electronics

MDPI

Article

CAPTCHA Recognition Using Deep Learning with Attached Binary Images

Alaa Thobhani [1], Mingsheng Gao [1,*], Ammar Hawbani [2], Safwan Taher Mohammed Ali [3] and Amr Abdussalam [4]

[1] College of Internet of Things (IoT) Engineering, Hohai University, Changzhou Campus, Changzhou 213022, China; althobhanialaa@gmail.com
[2] School of Computer Science and Technology, University of Science and Technology of China, Hefei 230026, China; anmande@ustc.edu.cn
[3] College of Communication Engineering, Jilin University, Changchun 130061, China; safwan.harbi@gmail.com
[4] Electronic Engineering and Information Science Department, University of Science and Technology of China, Hefei 230026, China; amr2010@mail.ustc.edu.cn
* Correspondence: gaoms@hhu.edu.cn

Received: 4 August 2020; Accepted: 9 September 2020; Published: 17 September 2020

Abstract: Websites can increase their security and prevent harmful Internet attacks by providing CAPTCHA verification for determining whether end-user is a human or a robot. Text-based CAPTCHA is the most common and designed to be easily recognized by humans and difficult to identify by machines or robots. However, with the dramatic advancements in deep learning, it becomes much easier to build convolutional neural network (CNN) models that can efficiently recognize text-based CAPTCHAs. In this study, we introduce an efficient CNN model that uses attached binary images to recognize CAPTCHAs. By making a specific number of copies of the input CAPTCHA image equal to the number of characters in that input CAPTCHA image and attaching distinct binary images to each copy, we build a new CNN model that can recognize CAPTCHAs effectively. The model has a simple structure and small storage size and does not require the segmentation of CAPTCHAs into individual characters. After training and testing the proposed CAPTCHA recognition CNN model, the achieved experimental results reveal the strength of the model in CAPTCHA character recognition.

Keywords: recognition; text-based CAPTCHA; convolutional neural network; deep learning; digital image processing

1. Introduction

The Completely Automated Public Turing test to tell Computers and Human Apart (CAPTCHA) [1–4] is a type of test to differentiate between humans and computer programs on Internet websites. CAPTCHA attempts to provide security against bots and can appear in many forms, including text, image, audio and video. Conducting research on recognizing CAPTCHA images is important because it helps identify weak points and loopholes in the generated CAPTCHAs and consequently leads to the avoidance of these loopholes in newly designed CAPTCHA-generating systems, thus boosting the security of the Internet.

Text-based CAPTCHAs are still a much popular and powerful tool against malicious computer program attacks due to their extensive usability and easy implementation. The majority of text-based CAPTCHAs consist of English uppercase letters (A to Z), English lowercase letters (a to z), and numerals (0 to 9). Other new large character sets, such as Chinese characters, have been used

recently in text-based CAPTCHAs [5,6]. Numerous mechanisms have been developed to secure and strengthen text-based CAPTCHAs including background noise, text distortion, rotating and warping, variable string length, and merging of characters. However, due to the rapid evolution of deep learning in the past few years, CAPTCHA recognition systems have become more competent than before in breaking most of the current defense mechanisms of text-based CAPTCHAs [7,8]. As a result, sophisticated security mechanisms need to be developed to make text-based CAPTCHAs more robust against malicious attacks.

Deep learning techniques demonstrate an excellent ability to extract meaningful features from input images and have numerous applications in various areas, such as image restoration [9–11] and object detection [12–14]. These powerful characteristics of deep learning techniques make them a good choice for building robust CAPTCHA recognition networks to perform attacks against text-based CAPTCHAs. Although several CAPTCHA recognition algorithms use traditional digital image processing techniques in their implementation, these techniques still suffer from drawbacks (e.g., weak feature extraction ability and easily influenced by noise in input images). As a result, these techniques are being gradually replaced by the powerful deep learning approaches.

Most of the currently available text-based CAPTCHA recognition algorithms can be classified into two main categories: segmentation-based and segmentation-free. Segmentation-based algorithms typically involve two main steps: segmentation [15] and character recognition [16]. In the segmentation step, the CAPTCHA image is segmented into individual characters, and these individual characters are recognized using character recognition module. The segmentation step is considered the most important part because it can considerably affect the overall accuracy and efficiency of the entire system. However, most segmentation algorithms do not perform well and suffer from low efficiency and efficacy. Thus, many researchers began to use segmentation-free CAPTCHA recognition algorithms to avoid the unfavorable consequences of the segmentation step.

Currently, segmentation-free models are widely used for recognizing CAPTCHAs. They can directly recognize and classify the characters of input CAPTCHAs without the need for segmenting CAPTCHAs into individual characters. In addition, most segmentation-free models demonstrate high accuracy and efficiency. However, deep-learning-based CAPTCHA recognition algorithms need large datasets to extract features efficiently. Moreover, segmentation-free models usually have complex architectures and require relatively large storage resources, especially when the number of characters in the text-based CAPTCHAs increases. Figure 1 presents the difference between segmentation-based and segmentation-free CAPTCHA recognition models.

In this study, we propose a new, simple and efficient deep-learning-based CAPTCHA breaking system with low complexity. Unlike segmentation-based models that suffer from the use of inefficient segmentation methods, the proposed model avoids the segmentation step by using the proposed attached binary images algorithm. First, we make several copies of the CAPTCHA image equal to the number of characters on CAPTCHA. Second, we create several external distinct binary images equal to the number of characters on CAPTCHA. Third, we attach the first distinct binary image to the first CAPTCHA copy, the second binary image to the second copy, and so on. Fourth, a label that represents the character class is added to each CAPTCHA copy. Lastly, the CAPTCHA copies with their labels are used to train a convolutional neural network (CNN) [17,18] model to classify CAPTCHA characters. The characters of the CAPTCHA input images can be directly and smoothly recognized and classified without segmentation by using the proposed attached binary images algorithm.

Unlike most of the current segmentation-free CAPTCHA recognition systems that suffer from complex architectures, our proposed model's architecture is simple, flexible, and uncomplicated. The proposed model consists of only one CNN with exactly one softmax layer in the output, and the number of neurons is equal to the character classes. The adoption of the attached binary images algorithm simplifies the architecture of our CAPTCHA-breaking CNN and makes it unaffected by the number of characters in the CAPTCHA image. When the number of characters in the CAPTCHA image increases, only the number of binary attached images and the number of CAPTCHA copies

will change while the architecture of the recognition CNN is unchanged. Furthermore, the simplified architecture of our model considerably reduces the model's storage size. The storage size of the model remains small, unchanged, and unaffected even when the number of characters on CAPTCHA is increased. This is not the case with other segmentation-free models whose internal architecture needs to be modified when the number of characters in the CAPTCHA image increases; in this case, the models also become increasingly complex, and their storage size increases accordingly.

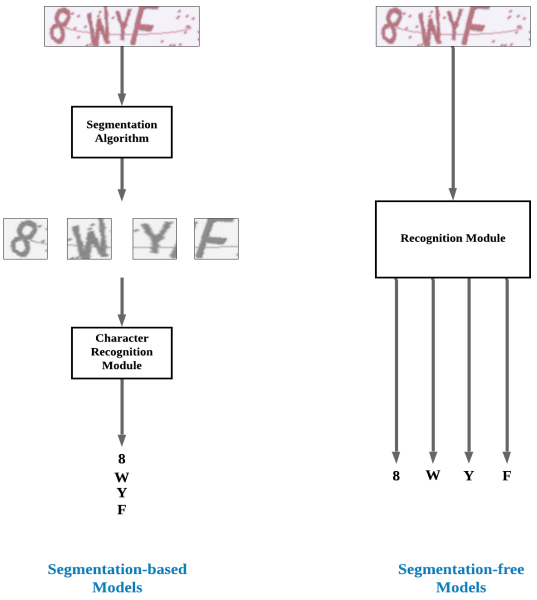

Figure 1. Description of the difference between segmentation-based and segmentation-free CAPTCHA recognition models.

We evaluate our proposed CAPTCHA recognition algorithm on two schemes of datasets, namely Weibo and Gregwar. Weibo CAPTCHAs are collected manually from the famous Chinese Weibo social media website while Gregwar CAPTCHAs are generated from the free and strong Gregwar CAPTCHA generating library. We analyzed the security of the two schemes. All targeted CAPTCHAs were broken with high success rates without using any segmentation. We achieved a success rate of 92.68% and 54.2% on Weibo and Gregwar CAPTCHA schemes, respectively. Moreover, the proposed algorithm could be useful for researchers on different fields that may have similar dataset structure.The main contributions of this paper include:

- We propose an Attached Binary Images (ABI) algorithm that is used to recognize CAPTCHA characters without the need for segmenting CAPTCHA into individual characters.
- With the adoption of ABI algorithm, we significantly reduce the storage size of our model and simplify the entire architecture of CNN model.
- We conduct our experiments on two CAPTCHA dataset schemes. The proposed model has efficiently improved the recognition accuracy and simplified the structure of the CAPTCHA recognition system as compared to other competitive CAPTCHA recognition methods.

The remainder of this paper is organized as follows. Section 2 introduces several CAPTCHA recognition and segmentation methods and algorithms. Section 3 presents the basic idea of the proposed CAPTCHA recognition algorithm and describes the structure and parameters of the recognition CNN. Section 4 shows the structure of the adopted datasets, evaluates the accuracy of proposed CAPTCHA recognition model, provides a comparison of the results, and presents a discussion of the proposed CAPTCHA recognition algorithm. The conclusion of the study is shown in Section 5.

2. Related Work

Segmentation-based CAPTCHA recognition systems are still widely used for CAPTCHA breaking purposes. The segmentation step is the main component in the recognition process of these segmentation-based models. Several algorithms have been proposed to segment text-based CAPTCHAs into separate characters. Zhang et al. [19] used the vertical projection technique [20–22] for CAPTCHA segmentation. They improved the vertical projection to deal with conglutination characters by combining the size features of characters and their locations with the vertical projection histogram. They also covered the segmentation of different types of conglutination. Chellapilla and Simard [23] used the connected component algorithm [24,25] to segment several CAPTCHA schemes, including Yahoo and Google, and achieved a success rate between 4.89% and 66.2%. However, vertical projection and connected component algorithms involve numerous preprocessing operations that are computationally expensive and time consuming. Hussain et al. [26] presented another CAPTCHA segmentation method in which the segmentation is based on recognition. First, an artificial neural network (ANN) was trained to recognize manually cropped CAPTCHA characters, then this trained ANN was used to segment a CAPTCHA image and crop its characters by using sliding windows. This segmentation method involves the application of the trained ANN to many extracted sub-windows to obtain their percentage of confidence, which could increase the segmentation time.

The character recognition module is also considered a crucial component of segmentation-based CAPTCHA recognition systems because it can influence the recognition accuracy of these systems. Sakkatos et al. [27] used the template matching [28,29] approach to recognize characters by comparing the separate characters with template characters using character's coefficient values. Errors arising from similarities between characters are considered a weak point in recognition via template matching unless more advanced solutions are incorporated. Chen et al. [30] introduced a CAPTCHA character recognition method called selective learning confusion class (SLCC). SLCC uses a two-stage Deep Convolutional Neural Network (DCNN) frame to recognize CAPTCHA characters. First, the characters are classified using the all-class DCNN. Then, a confusion relation matrix and a set partition algorithm are used to construct confusion class subsets. This CAPTCHA character recognition method has high character recognition accuracy, especially for confusion-class characters; however, assigning a new DCNN to each confusion class subset could considerably increase the storage size of the whole system.

To avoid the drawbacks of ineffective CAPTCHA segmentation algorithms, researchers have recently begun to adopt deep-learning-based segmentation-free CAPTCHA recognition systems for recognizing CAPTCHAs directly without segmentation. The authors of [31,32] used a segmentation-free CAPTCHA recognition CNN that is trained to recognize all CAPTCHA characters simultaneously. A specific number of neurons (equal to character classes) in the output layer was assigned to each CAPTCHA character for classification. This recognition model has fast recognition speed and avoids CAPTCHA segmentation. However, as the number of CAPTCHA characters increases, the number of neurons in the output layer also increases consequently, the storage size increases as well. Another segmentation-free multi-label CNN model was presented by Qing et al. [33] to recognize CAPTCHAs with connected and distorted characters.The internal structure of this CNN model was designed to consider the correlation between adjacent characters to improve recognition accuracy. However, this model uses a separate set of convolutional and fully connected layers for each character on CAPTCHA, which greatly complicates the architecture and increases the storage size of the model when the number of CAPTCHA characters is increased.

The authors in [34,35] used a segmentation-free model that combines CNN and attention-based recurrent neural network (RNN) to accomplish CAPTCHA recognition. The CNN part extracts features from a CAPTCHA image and produces feature vectors, and an Long Short-Term Memory (LSTM) network transforms the feature vectors into a text sequence. This model has high recognition speed and can be used to recognize CAPTCHAs of various lengths. However, the model's architecture is relatively complex because it consists of CNN and RNN parts, which could result in increased storage size.

3. Proposed Method

In this section, we describe the basic idea of our proposed CAPTCHA recognition algorithm. Also we explain the characteristics of the adopted attached binary images. Then, we present the internal structure of the CAPTCHA recognition CNN and its training parameters.

3.1. Basic Concept of Proposed Recognition Approach

For simplifiation, we use the term CRABI which stands for CAPTCHA Recognition with Attached Binary Images to refer to our proposed CAPTCHA recognition algorithm. The main idea behind the proposed method is to make several copies of the input CAPTCHA image and then attach external distinct binary images to these copies. We refer to the binary images as attached binary images (ABIs). We use the resultant CAPTCHA copies to train a CNN model to recognize CAPTCHA characters. We refer to this proposed CAPTCHA recognition CNN as "CRABI-CNN" throughout this paper. A description of the proposed CRABI algorithm is shown in Figure 2. We begin by explaining the recognition process during the training phase. The testing phase is then clarified.

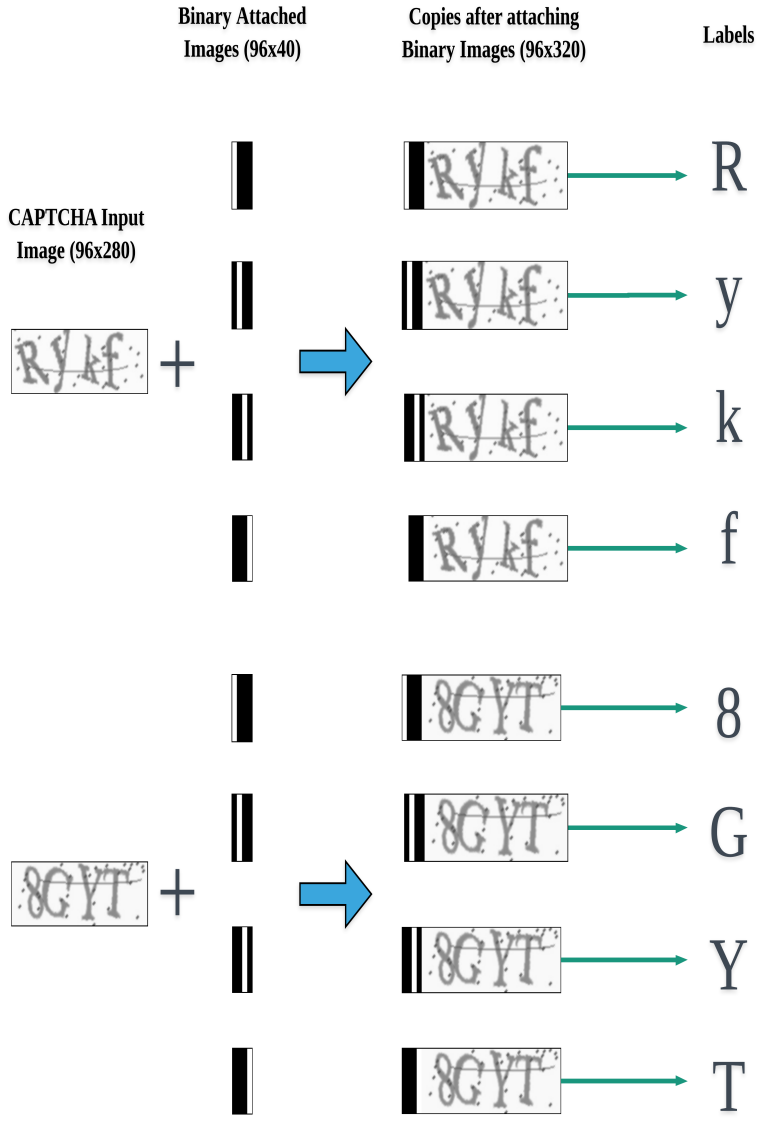

Figure 2. Description of main concept of the proposed method of CRABI algorithm.

3.1.1. Training Phase

Suppose that the original training set consists of M text-based CAPTCHA images, and the number of characters in each CAPTCHA image is n. The proposed algorithm is explained as follows:

1. Making Copies: We make n copies of each CAPTCHA image in the original training set. We end up with n identical copies of the training set.
2. Preparing ABIs: We have to define n external distinct fixed-size binary (black and white) images. These distinct binary images are used to represent the position or location information of the CAPTCHA characters. Each of the n distinct binary images is always responsible for locating exactly one character of the n CAPTCHA characters. That is, the first binary image will be always responsible for locating the first character in each CAPTCHA image, the second binary image is always responsible for locating the second character in each CAPTCHA image, and so on. The specific design of the binary images is presented in Section 3.2.
3. Attaching ABIs: We attach the distinct binary images to the CAPTCHA copies. The first distinct binary image is attached to the first copy of each CAPTCHA image in the training set, the second distinct binary image is attached to the second copy of each CAPTCHA image in the training set, and so on. We end up with a new training set consisting of $n \times M$ images. We refer to this new dataset as the "resultant dataset", and each image in this dataset is referred to as the "resultant CAPTCHA copy". Each resultant CAPTCHA copy consists of a CAPTCHA image and its ABI.
4. Labeling: We add labels to each resultant CAPTCHA copy in the resultant dataset. Every resultant CAPTCHA copy is given only one character class to be its label. Every resultant CAPTCHA copy consists of a CAPTCHA image and an ABI. The ABI of each resultant CAPTCHA copy determines the location of the character class on the CAPTCHA copy that is added as a label. In this way, labels can be added directly to all resultant CAPTCHA copies on the resultant dataset.
5. Training: The resultant CAPTCHA copies and their labels are used to train a CNN model to classify and recognize CAPTCHA characters. This CNN is trained to use attached binary images for locating CAPTCHA characters, and labels for recognizing character classes.

Figure 2 provides a full description of the proposed algorithm. The steps mentioned above can be summarized as follows: make n copies of the entire original training set; create n external distinct binary images; attach the first distinct binary image to all images of the first copy of the entire training set, attach the second distinct binary image to all images of the second copy of the entire training set, and so on. Attach labels to each resultant CAPTCHA copy; and use the resultant dataset to train a CNN network for classifying CAPTCHA characters. A complete framework description of the whole pipeline in training phase is shown in Figure 3.

3.1.2. Testing Phase

To test CAPTCHA images, we perform the following:

1. We make n copies of the CAPTCHA input image.
2. We attach each one of the n distinct binary images to one of the n copies of the CAPTCHA input image.
3. We submit the resultant n CAPTCHA copies directly to the trained CAPTCHA recognition CNN.
4. The CNN locates and classifies the characters of each resultant CAPTCHA copy, and the desired output is obtained. A complete framework description of the whole pipeline in testing phase is shown in Figure 4.

We use four-character CAPTCHA images of the shape 96×280 to implement our CAPTCHA recognition system. First, we make four copies of each CAPTCHA image. Second, we prepare four distinct binary images of the shape 96×40. Third, we attach the first binary image to the left of the first copy of each CAPTCHA image, the second binary image to the left of the second copy of each CAPTCHA image, and so on. The resultant CAPTCHA copies acquire the shape 96×320. Lastly,

labels are given to each resultant CAPTCHA copy. That is, the resultant CAPTCHA copy which its ABI is responsible for locating the first character is given its first character's class as its label, and the resultant CAPTCHA copy whose ABI is responsible for locating the second character is given its second character's class as its label. The same procedure is repeated for all resultant CAPTCHA copies.

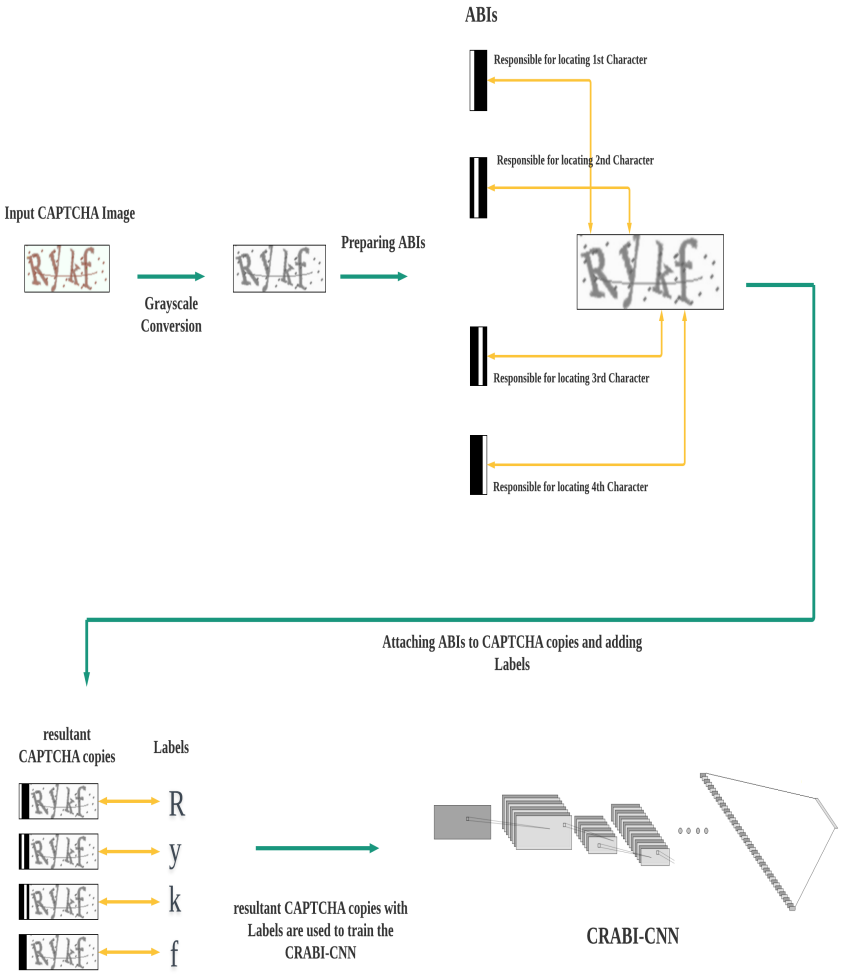

Figure 3. Framework description of the whole pipeline in training phase.

We now have four resultant CAPTCHA copies for each CAPTCHA input image. Each resultant CAPTCHA copy contains an ABI that represents the location (or position) of the CAPTCHA character and a label that represents the character's class. These resultant CAPTCHA copies are then used to train our CRABI-CNN model to recognize and classify label characters. In this way, we have avoided the need to segment CAPTCHA images into isolated characters and have simplified the entire structure of the CAPTCHA recognition CNN model.

Our CRABI-CNN, to some extent, is similar to the character recognition CNNs that are usually used in the character recognition part of segmentation-based CAPTCHA recognition systems. However, in character recognition CNNs, CAPTCHA input images are required to be segmented first; then, the separate characters are submitted individually to the character recognition CNN. By contrast, in our proposed CRABI algorithm, we do not need to segment CAPTCHA images because we replaced the segmentation step by making copies of CAPTCHA images and attaching binary images to these copies. The ABIs help in locating the characters to be recognized, and the resultant CAPTCHA copies are submitted individually to the CRABI-CNN for character recognition.

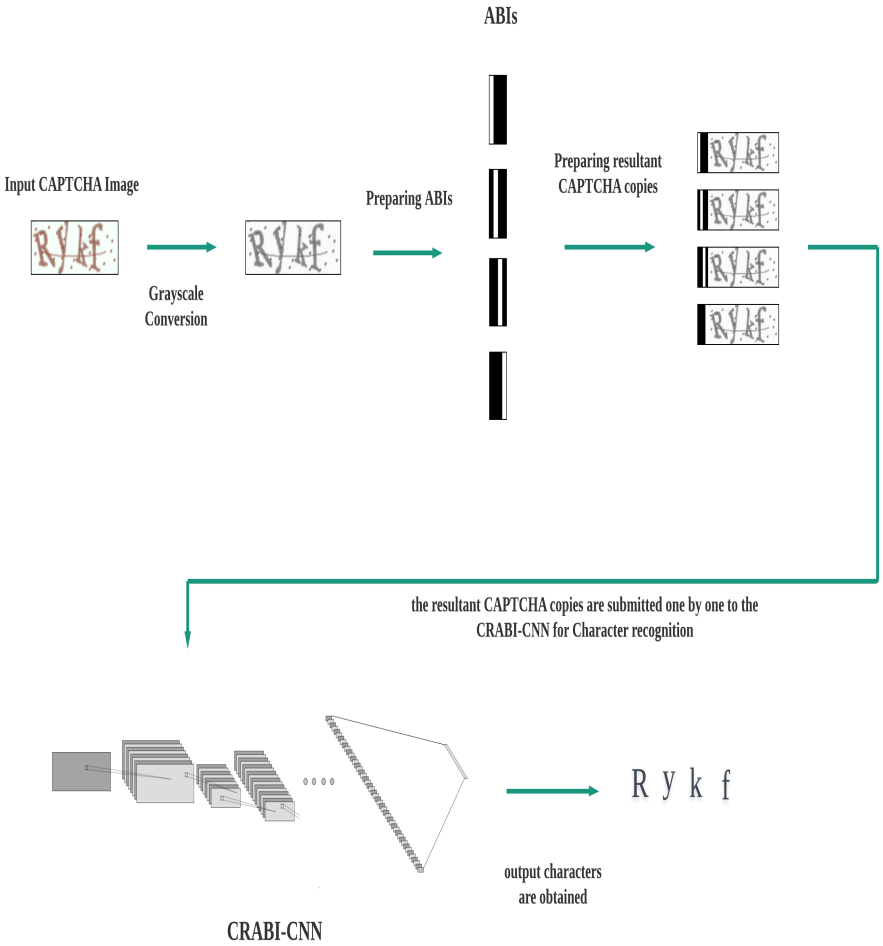

Figure 4. Framework description of the whole pipeline in testing phase.

3.2. *Characteristics of Attached Binary Images Adopted in Our Captcha Recognition Model*

Before starting to explain our adopted ABIs design, we need to mention that the design of ABI is not unique and can differ from researcher to another, and any researcher can use a design that is suitable for his or her application. For example, ABIs can just be binary images for numbers from 1 to n that distinguish the n character locations of a CAPTCHA image.

In our case, an ABI is a binary image (an image that consists only of black pixels with a value of 0 and white pixels with a value of 255) whose number of rows r is similar to the number of rows of the input CAPTCHA image, which is 96 in our case. The number of columns c of an ABI is given by the following formula:

$$c = n \times w \tag{1}$$

where n is the number of characters of the input CAPTCHA and w is the range of white columns in ABI. In our case, the number of characters n is equal to 4, and the range of white columns w is set to be 10. Thus, the number of columns c of an ABI is 40, and its shape is 96×40. Given that we have four characters in the input CAPTCHA image, we need to make four ABIs (one for each character). The characters in the CAPTCHA image are represented by $x \in$ {Numeral digits or English letter}, and each character is given an index $i \in \{1, 2, 3, 4\}$ representing its location (or order) in the CAPTCHA image, such that the first character of the CAPTCHA image is denoted by x_1, the second character is denoted by x_2, and so on. Each character x_i is attached to its corresponding ABI. The white and black regions of ABIs can be calculated with:

$$c_{B(i)} = (i-1) \times w + 1 \tag{2}$$

$$c_{F(i)} = i \times w \tag{3}$$

where $c_{B(i)}$ is the column's number at which the white columns range begins and $c_{F(i)}$ is the column's number at which the white columns range ends. For example, for the third CAPTCHA character x_3, $c_{B(3)}$ = 21 and $c_{F(3)}$ = 30, which means that columns 21 to 30 are white, whereas the remaining columns (from 1 to 20 and 31 to 40) are black. The ABIs are depicted in Figure 2.

3.3. Structure and Parameters of the Proposed CRABI-CNN

The feature extraction part of the proposed CRABI-CNN is based on the Model-5 architecture introduced in [36]. The CRABI-CNN architecture consists of 17 convolutional layers, 5 maxpooling layers, 1 flatten layer, 1 dropout layer, and 1 output softmax layer and their parameters are set to accelerate the training process and improve features extraction. A full description of CRABI-CNN architecture is shown in Table 1, and the entire architecture of the CRABI-CNN model is depicted in Figure 5. The output softmax layer contains several neurons equal to the number of character classes. For example, if a CAPTCHA scheme uses 62 character classes (10 numeral digits, 26 uppercase English letter and 26 lowercase English letter) to represent its characters, then the output softmax layer of our CRABI-CNN will contain only 62 neurons, such that each neuron corresponds to exactly one character class.

Table 1. Proposed CRABI-CNN Structure.

CRABI-CNN Layers
Convolutional (3 × 3), 32
Maxpooling (2 × 2), 2
Convolutional (3 × 3), 64
Maxpooling (2 × 2), 2
Convolutional (3 × 3), 128
Convolutional (1 × 1), 64
Convolutional (3 × 3), 128
Maxpooling (2 × 2), 2
Convolutional (3 × 3), 256
Convolutional (1 × 1), 128
Convolutional (3 × 3), 256
Convolutional (1 × 1), 128
Convolutional (3 × 3), 256
Maxpooling (2 × 2), 2
Convolutional (3 × 3), 512
Convolutional (1 × 1), 256
Convolutional (3 × 3), 512
Convolutional (1 × 1), 256
Convolutional (3 × 3), 512
Convolutional (1 × 1), 256
Convolutional (3 × 3), 512
Maxpooling (2 × 2), 2
Flatten
Dropout (0.5)
Softmax Layer

In the training of our proposed CAPTCHA recognition CNN, the cross-entropy loss function is used to measure differences between predicted and true classes. The Adam optimizer is adopted to optimize loss function. The learning rate is set to 0.00001 to provide improved optimization by experimentation. The training batch size is 128 images per batch, and the training process is for 120 epochs.

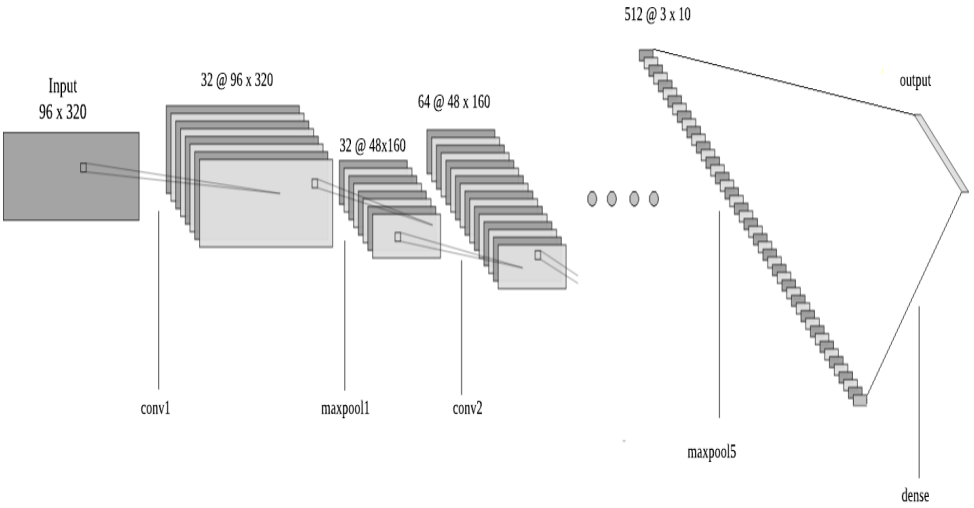

Figure 5. Architecture of proposed CRABI-CNN.

4. Experiments and Results

In this section, we explain two CAPTCHA scheme datasets used to train, validate, and test CRABI-CNN and clarify the labeling process. Next, the accuracy of the CRABI algorithm is calculated. Then, the results of comparing the CRABI algorithm with other CAPTCHA recognition systems are presented, and a discussion of the advantages and shortcomings of the CRABI algorithm is provided. Notably, we have trained, validated, and tested our CRABI model in the following environment: Floydhub cloud server, Tesla K80 GPU with 12 GB memory, 61 GB RAM, 100 GB SSD, Cuda v9.1.85, CuDNN 7.1.2, TensorFlow 1.9.0, and Keras 2.2.0 on Python 3.6.

4.1. Used Dataset and Labeling Description

Given that no publicly available standard datasets of CAPTCHA images can be used for CAPTCHA recognition purposes, we need to obtain CAPTCHA images either by collecting them from real online websites or by generating them using CAPTCHA generation software. To conduct our experiments, we adopt two CAPTCHA dataset schemes: Weibo (https://www.weibo.com/) and Gregwar (https://packagist.org/packages/gregwar/captcha). Figure 6 shows samples of the two CAPTCHA schemes.

Weibo CAPTCHA scheme samples

Gregwar CAPTCHA scheme samples

Figure 6. Samples of CAPTCHA images datasets.

4.1.1. Weibo Captcha Scheme

Weibo is one of the largest Chinese social media platforms. It is among the most popular websites globally as ranked by Alexa. In 2018, Weibo's monthly active users exceeded 400 million. The Weibo CAPTCHA scheme uses resistance mechanisms, including distortion, character overlapping, rotation and warping. Its CAPTCHAs contain four characters with character classes of either numeral digits or uppercase English letters. The excluded characters are 0, 1, 5, D, G, I, Q, U [7]. We manually collect and label 70,000 random Weibo CAPTCHA images as a dataset.

4.1.2. Gregwar Captcha Scheme

Gregwar is a free and open-source CAPTCHA generating library in PHP. It is among the strongest CAPTCHA schemes that show effective resistance against CAPTCHA breaking bots. It incorporates several security mechanisms, such as dense noise lines, color background, and rotation. We generate CAPTCHAs of four characters with character classes of either numerical digits, uppercase English letters, or lowercase English letters. We randomly generate 70,000 Gregwar CAPTCHA images as a dataset. All of the four characters in each generated CAPTCHA image are selected randomly, and we have verified that no repeated or duplicated CAPTCHA images are present.

Each CAPTCHA dataset scheme is divided into 50,000 CAPTCHA images as a training set, 10,000 CAPTCHA images as a testing set, and 10,000 CAPTCHA images as a validating set. Each CAPTCHA image in the two dataset schemes contains a label included in the image's name. This label consists of a four-characters text or string that represents the four characters found on this CAPTCHA. First, all CAPTCHA images in the two dataset schemes are converted into grayscale for simplification. Second, the images are reshaped into 96×280. Notably, the selection of the training, validating, and testing set images is performed randomly to avoid any subjective effects.

To train our CRABI-CNN, we make four copies of each CAPTCHA image and attach binary images of size 96×40 to each copy so that the size of the resultant CAPTCHA copies will become 96×320. Then, each resultant CAPTCHA copy is assigned a label consists of only one character.

Since we have made four resultant CAPTCHA copies of each original CAPTCHA image, the resultant CAPTCHA copy with the first ABI is assigned the first character of the four-character text of the original CAPTCHA image as a label, the resultant CAPTCHA copy with the second ABI will be assigned the second character of the 4-characters text of the original CAPTCHA image as a label, and so on. After finishing the labeling process, we end up with new schemes of datasets, each of which consists of 280,000 images of the resultant CAPTCHA copies (4 copies $\times$ 70,000 CAPTCHA images) with their corresponding character labels. These resultant dataset schemes are used individually to train, validate, and test our CRABI-CNN. The training set of each resultant dataset scheme consists of 200,000 images (4 copies $\times$ 50,000 CAPTCHA images), the testing set consists of 40,000 images (4 copies $\times$ 10,000 CAPTCHA images), and the validating set has 40,000 images (4 copies $\times$ 10,000 CAPTCHA images).

4.2. Accuracy and Training Description

We use the resultant Weibo scheme dataset to train CRABI-CNN for 120 epochs. The training time is 112,407,427 ms with an average epoch time of 936,729 ms and a batch size of 128. Then, we use the resultant Gregwar scheme dataset to separately retrain CRABI-CNN for 120 epochs with a total training time of 112,356,351 ms , an average epoch time of 936,303 ms , and a batch size of 128.

The accuracy of CRABI-CNN can be evaluated using two criteria: total character recognition accuracy and overall CAPTCHA image accuracy. For total character recognition accuracy, all characters of all CAPTCHA images are classified individually, and accuracy is calculated by dividing the number of correctly recognized characters by the total number of characters of all CAPTCHA images. For overall CAPTCHA image accuracy, all four characters of a single CAPTCHA image must be classified correctly for the CAPTCHA image to be recognized correctly; if one of the four characters of a single CAPTCHA image is wrongly classified, then the recognition result of this CAPTCHA image will be considered false. Afterward, the number of correctly recognized CAPTCHA images is calculated and divided by the total number of CAPTCHA images. Table 2 shows detailed results of the accuracies of the training, validating, and testing sets of Weibo and Gregwar datasets. Figure 7 shows the training and validating total character recognition accuracies of Weibo and Gregwar datasets over 120 epochs of training.

Table 2. Accuracies of individual characters and overall CAPTCHAs in CRABI-CNN.

	Weibo CAPTCHA Scheme			**Gregwar CAPTCHA Scheme**		
	Training Set	**Validating Set**	**Testing Set**	**Training Set**	**Validating Set**	**Testing Set**
1st Character Accuracy	99.75% (49,874/50,000)	98.71% (9871/10,000)	98.70% (9870/10,000)	99.84% (49,920/50,000)	93.56% (9356/10,000)	93.12% (9312/10,000)
2nd Character Accuracy	99.84% (49,919/50,000)	98.57% (9857/10,000)	98.35% (9835/10,000)	99.57% (49,784/50,000)	84.93% (8493/10,000)	85.28% (8528/10,000)
3rd Character Accuracy	99.26% (49,629/50,000)	96.01% (9601/10,000)	95.83% (9583/10,000)	98.15% (49,075/50,000)	74.20% (7420/10,000)	74.03% (7403/10,000)
4th Character Accuracy	99.37% (49684/50,000)	98.94% (9894/10,000)	98.68% (9868/10,000)	98.44% (49,221/50,000)	89.09% (8909/10,000)	88.68% (8868/10,000)
Total Character Accuracy	99.55% (199,106/20,0000)	98.06% (39,223/40,000)	97.89% (39,156/40,000)	99.00% (198,000/20,0000)	85.45% (34,178/40,000)	85.28% (34,111/40,000)
Overall CAPTCHA Accuracy	98.45% (49,226/50,000)	93.26% (93.26/10,000)	92.68% (9268/10,000)	96.26% (48,130/50,000)	54.30% (5430/10,000)	54.20% (5420/10,000)

After a training period of 120 epochs on the resultant Weibo dataset, we reach a testing total character recognition accuracy of 97.89%, such that 39,156 characters out of the 40,000 resultant testing set characters are recognized correctly. Also we reach an overall CAPTCHA testing accuracy of 92.68% with 9268 CAPTCHAs out of the 10,000 original testing set CAPTCHA images were correctly recognized. After a training period of 120 epochs on the resultant Gregwar dataset, we achieve a testing total character recognition accuracy of 85.28% with 34,111 characters out of the 40,000 resultant

testing set characters are recognized correctly. We also obtain an overall CAPTCHA testing accuracy of 54.20%, such that 5420 CAPTCHAs out of the 10,000 original testing set CAPTCHA images are correctly recognized.

Table 2 shows that the testing and validating accuracies of the Gregwar dataset are lower than those of the Weibo dataset. These results are expected because the defense mechanisms adopted by each CAPTCHA scheme differ. For instance, the Weibo CAPTCHA scheme contains only 28 character classes and always has a white background without any noise line passing through characters. Meanwhile, the Gergwar CAPTCHA scheme has 62 character classes and contains diverse foreground and background colors in addition to dense noise lines passing through characters. The strong defense mechanisms used by Gregwar CAPTCHA images greatly contribute to reducing the testing and validating accuracies of CRABI-CNN.

The total number of trainable and non-trainable parameters of the CRABI-CNN model is 6,670,812 for Weibo and 7,193,086 for Gregwar schemes. The number of trainable parameters is larger in Gregwar scheme CRABI-CNN because Gregwar scheme has more character classes and consequently more neurons in the output layer of CRABI-CNN. The size of CRABI-CNN weights on the hard disk is 25.5 MB and 27.5 MB for Weibo and Gregwar schemes, respectively.

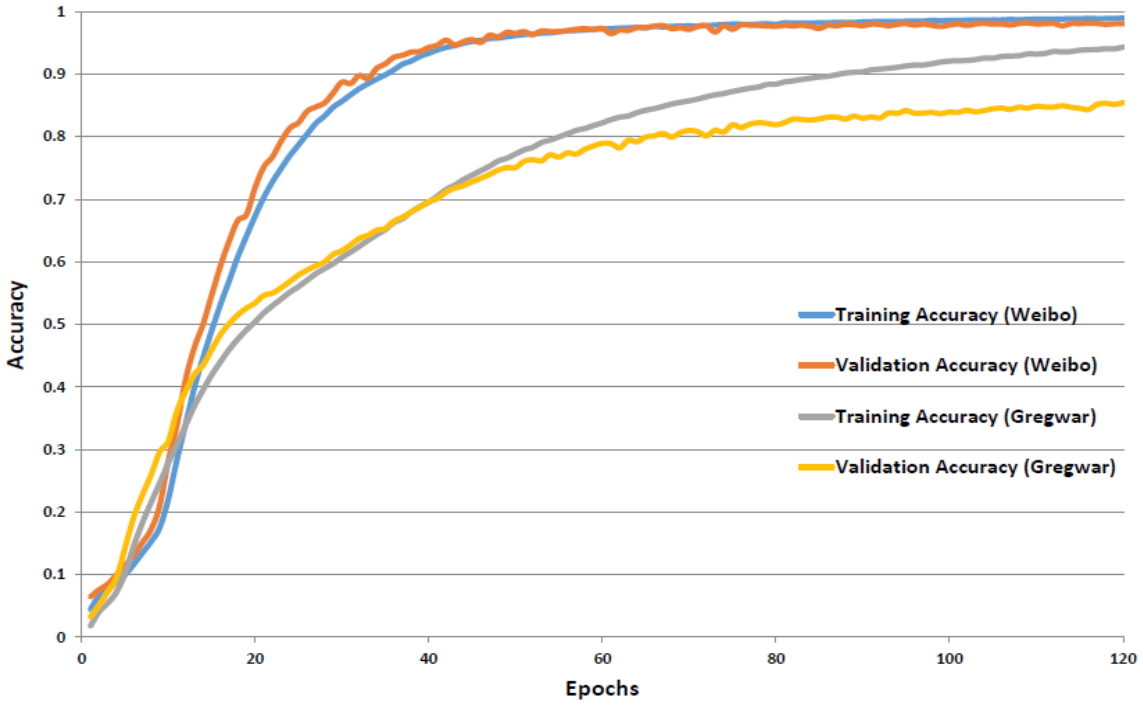

Figure 7. The training and validating total character recognition accuracies of CRABI-CNN on Weibo and Gregwar dataset schemes.

4.3. Comparison Results

To show the strong and weak points of the CRABI-CNN model, we compare it with other common CAPTCHA recognition algorithms on the same datasets. Two of the most currently popular models used for recognizing text and CAPTCHA images are the multilabel CNN and the Convolutional Recurrent Neural Network (CRNN).

The multilabel model that we use for comparison consists of exactly the same layers shown in Table 1, with four softmax layers on the output instead of only one softmax layer. Each of the four softmax layers is responsible for recognizing one corresponding character of the four-character CAPTCHA image. The structure of this multilabel-CNN model is similar to that of model-5 CNN

proposed in [36]. The Adam optimizer with a learning rate of 0.00001 is used to optimize the cross-entropy loss functions of this CNN.

We have trained this multilabel-CNN model on the Weibo and Gregwar dataset schemes separately. Each dataset scheme consists of 50,000 CAPTCHA images as a training set, 10,000 CAPTCHA images as a testing set, and 10,000 CAPTCHA images as a validation set. Before training, we convert CAPTCHA images into grayscale and reshape them to 96 × 320. We have trained the multilabel-CNN model with each dataset separately for 120 epochs. The training time is 28,813,055 ms with an average epoch time of about 240,109 ms on the Weibo dataset scheme, and 28,463,882 ms with an average epoch time of about 237,199 on the Gregwar dataset scheme.

The second model, used in our comparison is the CRNN model which is based on the models in [37,38]. In this CRNN model, we use eight convolutional layers, five maxpooling layers, two batch normalization layers, and two bidirectional gated recurrent unit (GRU) layers. All convolutional layers have a kernel size of 3 × 3, except for the last one that has a kernel size of 2 × 2, and the number of filters in these convolutional layers start from 64,128 until 512. All maxpooling layers are of size 2 × 2 and each of the bidirectional GRU layers has 128 units. The input CAPTCHA images are converted to grayscale and reshaped to 64 × 256, and the parameters of the convolutional and maxpooling layers are set to obtain 7 of 512-dimensional feature sequences which are forwarded to the GRU layers. The connectionist temporal classifier (CTC) [39] loss function is used to train this CRNN model.

The CRNN model is trained on the two schemes of datasets, namely Weibo and Gregwar. The training process on each dataset scheme lasts for 120 epochs. The training period is about 45,107,022 ms with an average epoch time of 375,892 ms on the Weibo dataset and about 24,276,033 ms with an average epoch time of 202,300 ms on the Gregwar dataset. Table 3 shows the results of comparing our CRABI-CNN model with the multilabel-CNN model and the CRNN model.

Table 3. Comparison Results.

	Weibo CAPTCHA Scheme			**Gregwar CAPTCHA Scheme**		
	CRABI CNN	**Multilabel CNN**	**CRNN**	**CRABI CNN**	**Multilabel CNN**	**CRNN**
Testing Total Character Accuracy	97.89% (39,156/40,000)	96.03% (38,411/40,000)	–	85.28% (34,111/40,000)	83.31% (33,322/40,000)	–
Testing Overall CAPTCHA Accuracy	92.68% (9268/10,000)	86.24% (8624/10,000)	91.05% (9105/10,000)	54.20% (5420/10,000)	51.23% (5123/10,000)	49.98% (4998/10,000)
Validating Total Character Accuracy	98.06% (39,223/40,000)	96.27% (38,506/40,000)	–	85.45% (34,178/40,000)	83.62% (33,448/40,000)	–
Validating Overall CAPTCHA Accuracy	93.26% (9326/10,000)	86.89% (8689/10,000)	91.09% (9109/10,000)	54.30% (5430/10,000)	51.63% (5163/10,000)	49.67% (4967/10,000)
Non-trainable and Trainable Parameters	6,670,812	7,961,136	10,477,853	7,193,086	10,050,232	10,486,591
Size of Weights on Hard Disk	25.5 MB	30.4 MB	40 MB	27.5 MB	38.4 MB	40.1 MB
Average Training Epoch Time	936,729 ms	240,109 ms	375,892 ms	936,303 ms	237,199 ms	202,300 ms
Testing Time on Entire Testing Set	68,170 ms	17,558 ms	16,125 ms	68,427 ms	19,055 ms	16,921 ms

The comparison results in Table 3 indicate that the testing total character recognition accuracy of our proposed CRABI-CNN model is better than that of the multilabel-CNN model in both CAPTCHA schemes. In addition, the testing overall CAPTCHA recognition accuracy of the CRABI model is the highest among the three models in Weibo and Gregwar dataset schemes. There is a difference of about 644 correctly recognized Weibo CAPTCHA images and about 297 correctly recognized Gregwar

CAPTCHA images between our CRABI model and the multilabel model, and a difference of about 163 correctly recognized Weibo CAPTCHA images and about 422 correctly recognized Gregwar CAPTCHA images between our CRABI model and the CRNN model. These results show the superiority of the testing accuracy of our CRABI model.

Moreover, Table 3 indicates that the number of trainable and non-trainable parameters and the size of weights of the CRABI-CNN model are much smaller than those of multilabel-CNN and CRNN models in both CAPTCHA schemes. This result is expected because the CRABI-CNN model contains only one softmax output layer with a limited number of neurons equal to the number of character classes adopted by a CAPTCHA scheme. However, the multilabel-CNN model contains four softmax output layers, such that each softmax layer contains several neurons equal to the number of character classes adopted by a CAPTCHA scheme. For example, if a four-character CAPTCHA scheme has 62 character classes (10 numeral digits, 26 English uppercase letters, and 26 English lowercase letters), then each softmax output layer in the multilabel-CNN model will contain 62 neurons, and the total number of neurons in the output layer will be $4 \times 62 = 248$ neurons. This increase in the number of neurons in the output layers of the multilabel model will increase the number of trainable and non-trainable parameters and the size of the multilabel model. Table 3 also shows that the CRNN model has the largest storage size among the three models because it includes the size of CNN and RNN layers.

Table 3 indicates that the average time of one training epoch of the CRABI model is longer than that of multilabel and CRNN models in both CAPTCHA schemes. The reason is that the training set used for the CRABI model is the resultant training set, which is four times the original training set since we have made four resultant CAPTCHA copies of each original CAPTCHA image in the original training set. This increase in the dataset caused the training time to increase by almost four times because the number of characters found in CAPTCHA images is four. Moreover, the testing time for the CRABI model is the longest among all models due to the same increase in the testing set of our CRABI model.

4.4. Discussion of the Proposed Crabi Algorithm

Any proposed algorithm or approach has its benefits and shortcomings. In this section, we show the advantages of our CRABI algorithm that make it a good choice for CAPTCHA recognition. We also discuss several disadvantages that must be considered.

4.4.1. Captcha Breaking Ability

Our proposed method achieved relatively high success rates for both the targeted schemes as shown in Table 2. CAPTCHA scheme can be considered broken when the success automated attack rate is 1% according to [40]. We have successfully broken several resistance mechanisms found on both CAPTCHA schemes which are commonly adopted by many popular CAPTCHA schemes including distortion, character overlapping, dense noise lines, rotation, warping, and color background. Also it can be noted from the results shown on Table 2 that the defense mechanisms adopted by Gregwar scheme is stronger than those of Weibo scheme. Gregwar CAPTCHA scheme incorporates strong security mechanism such as dense noise lines, diverse foreground and background colors, and wider range of character classes which makes it more difficult for even humans to recognize.

4.4.2. Avoiding Segmentation

One of the advantages of the proposed CRABI algorithms is the avoidance of segmenting CAPTCHA images into individual characters. By using ABIs and attaching them to CAPTCHA copies, CRABI-CNN can be trained to simultaneously locate and recognize characters without segmentation. Segmentation-based CAPTCHA recognition systems usually suffer from inefficient CAPTCHA segmentation techniques that could adversely affect their performance.

4.4.3. Small Storage

CRABI-CNN contains only one softmax layer in the output with the number of neurons equal to the character classes adopted by a CAPTCHA scheme. This feature means that even if the number of characters in the input CAPTCHA image is increased, only the number of ABIs and CAPTCHA copies will increase accordingly, and the CAPTCHA recognition CNN will still have only one output softmax layer with the same number of neurons. As a result, the number of trainable and non-trainable parameters, weights and storage size of CRABI-CNN are fixed and do not increase as the number of characters in CAPTCHA images increases. A different case applies to many other CAPTCHA recognition systems, such as multilabel-CNNs. Multilabel-CNNs contain multiple softmax output layers and many output neurons. The number of softmax output layers and neurons of multilabel-CNNs directly depends on the number of characters in the CAPTCHA images. Consequently, the number of trainable and non-trainable parameters, weights, and storage size of these CNNs will increase as the number of characters in the CAPTCHA images increases.

4.4.4. Simplicity and Flexibility

The structure of CRABI-CNN is (and keeps) simple, flexible, and unchangeable against an increasing or decreasing number of characters in the CAPTCHA image. The internal structure of CRABI-CNN does not need to be modified when the number of CAPTCHA characters is changed; modification is made only on the number of ABIs and CAPTCHA copies to be equal to the new number of characters. However, in the case of multilabel CAPTCHA recognition CNNs, the number of softmax output layers needs to be modified to be equal to the new number of characters in the CAPTCHA image; the number of neurons in the output layers needs to be updated accordingly. This task involves modifying the internal structure of the recognition CNN, thereby increasing its size and complicating its internal structure as the number of characters in the CAPTCHA image increases.

4.4.5. Long Training and Testing Time

The CRABI algorithm still suffers from some shortcomings that must be considered. In CRABI-CNN, the training set used for training is increased by a factor of n, where n is the number of characters in the CAPTCHA image, because we need to make n copies of each input CAPTCHA image in the original training set and attach binary images to each copy. This increase in the training set leads to an increase in the training time of each epoch of CRABI-CNN by approximately a factor of n, thus increasing the training time needed to reach the required accuracy. In addition, the testing set increases in the same manner as the training set. This increment, in turn, will increase the testing time needed by the CRABI model.

4.4.6. Memory Use

The increase in the original training set by a factor of n as required by our CRABI algorithm uses considerable memory (RAM) during the training phase. However, the capabilities of recent hardware resources (e.g., GPUs, RAMs, SSDs) used in deep learning applications have become increasingly powerful and robust, and the increase in the size of datasets is no longer a serious issue.

5. Conclusions

In this study, we present a new segmentation-free algorithm for recognizing CAPTCHA image. The algorithm is based on deep learning and uses ABIs with copies of the CAPTCHA images to locate and recognize the characters of CAPTCHA image. The adoption of ABIs in the proposed model decreases the overall size of the recognition system and reduces the complexity of the CNN model structure because the number of neurons in the output softmax layer is fixed and independent of the number of characters shown in the CAPTCHA image. Furthermore, the avoidance of the segmentation step by attaching ABIs, the adoption of a strong feature extraction CNN architecture, and the use of

ABIs for locating characters of CAPTCHA images significantly contribute in increasing the CAPTCHA characters recognition accuracy.

The proposed algorithm is evaluated on two schemes of datasets with about 70,000 CAPTCHA images in each scheme. The experimental results show that the proposed algorithm has a relatively higher CAPTCHA recognition accuracy than the state-of-the-art multilabel-CNN and CRNN models in both CAPTCHA scheme datasets. The storage size of the proposed model is also much smaller than that of both models. The proposed algorithm can be considered a new fundamental algorithm for recognizing CAPTCHAs in addition to the multilabel-CNN, CRNN, and segmentation-based CAPTCHA recognition models.

Author Contributions: Conceptualization, methodology, writing—original draft preparation, A.T.; software, validation, formal analysis, investigation, resources, data curation, visualization, A.T., A.A., S.T.M.A.; supervision, project administration, M.G.; writing—review and editing, A.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work was jointly supported by a project of the National Key Research and Development Program under grant 2018YFC0407101 and projects of National Natural Science Foundation of China under grants 61671202 and 61571063.

Acknowledgments: We would like to express our gratitude to Suad Abdullah and Ali Saleh for their support and advice. We also acknowledge with considerable appreciation the help and encouragement extended by Marwan Al-Srori, Majjed Alqatf, and Adnan Saifan.

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

References

1. Hussain, R.; Kumar, K.; Gao, H.; Khan, I. Recognition of merged characters in text based CAPTCHAs. In Proceedings of the 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016; pp. 3917–3921.
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, Warsaw, Poland, 4–8 May 2003; Biham, E., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; pp. 294–311.
3. von Ahn, L.; Blum, M.; Langford, J. Telling Humans and Computers Apart Automatically. *Commun. ACM* **2004**, *47*, 56–60. [CrossRef]
4. Gao, H.; Tang, M.; Liu, Y.; Zhang, P.; Liu, X. Research on the Security of Microsoft's Two-Layer Captcha. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*. [CrossRef]
5. Tang, M.; Gao, H.; Zhang, Y.; Liu, Y.; Zhang, P.; Wang, P. Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha. *IEEE Trans. Inf. Forensics Secur.* **2018**, *13*, 2522–2537. [CrossRef]
6. Wang, P.; Gao, H.; Rao, Q.; Luo, S.; Yuan, Z.; Shi, Z. A Security Analysis of Captchas with Large Character Sets. *IEEE Trans. Dependable Secur. Comput.* **2020**, *1*. [CrossRef]
7. 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.
8. Gao, H.; Wang, W.; Qi, J.; Wang, X.; Liu, X.; Yan, J. The Robustness of Hollow CAPTCHAs. In Proceedings of the Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; Association for Computing Machinery: New York, NY, USA, 2013; pp. 1075–1086.
9. Malik, S.; Soundararajan, R. Llrnet: A Multiscale Subband Learning Approach for Low Light Image Restoration. In Proceedings of the 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 22–25 September 2019; pp. 779–783.
10. Jin, Z.; Iqbal, M.Z.; Bobkov, D.; Zou, W.; Li, X.; Steinbach, E. A Flexible Deep CNN Framework for Image Restoration. *IEEE Trans. Multimedia* **2020**, *22*, 1055–1068. [CrossRef]
11. Dong, W.; Wang, P.; Yin, W.; Shi, G.; Wu, F.; Lu, X. Denoising Prior Driven Deep Neural Network for Image Restoration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2305–2318. [CrossRef] [PubMed]

12. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
13. 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] [PubMed]
14. Liu, Y. An Improved Faster R-CNN for Object Detection. In Proceedings of the 2018 11th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 8–9 December 2018; Volume 2, pp. 119–123.
15. Abdussalam, A.; Sun, S.; Fu, M.; Sun, H.; Khan, I. License Plate Segmentation Method Using Deep Learning Techniques. In *Proceedings of the Signal and Information Processing, Networking and Computers*; Sun, S., Ed.; Springer: Singapore, 2019; pp. 58–65.
16. Abdussalam, A.; Sun, S.; Fu, M.; Ullah, Y.; Ali, S. Robust Model for Chinese License Plate Character Recognition Using Deep Learning Techniques. In *CSPS 2018: Communications, Signal Processing, and Systems*; Liang, Q., Liu, X., Na, Z., Wang, W., Mu, J., Zhang, B., Eds.; Springer: Singapore, 2020; Volume 517, pp. 121–127.
17. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
18. Jmour, N.; Zayen, S.; Abdelkrim, A. Convolutional neural networks for image classification. In Proceedings of the 2018 International Conference on Advanced Systems and Electric Technologies (IC-ASET), Hammamet, Tunisia, 22–25 March 2018; pp. 397–402.
19. Zhang, L.; Xie, Y.; Luan, X.; He, J. Captcha automatic segmentation and recognition based on improved vertical projection. In Proceedings of the 2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN), Guangzhou, China, 6–8 May 2017; pp. 1167–1172.
20. Chen, C.J.; Wang, Y.W.; Fang, W.P. A Study on Captcha Recognition. In Proceedings of the 2014 Tenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, Kitakyushu, Japan, 27–29 August 2014; pp. 395–398.
21. Anagnostopoulos, C.E.; Anagnostopoulos, I.E.; Psoroulas, I.D.; Loumos, V.; Kayafas, E. License Plate Recognition From Still Images and Video Sequences: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2008**, *9*, 377–391. [CrossRef]
22. Wang, Q. License plate recognition via convolutional neural networks. In Proceedings of the 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 24–26 November 2017; pp. 926–929.
23. Chellapilla, K.; Simard, P.Y. Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In Proceedings of the 17th International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 13–18 December 2004; MIT Press: Cambridge, MA, USA, 2004; pp. 265–272.
24. Saleem, N.; Muazzam, H.; Tahir, H.M.; Farooq, U. Automatic license plate recognition using extracted features. In Proceedings of the 2016 4th International Symposium on Computational and Business Intelligence (ISCBI), Olten, Switzerland, 5–7 September 2016; pp. 221–225.
25. Sasi, A.; Sharma, S.; Cheeran, A.N. Automatic car number plate recognition. In Proceedings of the 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, India, 17–18 March 2017; pp. 1–6.
26. Hussain, R.; Gao, H.; Shaikh, R.A.; Soomro, S.P. Recognition based segmentation of connected characters in text based CAPTCHAs. In Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; pp. 673–676.
27. Sakkatos, P.; Theerayut, W.; Nuttapol, V.; Surapong, P. Analysis of text-based CAPTCHA images using Template Matching Correlation technique. In Proceedings of the 4th Joint International Conference on Information and Communication Technology, Electronic and Electrical Engineering (JICTEE), Chiang Rai, Thailand, 5–8 March 2014; pp. 1–5.
28. Wu, C.; On, L.C.; Weng, C.H.; Kuan, T.S.; Ng, K. A Macao license plate recognition system. In Proceedings of the 2005 International Conference on Machine Learning and Cybernetics, Guangzhou, China, 18–21 August 2005; Volume 7, pp. 4506–4510.
29. Baten, R.A.; Omair, Z.; Sikder, U. Bangla license plate reader for metropolitan cities of Bangladesh using template matching. In Proceedings of the 8th International Conference on Electrical and Computer Engineering, Dhaka, Bangladesh, 20–22 December 2014; pp. 776–779.

30. Chen, J.; Luo, X.; Liu, Y.; Wang, J.; Ma, Y. Selective Learning Confusion Class for Text-Based CAPTCHA Recognition. *IEEE Access* **2019**, *7*, 22246–22259. [CrossRef]
31. Hu, Y.; Chen, L.; Cheng, J. A CAPTCHA recognition technology based on deep learning. In Proceedings of the 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), Wuhan, China, 31 May–2 June 2018; pp. 617–620.
32. Stark, F.; Hazırbaş, C.; Triebel, R.; Cremers, D. CAPTCHA Recognition with Active Deep Learning. In Proceedings of the German Conference on Pattern Recognition Workshop, Aachen, Germany, 7–10 October 2015.
33. Qing, K.; Zhang, R. A Multi-Label Neural Network Approach to Solving Connected CAPTCHAs. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; Volume 1, pp. 1313–1317.
34. 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]
35. Wang, P.; Gao, H.; Shi, Z.; Yuan, Z.; Hu, J. Simple and Easy: Transfer Learning-Based Attacks to Text CAPTCHA. *IEEE Access* **2020**, *8*, 59044–59058. [CrossRef]
36. Fu, M.; Chen, N.; Hou, X.; Sun, H.; Abdussalam, A.; Sun, S. Real-Time Vehicle License Plate Recognition Using Deep Learning. In *ICSINC 2018: Signal and Information Processing, Networking and Computers*; Sun, S., Eds.; Springer: Singapore, 2019; Volume 494, pp. 35–41.
37. Sun, H.; Fu, M.; Abdussalam, A.; Huang, Z.; Sun, S.; Wang, W. License Plate Detection and Recognition Based on the YOLO Detector and CRNN-12. In *Proceedings of the Signal and Information Processing, Networking and Computers*; Sun, S., Ed.; Springer: Singapore, 2019; pp. 66–74.
38. 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.* **2017**, *39*, 2298–2304. [CrossRef] [PubMed]
39. Graves, A.; Fernández, S.; Gomez, F.; Schmidhuber, J. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd International Conference on Machine Learning, Corvallis, ON, USA, 25–29 June 2006; Volume 148, pp. 369–376. [CrossRef]
40. Bursztein, E.; Martin, M.; Mitchell, J. Text-Based CAPTCHA Strengths and Weaknesses. In Proceedings of the Proceedings of the 18th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 17–21 October 2011; Association for Computing Machinery: New York, NY, USA, 2011; pp. 125–138.