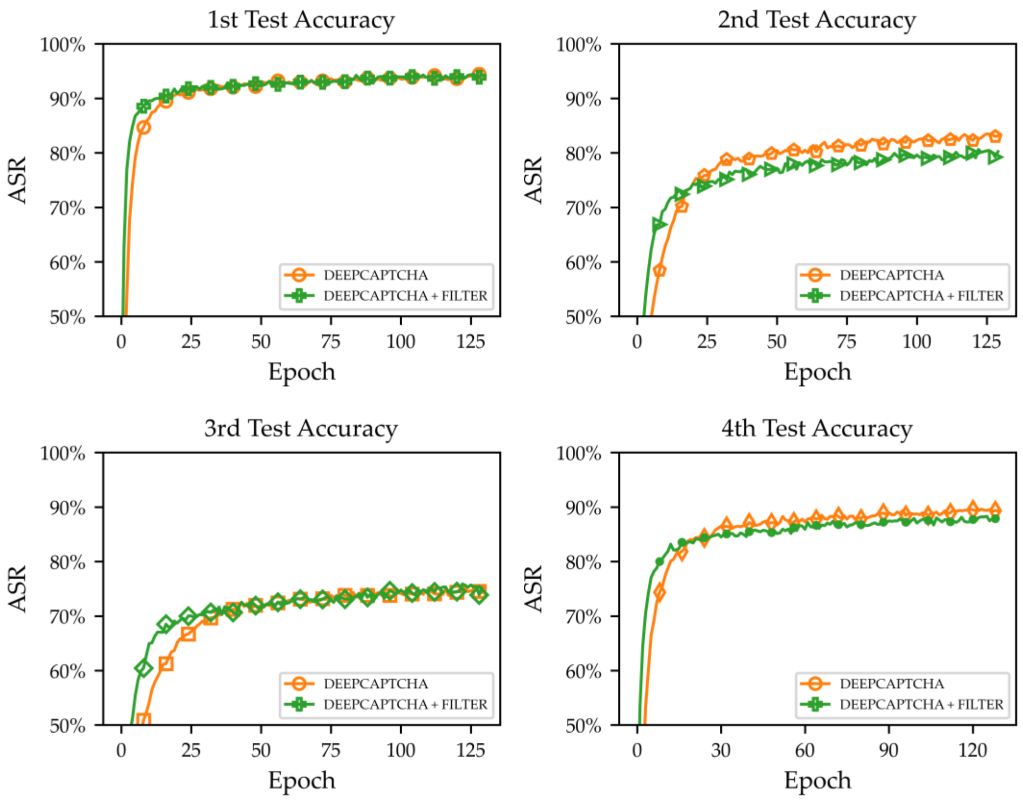


Figure 11. ASR with and without filter networks on the P-CAPTCHA.

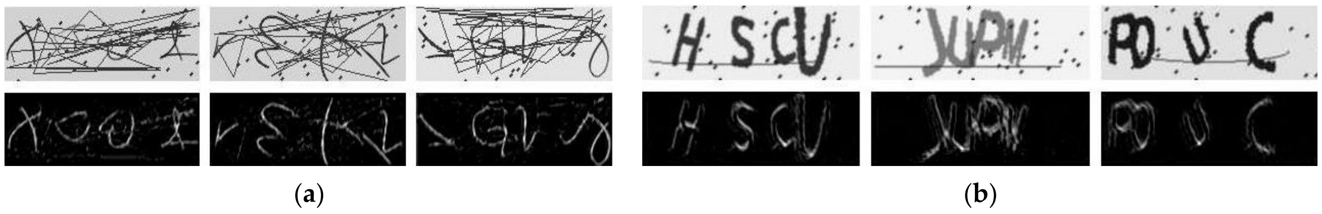


Figure 12. Comparison of images before and after filtering: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

The losses with and without the filter networks on the M-CAPTCHA and P-CAPTCHA are shown in Figure 13a,b, respectively. The curves in the two figures indicate that the convergence speeds increased after adding the filter networks, suggesting that the interference of the 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.

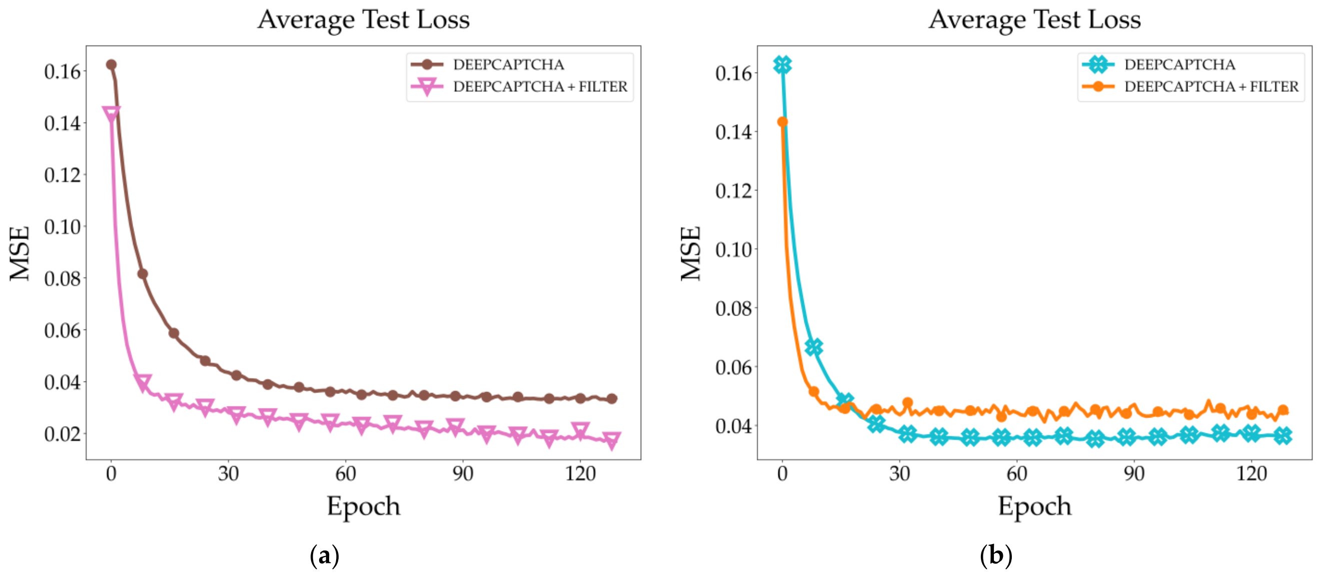


Figure 13. Loss comparison before and after filtering: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

Therefore, the number of filter units in Adaptive CAPTCHA should be proportional to the noise level. It is recommended to integrate more filter units for noisier CAPTCHAs and fewer for less noisy ones. Figure 14 shows the AASR when the filter network is configured with different units of layers, and the best AASR on the P–dataset was obtained when the filter network is AFFN. When configuring the Deep CAPCHA with two filtering units, the AASR is better than that with four filtering units. This is because the P–dataset is a low-noise CAPTCHA dataset, and strong filtering ability may damage the characters in the CAPTCHAs. It can also be seen that the AASR of the Deep CAPTCHA without any filtering units is the lowest. Therefore, by using the fusion factor Alpha, the model can find the optimal filtering unit weights, thereby achieving better performance.

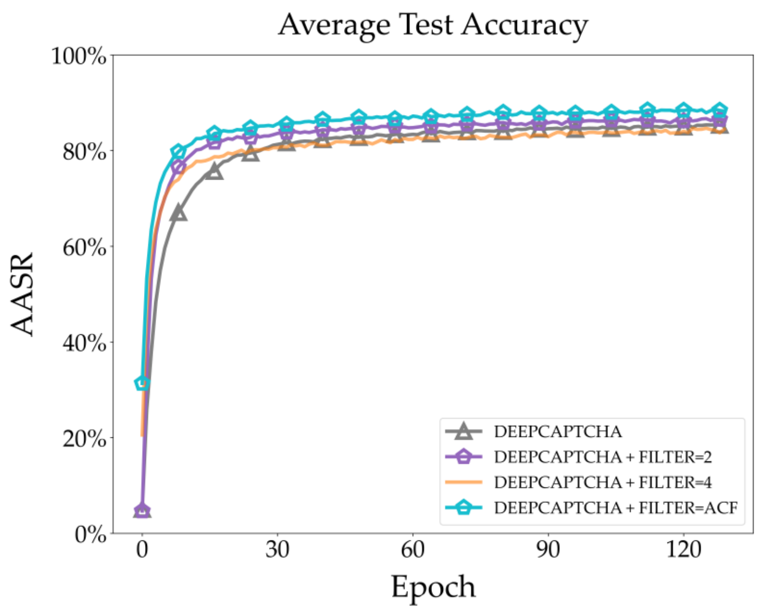


Figure 14. AASR with different filter units on the P–dataset.

4.2. Visual Analysis of CRNN

To compare under the same benchmark, this experiment did not use the filter networks but only replaced the FC layers of Deep CAPTCHA with CRNNs, 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 15a. The CRNN layers improved the AASR on the P–dataset from about 85% to almost 99%, as shown in Figure 15b. These two results indicated that the 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.

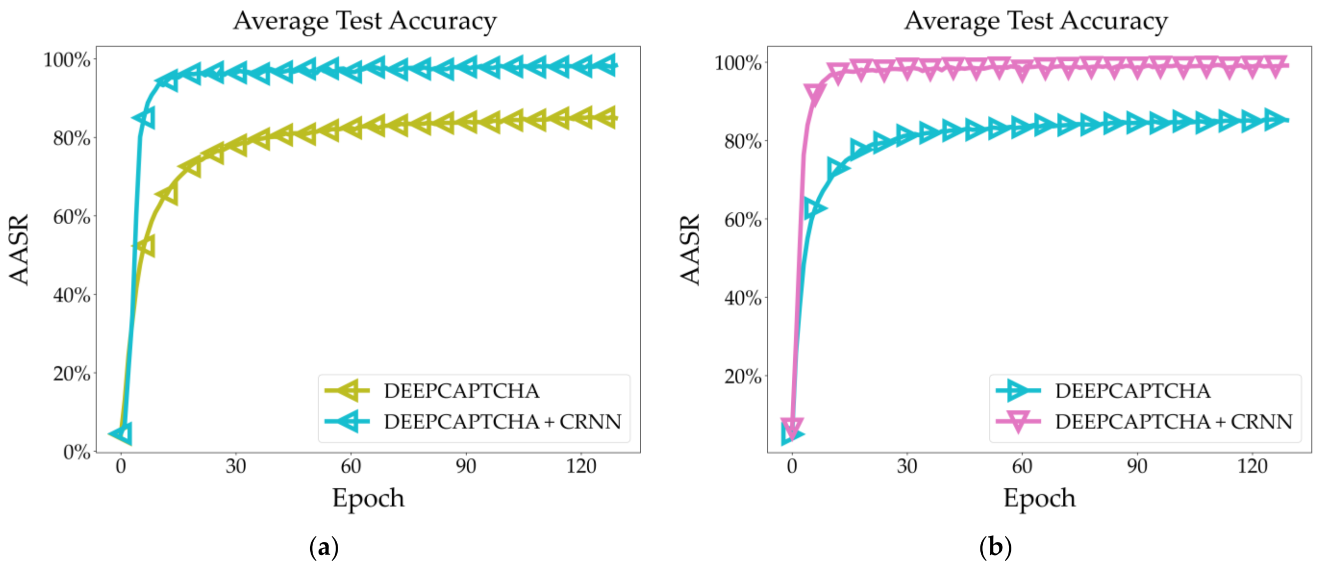


Figure 15. AASR comparison using FC and CRNN: (**a**) M-CAPTCHA; (**b**) P-CAPTCHA.

In the 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 seen from Figure 16 that, after using LSTM without BN layers, the AASR curve has a plateau area between about zero and five epoch intervals. Adding the BN layer speeds up gradient propagation and makes training more stable.

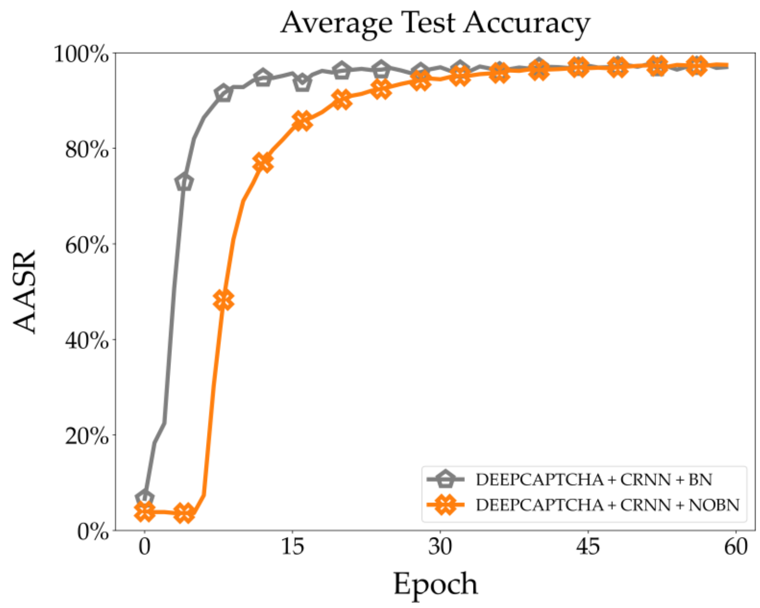


Figure 16. AASR comparison with and without BN in CRNN.

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 17. On the curve, one-layer LSTM even slightly outperformed two and three layers, which also had a minimum number of parameters.

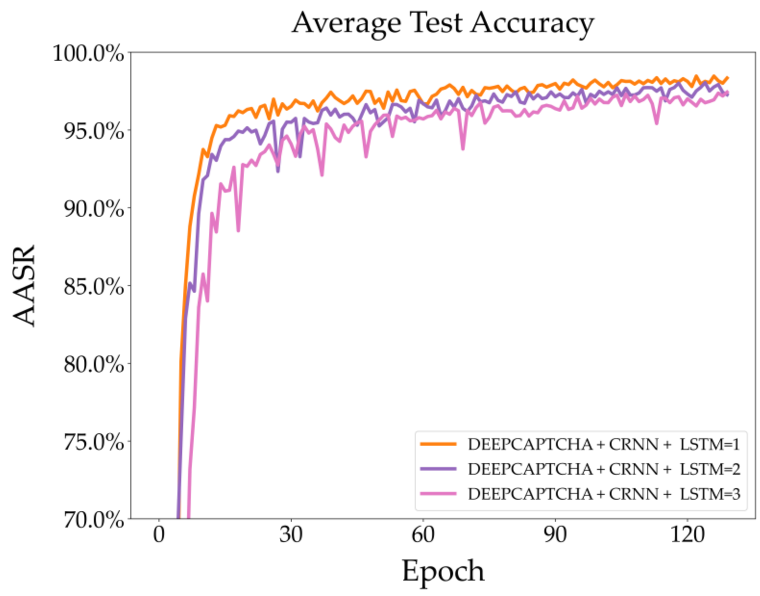
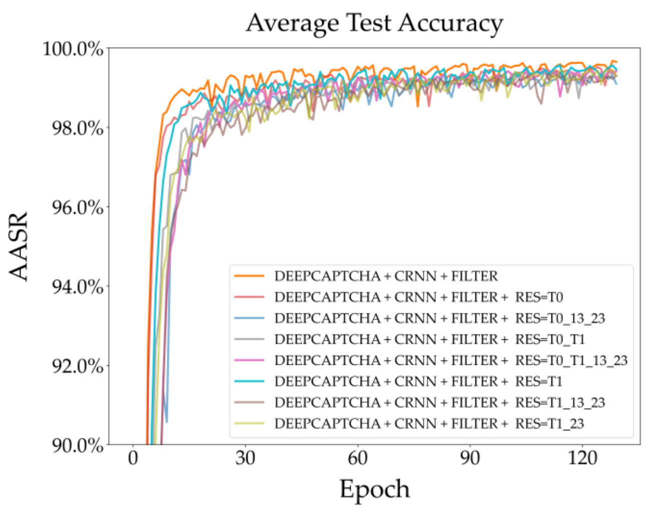


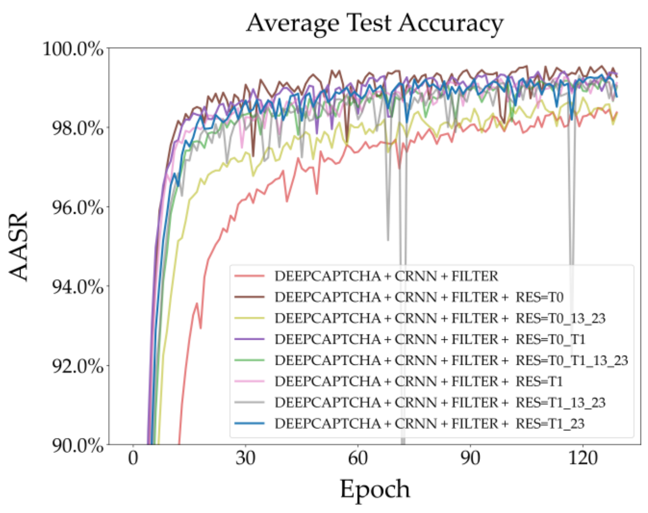
Figure 17. AASR with different layers of LSTM on P–dataset.

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 18a 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 while 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 18b.



(**a**) M-dataset



(**b**) P-dataset

Figure 18. AASR with different residual connections: (**a**) M-CAPTCHA (**b**) P-CAPTCHA.

4.4. Visual Analysis of Loss Functions

Figure 19 demonstrates AASR with different loss functions, and there were no significant performance discrepancies.

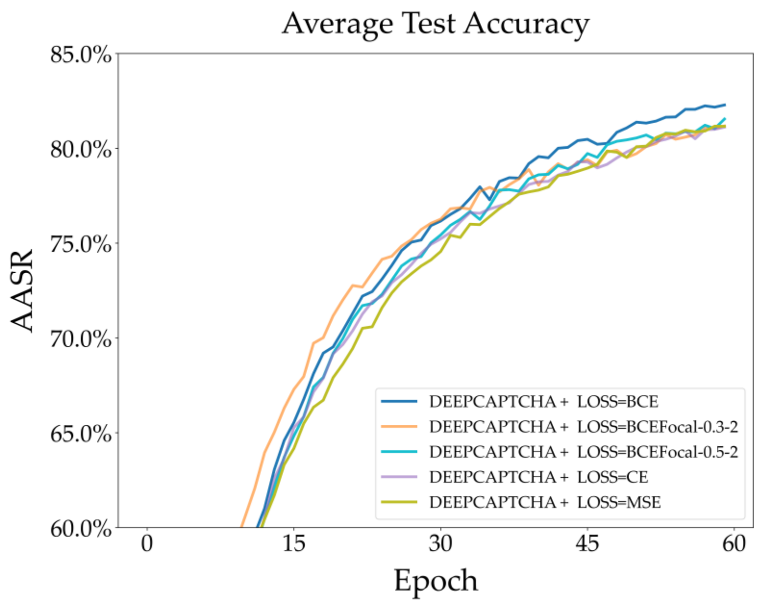


Figure 19. AASR with different loss functions.

The legend BCEFocal-alpha-gamma in the figure represents the weight of the positive sample loss term in BCE, while gamma represents the exponential weight of the predicted value. 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.

To better visualize the results of this study, a confusion matrix was employed. Figure 20 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.

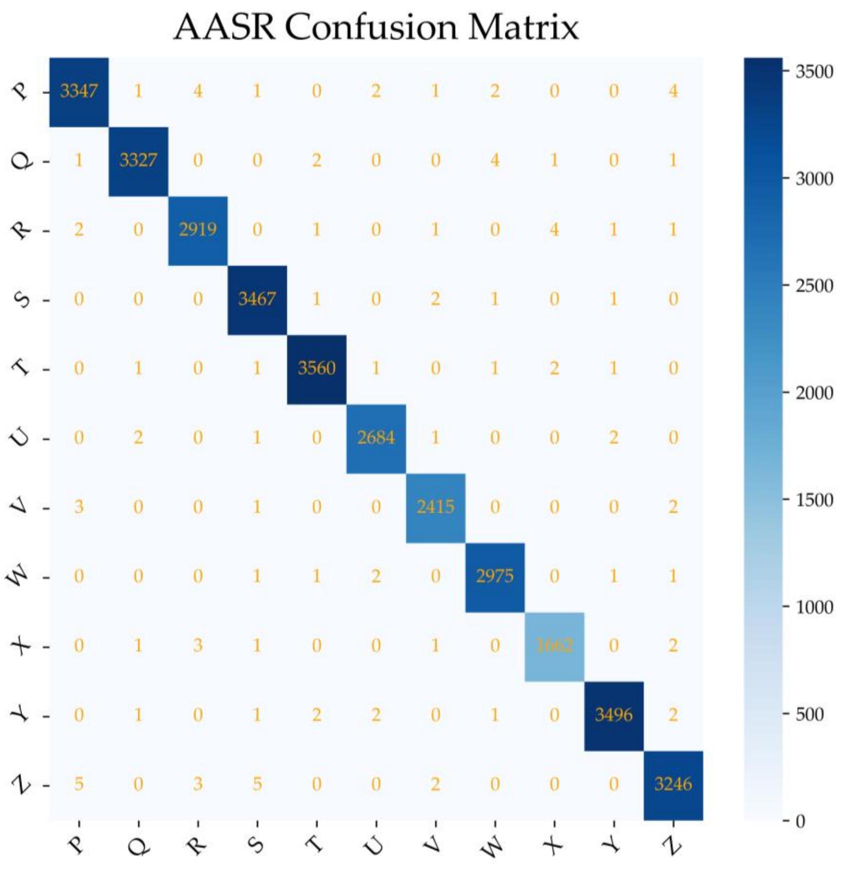


Figure 20. AASR confusion matrix of Adaptive CAPTCHA on M–dataset.

4.5. 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. R (T0, 13, 23) represents the model using three residual connections simultaneously, from position T to 0, from position 1 to 3, and from position 2 to 3, and others are also similar. 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 the CRNN was reduced by almost 41% compared to the PARAMs using FC layers, 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-noise P-CAPTCHA, the CRNN has a clear advantage in terms of accuracy and complexity.

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	193.1 M	6.46 M	85%	85%	120	120
A	Baseline + CRNN	109	276.1 M	3.82 M	98%	99%	110	110
B	A + AFFN	77	319.1 M	3.82 M	99%	98%	30	100
E	B + R (T0)	77	319.1 M	3.82 M	99%	99%	50	30
F	B + R (T0, 13, 23)	71	320.2 M	3.82 M	99%	99%	70	20
G	B + R (T0, T1)	67	319.6 M	3.82 M	99%	99%	70	70
H	B + R (T0, T1, 13, 23)	66	320.7 M	3.82 M	99%	99%	60	20
I	B + R (T1) (Adaptive CAPTCHA)	72	319.6 M	3.82 M	99%	99%	40	20
J	B + R (T1, 13, 23)	70	320.7 M	3.82 M	99%	98%	80	40
L	B + R (T1, 23)	70	320.3 M	3.82 M	99%	99%	70	80
M	O + AFFN	81	236.1 M	6.46 M	93%	84%	100	120
N	O + STN	67	226.2 M	6.53 M	62%	95%	120	60
O	O + AFFN + STN	60	269.2 M	6.53 M	89%	97%	110	60

Although the STN can theoretically be used to correct the geometric deformation of characters, the experimental results are not ideal when only promoting a little AASR on the M-CAPTCHA but increasing the number of parameters. The reason is that there are many irregular and non-linear deformations across a single CAPTCHA, making the STN difficult to calibrate. The performance improvement of the 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 the STN on simple datasets is still not as good as the 99% ASR of the CRNN, not to mention the fact that the STN is ineffective on complex datasets, and therefore the STN is not the best choice for text CAPTCHA breaking.

Adaptive CAPTCHA converges much faster than Deep CAPTCHA on both datasets. Among all possible residual connections in the 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 also shows that 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 the Adaptive CAPTCHA. The proposed model, Adaptive CAPTCHA, performed well in terms of AASR, PARAMs, and convergence speed while maintaining proper performance in FPS and MACs, which can meet the requirements of web crawlers and security vulnerability assessment.

4.6. Comparison with Other Works

In this section, we compare the performance of the proposed Adaptive CAPTCHA with four other models. The first model is the baseline model, Deep CAPTCHA, as mentioned before. The second is the Multiview Network (MultiviewNet) proposed by Mukhtar Opeyemi Yusuf et al., which applies two different filters to the same image and then sends them to two CNN paths for processing before making predictions. In this experiment, median filtering and bilateral filtering were used. The third and fourth models are ConvNet proposed by Ke Qing et al., and Capsule, proposed by Ionela Georgiana Mocanu et al.

It can be seen from Figure 21 that ConvNet and Adaptive CAPTCHA have the highest AASR exceeding 99%, but the convergence of ConvNet is faster under the same learning rate. The AASR of MultiviewNet and Capsule is between 90% and 95%, while Deep CAPTCHA is about 85%.

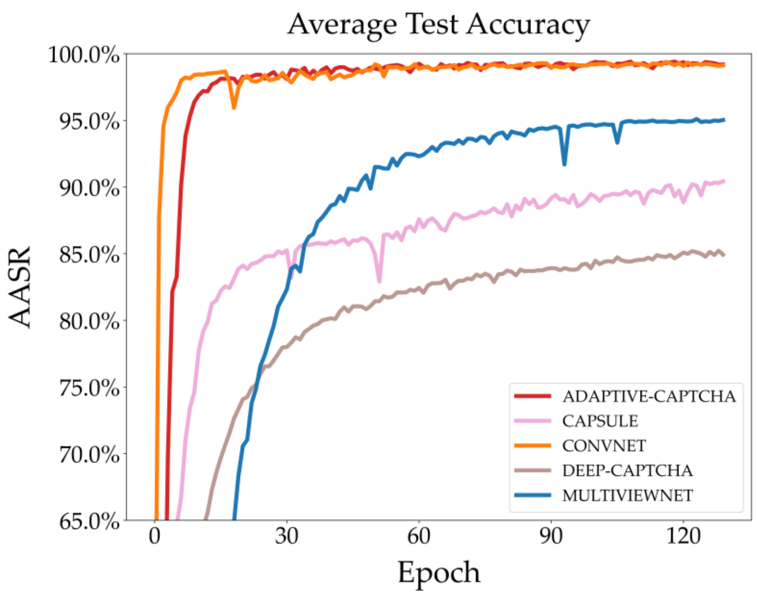


Figure 21. AASR of different models.

Table 2 lists other evaluation metrics of the different models. The chart clearly indicates that while ConvNet exhibits the highest AASR, its PARAMs and MACs are suboptimal, being almost four times more than those of Adaptive CAPTCHA. Besides, the FPS of ConvNet is the lowest, only around 22.5. It is worth noting that Capsule is the lightest framework with PARAMs of 691.26 K and has the lowest MACs among all models, but its inference speed is slower than Deep CAPTCHA and Adaptive CAPTCHA. The AASR of MultiviewNet is about 95% with small PARAMs, but the FPS is indeed significantly low. Adaptive CAPTCHA is in the first two on all indicators, which indicates that the proposed model does have comprehensive performance advantages.

Table 2. Comparison with other latest models.

No.	Models	FPS	MACs	Params	AASR
A	Deep CAPTCHA [4]	126	193.1 M	6.46 M	85%
B	MultiviewNet [25]	45.8	968.8 M	1.22 M	95%
C	Capsule [20]	59.4	77.77 M	691.26 K	90%
D	ConvNet [23]	22.5	1.47 G	15.23 M	99%
E	Adaptive CAPTCHA	72	319.6 M	3.82 M	99%

5. 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, an AFFN based on an encoder–decoder architecture was proposed, whose weights of filter units are learnable according to the noise level of the target CAPTCHA dataset. A notable reduction in the model's parameters 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 complexities. In addition, a series of ablation studies have been carried out to examine the impact of various components such as the STN, filter units, LSTM, and residual connections. We also 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.

Author Contributions: Conceptualization, X.W.; Methodology, J.J.; Software, X.W.; Validation, X.W.; Resources, X.W.; Data curation, X.W.; Writing—original draft, X.W.; Supervision, F.A.R.; Project administration, F.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [Leshan Vocational and Technical College] grant number [21GZD032], and the APC was funded by [Universiti Teknologi MARA].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. von Ahn, L.; Blum, M.; Langford, J. Telling humans and computers apart automatically. *Commun. ACM* **2004**, *47*, 56–60. [CrossRef]
2. von Ahn, L.; Blum, M.; Hopper, N.J.; Langford, J. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology—EUROCRYPT 2003*; Biham, E., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 294–311. [CrossRef]
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. [CrossRef]
4. Noury, Z.; Rezaei, M. Deep-CAPTCHA: A deep learning based CAPTCHA solver for vulnerability assessment. *arXiv* **2020**, arXiv:2006.08296. [CrossRef]
5. Baird, H.S.; Popat, K. Human Interactive Proofs and Document Image Analysis. In *Document Analysis Systems V*; Lopresti, D., Hu, J., Kashi, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; pp. 507–518. [CrossRef]
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), Antalya, Turkey, 22–25 March 2018; pp. 1–6.
7. Bostik, O.; Klecka, J. Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms. *IFAC-Paper* **2018**, *51*, 208–213. [CrossRef]
8. 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. [CrossRef]
9. Igbekele, E.O.; Adebiyi, A.A.; Ibikunle, F.A.; Adebiyi, M.O.; Oludayo, O.O. Research trends on CAPTCHA: A systematic literature. *Int. J. Electr. Comput. Eng. IJECE* **2021**, *11*, 4300–4312. [CrossRef]
10. 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), Coimbatore, India, 10–11 December 2020; pp. 1–6. [CrossRef]
11. 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 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 332–348. [CrossRef]
12. Zheng, Y. Captcha Recognition Based on Attention Mechanism. In Proceedings of the 6th International Conference on Control Engineering and Artificial Intelligence, Virtual Event, Japan, 11–13 March 2022; Association for Computing Machinery: New York, NY, USA, 2022; pp. 119–124. [CrossRef]
13. 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), Guangzhou, China, 28–30 October 2022; pp. 175–180. [CrossRef]
14. 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), Shanghai, China, 29 June–1 July 2023; pp. 171–175. [CrossRef]
15. 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), Columbus, OH, USA, 9–11 June 2003; pp. 106–109. [CrossRef]

16. Wang, J.; Qin, J.; Xiang, X.; Tan, Y.; Pan, N. CAPTCHA recognition based on deep convolutional neural network. *Math. Biosci. Eng.* **2019**, *16*, 5851–5861. [CrossRef] [PubMed]
17. Wang, Z.; Shi, P. CAPTCHA Recognition Method Based on CNN with Focal Loss. *Complexity* **2021**, *2021*, e6641329. [CrossRef]
18. 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. [CrossRef]
19. 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. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 2298–2304. [CrossRef]
20. Mocanu, I.G.; Yang, Z.; Belle, V. Breaking CAPTCHA with Capsule Networks. *Neural Netw.* **2022**, *154*, 246–254. [CrossRef] [PubMed]
21. Qing, K.; Zhang, R. Position-Encoding Convolutional Network to Solving Connected Text Captcha. *J. Artif. Intell. Soft Comput. Res.* **2021**, *12*, 121–133. [CrossRef]
22. Atri, A.; Bansal, A.; Khari, M.; Vimal, S. De-CAPTCHA: A novel DFS based approach to solve CAPTCHA schemes. *Comput. Electr. Eng.* **2022**, *97*, 107593. [CrossRef]
23. Qing, K.; Zhang, R. An Efficient ConvNet for Text-based CAPTCHA Recognition. In Proceedings of the 2022 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Penang, Malaysia, 22–25 November 2022; pp. 1–4. [CrossRef]
24. Bhowmick, R.S.; Indra, R.; Ganguli, I.; Paul, J.; Sil, J. Breaking CAPTCHA system with minimal exertion through deep learning: Real-time risk assessment on Indian government websites. *Digit. Threats Res. Pract.* **2023**, *4*, 1–24. [CrossRef]
25. Yusuf, M.O.; Srivastava, D.; Singh, D.; Rathor, V.S. Multiview deep learning-based attack to break text-CAPTCHAs. *Int. J. Mach. Learn. Cybern.* **2023**, *14*, 959–972. [CrossRef] [PubMed]
26. Derea, Z.; Zou, B.; Al-Shargabi, A.A.; Thobhani, A.; Abdussalam, A. Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy. *Sensors* **2023**, *23*, 9487. [CrossRef] [PubMed]
27. Sinha, S.; Surve, M.I. CAPTCHA Recognition And Analysis Using Custom Based CNN Model—Capsecure. In Proceedings of the 2023 International Conference on Emerging Techniques in Computational Intelligence (ICETCI), Hyderabad, India, 21–23 September 2023; pp. 244–250. [CrossRef]
28. Zou, Z.; Chen, K.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *Proc. IEEE* **2023**, *111*, 257–276. [CrossRef]
29. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; pp. 21–37. [CrossRef]
30. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
31. 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. [CrossRef]
32. 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. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023. [CrossRef]
33. Du, F.-L.; Li, J.-X.; Yang, Z.; Chen, P.; Wang, B.; Zhang, J. Captcha recognition based on faster R-CNN. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10362, pp. 597–605. [CrossRef]
34. 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. [CrossRef]
35. Carion, N.; Massa, F.; Synnaeve, G.; Usunier, N.; Kirillov, A.; Zagoruyko, S. End-to-End Object Detection with Transformers. *arXiv* **2020**, arXiv:2005.12872. [CrossRef]
36. Arjovsky, M.; Chintala, S.; Bottou, L. Wasserstein Generative Adversarial Networks. In Proceedings of the 34th International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 214–223.
37. 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*, 1–29. [CrossRef]
38. 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), Venice, Italy, 22–29 October 2017; pp. 2242–2251. [CrossRef]
39. 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. [CrossRef]
40. Chen, H.; Jiang, B.; Chen, H. StyleCAPTCHA: CAPTCHA Based on Stylized Images to Defend against Deep Networks. In Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference, Virtual, 19–20 October 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 161–170. [CrossRef]
41. Zhang, N.; Ebrahimi, M.; Li, W.; Chen, H. Counteracting Dark Web Text-Based CAPTCHA with Generative Adversarial Learning for Proactive Cyber Threat Intelligence. *ACM Trans. Manag. Inf. Syst.* **2022**, *13*, 1–21. [CrossRef]
42. Kimbrough, T.; Tian, P.; Liao, W.; Blasch, E.; Yu, W. Deep CAPTCHA Recognition Using Encapsulated Preprocessing and Heterogeneous Datasets. In Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2–5 May 2022; pp. 1–6. [CrossRef]
43. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. CBAM: Convolutional Block Attention Module. *arXiv*, 1807. [CrossRef]

44. Zi, Y.; Gao, H.; Cheng, Z.; Liu, Y. An End-to-End Attack on Text CAPTCHAs. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 753–766. [CrossRef]
45. 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, Chongqing, China, 28–30 May 2021; Association for Computing Machinery: New York, NY, USA, 2021; pp. 1–5. [CrossRef]
46. 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 4th International Conference on Graphics and Signal Processing, Nagoya, Japan, 26–29 June 2020; Association for Computing Machinery: New York, NY, USA, 2020; pp. 5–10. [CrossRef]
47. Zhao, R.; Deng, X.; Wang, Y.; Yan, Z.; Han, Z.; Chen, L.; Xue, Z.; Wang, Y. GeeSolver: A Generic, Efficient, and Effortless Solver with Self-Supervised Learning for Breaking Text Captchas. In Proceedings of the 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 21–25 May 2023; pp. 1649–1666. [CrossRef]
48. Li, Y.; Pan, H.; Ye, H.; Zheng, J. Transformer Encoder for Efficient CAPTCHA Recognize. In Proceedings of the 2023 2nd International Conference on Cloud Computing, Big Data Application and Software Engineering (CBASE), Chengdu China, 3–5 November 2023; pp. 355–361. [CrossRef]
49. Hallyal, R.A.; Sujatha, C.; Desai, P.; Meena, S.M. Optimized Recognition Of CAPTCHA Through Attention Models. In Proceedings of the 2023 IEEE 8th International Conference for Convergence in Technology (I2CT), Lonavla, India, 7–9 April 2023; pp. 1–7. [CrossRef]
50. 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.* **2023**, *55*, 1–38. [CrossRef]
51. Alsuhibany, S.A. Optimising CAPTCHA Generation. In Proceedings of the 2011 Sixth International Conference on Availability, Reliability and Security, Vienna, Austria, 22–26 August 2011; pp. 740–745. [CrossRef]
52. San Luo. m-CAPTCHA. Available online: https://www.kaggle.com/datasets/sanluo/mcaptcha (accessed on 5 February 2024).
53. 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), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]