



Research Article

CAPTCHA Recognition Method Based on CNN with Focal Loss

Zhong Wang ^{1,2} and Peibei Shi ¹

¹School of Computer Science and Technology, Hefei Normal University, Hefei 230601, China

²Anhui Province Key Laboratory of Big Data Analysis and Application, University of Science and Technology of China, Hefei 230026, China

Correspondence should be addressed to Peibei Shi; pb_shi@163.com

Received 1 November 2020; Revised 13 December 2020; Accepted 21 December 2020; Published 4 January 2021

Academic Editor: M. Irfan Uddin

Copyright © 2021 Zhong Wang and Peibei Shi. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In order to distinguish between computers and humans, CAPTCHA is widely used in links such as website login and registration. The traditional CAPTCHA recognition method has poor recognition ability and robustness to different types of verification codes. For this reason, the paper proposes a CAPTCHA recognition method based on convolutional neural network with focal loss function. This method improves the traditional VGG network structure and introduces the focal loss function to generate a new CAPTCHA recognition model. First, we perform preprocessing such as grayscale, binarization, denoising, segmentation, and annotation and then use the Keras library to build a simple neural network model. In addition, we build a terminal end-to-end neural network model for recognition for complex CAPTCHA with high adhesion and more interference pixel. By testing the CNKI CAPTCHA, Zhengfang CAPTCHA, and randomly generated CAPTCHA, the experimental results show that the proposed method has a better recognition effect and robustness for three different datasets, and it has certain advantages compared with traditional deep learning methods. The recognition rate is 99%, 98.5%, and 97.84%, respectively.

1. Introduction

CAPTCHA is an algorithm for regional human behavior and machine behavior [1]. With the rapid development of Internet technology, network security issues continue to expand. CAPTCHA recognition is an effective way to maintain network security and prevent malicious attacks from computer programs, and it has been widely used in major mainstream websites [2]. CAPTCHA is generally considered to be a reverse turing test to classify humans and computers [3].

The mainstream CAPTCHA is based on visual representation, including images such as letters and text. Traditional CAPTCHA recognition [4–6] includes three steps: image preprocessing, character segmentation, and character recognition. Traditional methods have generalization capabilities and robustness for different types of CAPTCHA. The stickiness is poor. As a kind of deep neural network, convolutional neural network has shown excellent performance in the field of image recognition, and it is much better than traditional

machine learning methods. Compared with traditional methods, the main advantage of CNN lies in the convolutional layer in which the extracted image features have strong expressive ability, avoiding the problems of data preprocessing and artificial design features in traditional recognition technology. Although CNN has achieved certain results, the recognition effect of complex CAPTCHA is insufficient [7].

This paper introduces the focal loss function based on the CNN model to solve the problem of complex CAPTCHA recognition and improves the problems of the traditional convolutional neural network training such as the complexity of the model and the redundancy of the output layer parameters. The test results on three different datasets show the effectiveness of the proposed method. The rest of the content is arranged as follows: Section 2 introduces the related work, and Section 3 focuses on the based on convolutional neural network. In Section 4, the performance of the proposed method is verified by experiments. Finally, the summary and prospect are given.

2. Related Works

CAPTCHA mainly includes text CAPTCHA [8], image CAPTCHA [9], and sound CAPTCHA [10], among which text CAPTCHA is the most widely used. Text CAPTCHA is mainly composed of numbers and English letters, and its security is mainly guaranteed by two factors: background interference information and character adhesion. Both of these security features increase the difficulty of recognition and segmentation to varying degrees. According to whether characters need to be segmented in the recognition process, text CAPTCHA recognition methods can be divided into segmentation recognition and overall recognition. Segmentation recognition is a common method for CAPTCHA cracking. Chellapilla and Simard [11] prove that the effective segmentation of characters in CAPTCHA can greatly improve the recognition accuracy. In the early stage, CAPTCHA service website was the representative of CAPTCHA, which was characterized by little or no background interference information, and the characters were also lacking complex transformation such as distortion, rotation, and adhesion, and the defense effect was limited. Yan and Ahmad [12] completely cracked this kind of CAPTCHA by calculating pixels. Since then, the CAPTCHA designer has improved the generation algorithm and added background interference information, but Yan and El Ahmad [13] have used the projection algorithm to effectively segment it with an accuracy of up to 90%, and the success rate of cracking is up to 60%. After two consecutive rounds of attacks and defenses, in order to better resist the segmentation algorithm, the designer further improved the CAPTCHA, adding multiple complex transformations such as character twist, rotation, and adhesion and more complex transformation of background interference information [14, 15]. For this kind of CAPTCHA, Gao et al. [16] used Gabor filtering to extract character strokes and used graph search to find the optimal combination for character segmentation, and the accuracy of reCAPTCHA cracking reached 77%.

With the development of deep learning technology, CAPTCHA recognition technology based on deep learning is widely used. Qing and Zhang [17] proposed a multilabel convolutional neural network for text CAPTCHA recognition without segmentation and achieved better results for character distortion and complex CAPTCHA. Shi et al. [18] combined CNN with recurrent neural network and proposed a convolutional recurrent neural network to realize the overall recognition of CAPTCHA. Du et al. [19] used fast RCNN for overall recognition, which has a better recognition effect for CAPTCHA of variable length sequences. Lin et al. [20] used convolutional neural network to learn stroke and character features of CAPTCHA, greatly improving the recognition accuracy of CAPTCHA with distortion, rotation, and background noise. Compared with the traditional methods, deep neural networks have better learning ability and can effectively improve the efficiency of classification and recognition [21–23]. For example, AlexNet [24] further improves the CNN architecture and significantly improves the classification effect. It has been widely used to train CNN

on GPU. However, deep learning technology is currently limited in the face of severe AI image processing problems (such as symmetry [25] and adversarial example [26]). Most end-to-end recognition algorithms directly use the existing convolutional neural network structure, which has deep network layers and large training parameters. When the number of effective samples is limited, it is easy to overfit and lack generalization ability [27]. Therefore, how to design CAPTCHA for the defects of deep learning is the key problem to be solved.

3. The Proposed Method

3.1. Preprocessing. Traditional processing methods are used to preprocess the CAPTCHA image, including grayscale, binarization, image denoising, image segmentation, and image annotation. Firstly, the weighted average method is used to process the gray level, and the formula is $Y = 0.30 * R + 0.59 * G + 0.11 * B$, where R, G, and B correspond to the values of the red, green, and blue components in the color image. Then, the image binarization is carried out. The Otsu algorithm is used to obtain the optimal threshold value of each image. The pixels higher than the threshold value are set to 255, and the pixels below the threshold value are set to 0. Then, the average filter is used to denoise the image, and the formula $g(x, y) = (1/M) * \sum f(x, y)$ is used to set the current pixel value as the average value of eight neighboring pixels. Finally, the image is segmented, and the specific process is shown in Figure 1.

3.2. Focal Loss. Focal loss [28] is to solve the problem of low accuracy in one-stage target detection. This loss function reduces the weight of a large number of simple negative samples in training. It can also be considered as a difficult sample mining. Focal loss is modified on the basis of the cross-entropy loss function, which can reduce the weight of easy to classify samples to make the model focus more on difficult to classify samples during training.

For the two-category cross-entropy function, the formula is as follows:

$$CE(p, y) = \begin{cases} -\log(p), & \text{if } (y = 1), \\ -\log(1 - p), & \text{otherwise,} \end{cases} \quad (1)$$

where p is the estimated probability that the prediction sample belongs to 1 (the range is 0-1), y is the label, and the value of y is $\{+1, -1\}$. For the convenience of representation, the variable p_t is introduced. The formula is as follows:

$$p_t = \begin{cases} p, & \text{if } (y = 1), \\ 1 - p, & \text{otherwise.} \end{cases} \quad (2)$$

The cross entropy of the two categories can be expressed as $CE(p, y) = CE(p_t) = -\log(p_t)$. The common method to solve the class imbalance is to introduce the weight factor, which is $\alpha \in [0, 1]$ for category 1 and $1 - \alpha$ for category -1 .

$$\alpha_t = \begin{cases} \alpha, & \text{if } (y = 1), \\ 1 - \alpha, & \text{otherwise.} \end{cases} \quad (3)$$



FIGURE 1: Preprocessing diagram.

Then, the balanced cross entropy is $CE(p_t) = -\alpha_t \log(p_t)$. Although α can balance the importance of positive and negative samples, it cannot distinguish the difficult and easy samples. Therefore, focal loss reduces the weight of easy samples and focuses on the training of difficult negative samples. By introducing the parameter γ to represent the difficulty of the weight difference between the difficulty and easy samples, the greater the γ , the greater the difference, so the focal loss is defined as follows:

$$FL(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t). \quad (4)$$

Therefore, focal loss is a cross-entropy loss function with dynamically adjustable scale, which has two parameters α_t and γ , where α_t is to solve the imbalance between positive and negative samples and γ is to solve the imbalance of difficult and easy samples.

3.3. Simple CAPTCHA. For simple CAPTCHA, due to its small data image format and less information after image preprocessing, the model is relatively simple. The network structure is (as shown in Table 1 and Figure 2), repeated two layers of convolution combined with layer 1 pooling, followed by a layer of flatten layer and a layer of dense layer. Sigmoid function is the activation parameter of the full connection layer, and the label one-hot coding matrix with the maximum probability is transformed into the one-hot coding matrix. It is worth noting that each convolution layer uses the ReLU activation function, followed by a batch normalization batch standardization layer.

- (1) Input layer: the input data is single-channel image data after binarization with a size of 25×12 .
- (2) Convolutional layer C1 layer: using 8 convolution kernels which size is 3×3 , padding using the same convolution, filling the edge of the input image data matrix with a circle of 0 values, and the convolution operation step size is 1, each convolution kernel contains 9 parameters, and adds a bias parameter, so the required parameters for this layer are $(3 \times 3 + 1) \times 8 = 80$. The activation function is the ReLU function, followed by a batch normalization layer, and outputs $25 \times 12 \times 8$ feature maps.
- (3) Convolutional layer C2 layer: using 8 convolution kernels, but the size of the convolution kernel becomes $3 \times 3 \times 8$, padding still has the same convolution, the convolution step size is 1, the total parameter $(3 \times 3 \times 8 + 1) \times 8 = 584$, and the activation function is the ReLU function, followed by a batch normalization layer, and output $25 \times 12 \times 8$ feature maps.
- (4) Pooling layer P3 layer: we apply the maximum pooling algorithm to the output result of the C2 layer for pooling operation; this layer is also called the

downsampling layer, the downsampling window size used is 2×2 , and the output is $12 \times 6 \times 8$ feature maps.

- (5) Convolutional layer C4 layer: using 16 convolution kernels whose size is $3 \times 3 \times 8$, padding has the same convolution, convolution step length is 1, the total required parameters are $(3 \times 3 \times 8 + 1) \times 16 = 1168$, the activation function is the ReLU function, followed by a layer of batch normalization, and output $12 \times 6 \times 16$ feature maps.
- (6) Convolutional layer C5 layer: using 16 convolution kernels which size is $3 \times 3 \times 16$, padding is same convolution, convolution step length is 1, the total required parameters are $(3 \times 3 \times 16 + 1) \times 16 = 2320$, the activation function is the ReLU function, followed by a layer of batch normalization layer, and output $12 \times 6 \times 16$ feature maps.
- (7) Pooling layer P6 layer: we apply the maximum pooling algorithm to the output result of the C5 layer for pooling operation; the size of the downsampling window used is 2×2 and the output $6 \times 3 \times 16$ feature maps.
- (8) Flattening layer F7: we flatten the data of feature maps output by P6 layer, a total of $6 \times 3 \times 16 = 288$ nodes.
- (9) The Dense layer is fully connected with the F7 layer. The activation function is the focal loss function. All features are classified. The classification result corresponds to the character category of the CAPTCHA, including 10 numbers and 26 English uppercase characters, which means 36 possible results. A total of $36 \times (6 \times 3 \times 16 + 1) = 10404$ parameters are required.

3.4. Complex CAPTCHA. Complex CAPTCHA is mainly aimed at the image CAPTCHA which is difficult to be segmented because of its more adhesion, slanting font, complex color, and more disturbing pixels. It is widely used in major Internet websites. For complex CAPTCHA, the end-to-end neural network is used to identify the CAPTCHA. The model structure is shown in Figure 3. This kind of problem is multilabel classification. It is repeated five times, two convolution layers, one pooling layer, and then one flattened layer, and finally four classifiers are connected. Each classifier is fully connected, including 62 neural nodes. Sigmoid function is the activation parameter of the full convergence layer, that is, the probability of each classifier outputting a character, and the final output is complete one-hot encoding of 4 characters of image.

- (1) Input layer: the input data is RGB image data with the size of 27×72 and 3 channels.
- (2) Convolutional layer C1 layer: 32 convolution kernels of $3 \times 3 \times 3$ size are used, and the same content is used for padding; that is, a circle of 0 is filled into the edge of input image data matrix, and the convolution operation step is 1. Each convolution kernel contains 27 parameters and adds a bias parameter.

TABLE 1: Design of convolution neural network model.

Network structure	Type	Parameter	Connection	Step
Input	Data	$64*64*3$	—	—
Conv_1	Convolutional	$31*31*16$	Input	2
PReLU_1	Activating	—	Conv_1	—
Pool_1	Pooling	$16*16*16$	PReLU_1	2
Conv_2	Convolutional	$14*14*32$	Pool_1	1
PReLU_2	Activating	—	Conv_2	—
Pool_2	Pooling	$7*7*32$	PReLU_2	2
Conv_3	Convolutional	$5*5*64$	Pool_2	1
PReLU_3	Activating	—	Conv_3	—
Pool_3	Pooling	$3*3*64$	PReLU_3	2
Fc_1	Full connection	$1*1*128$	Pool_3	—
Cls_prob	Full connection	$1*1*2$	Fc_1	—
Loss function	Loss	—	—	—
Bbox_pred	Full connection	$1*1*4$	Fc_1	—
Loss function	Loss	—	—	—

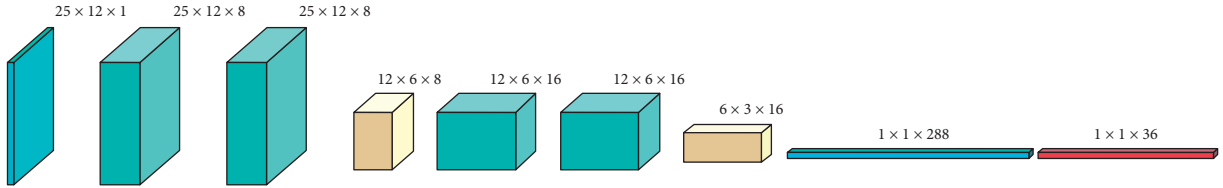


FIGURE 2: Network structure.

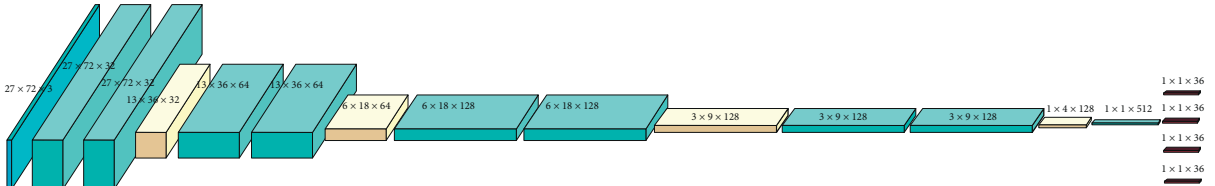


FIGURE 3: Network structure.

Therefore, the required parameters of this layer are $(3*3*3 + 1)*32 = 896$. The activation function is the ReLU function, which is next to the batch normalization layer and outputs $27*72*32$ feature maps.

- (3) Convolutional layer C2 layer: 32 convolution kernels are used, but the size of the convolution kernel becomes $3*3*32$, padding is still the same convolution, the convolution step is 1 and the total parameter is $(3*3*32 + 1)*32 = 9248$, and the activation function is the ReLU function, followed by a layer of batch normalization layer and output $27*72*32$ feature maps.
- (4) Pooling layer P3 layer: we use the maximum pooling algorithm for the output result of the C2 layer for pooling operation, and the downsampling

window size used is $2*2$ and output $13*36*32$ feature maps.

- (5) Convolutional layer C4 layer: we use 64 convolution kernels whose size is $3*3*32$, padding is the same convolution, convolution step is 1, the total required parameters are $(3*3*32 + 1)*64 = 18496$, and the activation function is the ReLU function, followed by a layer of batch normalization layer and output $13*36*64$ feature maps.
- (6) Convolutional layer C5: we use $3*3*64$ convolution kernels whose size is 64, padding is the same convolution, convolution step is 1, the total required parameters are $(3*3*64 + 1)*64 = 36928$, and the activation function is the ReLU function, followed by a layer of batch normalization layer and output $13*36*64$ feature maps.

- (7) Pooling layer P6 layer: the output result of C5 layer is used for the maximum pooling algorithm for pooling operation, and the size of the down-sampling window used is 2×2 and output $6 \times 18 \times 64$ feature maps.
- (8) Convolutional layer C7 layer: we use 128 convolution kernels whose size is $3 \times 3 \times 64$, padding is the same convolution, convolution step size is 1, the total required parameters are $(3 \times 3 \times 64 + 1) \times 128 = 73856$, and the activation function is the ReLU function, followed by a layer of batch normalization layer and output $6 \times 18 \times 128$ feature maps.
- (9) Convolutional layer C8: we use 128 convolution kernels whose size is $3 \times 3 \times 128$, padding is the same convolution, convolution step size is 1, the total required parameters are $(3 \times 3 \times 128 + 1) \times 128 = 147584$, and the activation function is the ReLU function, followed by a layer of batch normalization layer and output $6 \times 18 \times 128$ feature maps.
- (10) Pooling layer P9 layer: we use the maximum pooling algorithm for the output result of the C8 layer for pooling operation, and the size of the down-sampling window used is 2×2 and output $3 \times 9 \times 128$ feature maps.
- (11) Convolutional layer C10 layer: we use 128 convolution kernels whose size is $3 \times 3 \times 128$, padding is the same convolution, convolution step length is 1, the total required parameters are $(3 \times 3 \times 128 + 1) \times 128 = 147584$, and the activation function is the ReLU function, followed by a layer of batch standardization layer and output $3 \times 9 \times 128$ feature maps.
- (12) Convolutional layer C11 layer: we use 128 convolution kernels whose size is $3 \times 3 \times 128$, padding is the same convolution, convolution step is 1, the total required parameters are $(3 \times 3 \times 128 + 1) \times 128 = 147584$, and the activation function is the ReLU function, followed by a layer of batch standardization layer and output $3 \times 9 \times 128$ Feature Maps.
- (13) Pooling layer P12: the output of C11 layer is pooled by max pooling algorithm, and the size of the down-sampling window used is 2×2 and output $1 \times 4 \times 128$ feature maps.
- (14) Flattening layer F13 layer: we flatten the data of feature maps output by P12 layer and a total of $1 \times 4 \times 128 = 512$ nodes.
- (15) The dense layer is fully connected with the F13 layer, connecting 4 classifiers; each classifier contains 36 neural nodes, the activation function is the sigmoid function, and the maximum probability one-hot encoding of a character CAPTCHA is

output. Each classifier requires a total of $36 \times (1 \times 4 \times 128 + 1) = 18468$ parameters.

4. Experiments

4.1. Dataset. The CAPTCHA dataset used in this article includes CNKI CAPTCHA, Zhengfang CAPTCHA, and randomly generated CAPTCHA. All CAPTCHA datasets are composed of uppercase and lowercase letters and numbers, including 33 categories.

CNKI CAPTCHA contains common CAPTCHA interference methods, such as character scale change, linear noise, and character adhesion, which is more suitable for testing the applicability of CAPTCHA. The image dataset includes 4000 images in the training set and 600 images in the test set. The sample image is shown in Figure 4.

ZhengFang educational administration system CAPTCHA has the characteristics of point noise and partial distortion adhesion, which can be used to evaluate the performance of the recognition method of the adhesive character verification code. We use 2000 such CAPTCHA datasets as the training set and 200 as the test set, and we manually label some of the renamed pictures. The sample image is shown in Figure 5.

The random generated CAPTCHA has the characteristics of moderate distortion and adhesion, which cannot be recognized by the traditional CAPTCHA recognition methods. We generate 10,000 CAPTCHA images as the training set and 2000 as the test set. The naming format is the characters represented by the image plus the sequential serial number to prevent errors from appearing with the same image. The sample image is shown in Figure 6.

4.2. Performance. Firstly, the CAPTCHA is preprocessed, then the dataset is input into the network model for training and parameter adjustment, and then the test samples are predicted. We count the number of true positive (TP) and the number of true negative (TN) and finally calculate the accuracy rate according to the statistical results, $\text{acc} = \text{TP} / (\text{TP} + \text{TN})$. The following is a graph of the accuracy and loss value of the convolutional neural network on the three datasets as shown in Figures 7–9 and the test results on each dataset are also given in Figures 10–12. It can be seen from the figure that the method proposed in the paper has a higher recognition rate and better robustness.

For CAPTCHA that contains complex interference information or adhesions, traditional methods based on image segmentation are difficult to identify, and segmentation will destroy character information and cause errors to accumulate. With the end-to-end deep learning technology, there will be a prediction of the result from the input end to the output end. The prediction error is transmitted and adjusted



FIGURE 4: CNKI CAPTCHA.



FIGURE 5: ZhengFang CAPTCHA.

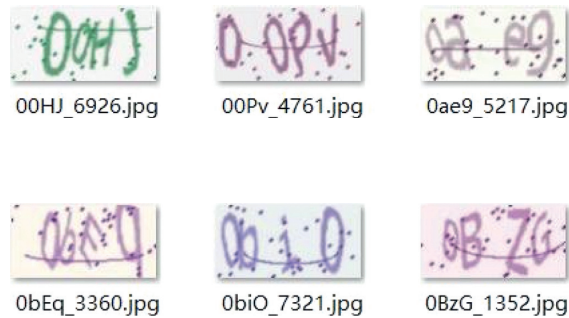


FIGURE 6: Random generated CAPTCHA.

in each layer of the network until the expected result is obtained. By introducing this self-learning network architecture into the CAPTCHA recognition, the character segmentation step can be removed, and the preprocessing operation can be selected according to the interference complexity in the training sample, so as to better highlight and retain the characteristic information between characters.

In order to further verify the performance of the method proposed in the paper, Table 2 shows the recognition rates of different deep learning methods under three different verification codes, including methods such as AlexNet, VGG, GoogleNet, and ResNet. As can be seen from the figure, the recognition rate of the proposed method for CNKI CAPTCHA is 2.05%, 2.42%, 1.66%, and 0.24% higher than AxNet, VGG-16, GoogleNet, and ResNet, respectively, and

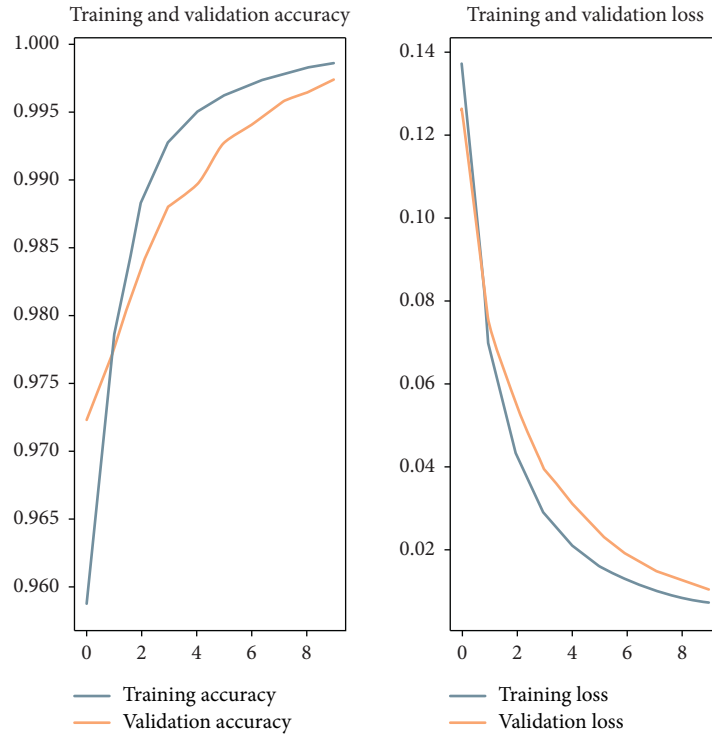


FIGURE 7: Curve of accuracy and loss value on CNKI CAPTCHA.

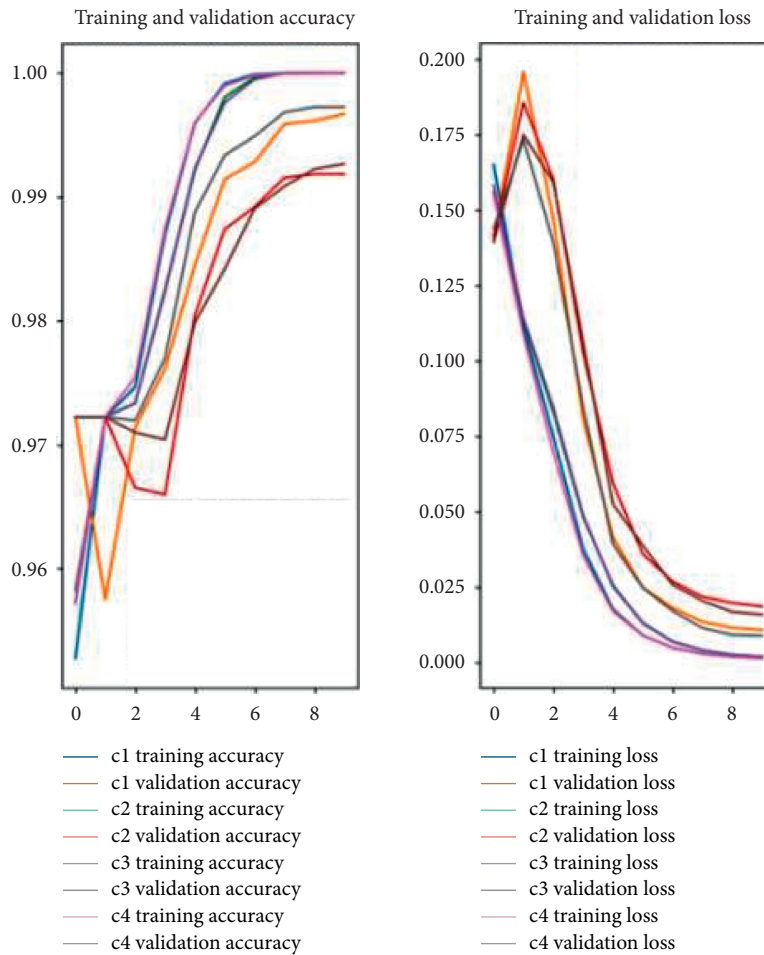


FIGURE 8: Curve of accuracy and loss value on ZhengFang CAPTCHA.

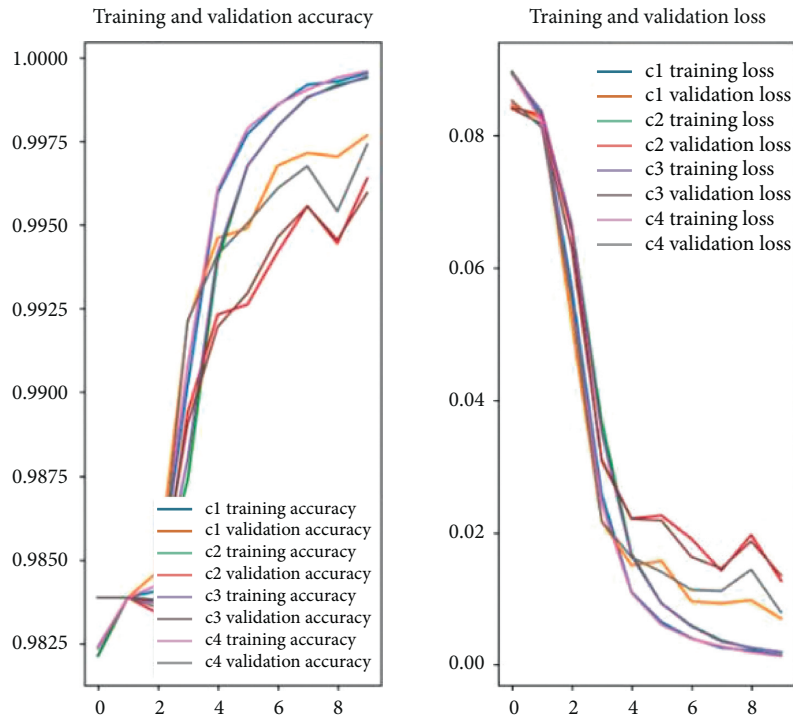


FIGURE 9: Curve of accuracy and loss value on random generated CAPTCHA.



FIGURE 10: Recognition results on CNKI.



FIGURE 11: Recognition results on ZhengFang.

the recognition rate of ZhengFang CAPTCHA is increased by 2.25%, 2.61%, 2.85%, and 2.3%, respectively. For the randomly generated CAPTCHA, it is increased by 3.46%,

1.95%, 0.98%, and 0.59%, respectively. The proposed method has high recognition rate, robustness, and good generalization ability for three different datasets.

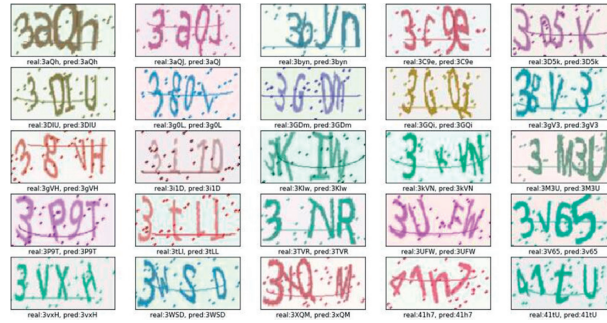


FIGURE 12: Recognition results on random generated CAPTCHA.

TABLE 2: Comparison of recognition accuracy of different CAPTCHA.

Type	AlexNet (%)	VGG-16 (%)	GoogleNet (%)	ResNet (%)	Our method (%)
CNKI CAPTCHA	96.95	96.58	97.34	98.76	99
ZhengFang CAPTCHA	96.25	95.89	95.65	96.2	98.5
Random generated CAPTCHA	94.38	95.89	96.86	97.25	97.84

5. Conclusion

This paper proposes a convolutional neural network method based on focal loss for CAPTCHA recognition. The focal loss function is introduced to solve the imbalance problem of positive and negative samples and difficult and easy samples. Firstly, preprocessing such as graying, binarization, denoising, segmentation, and labeling is carried out, and then a simple neural network model is constructed by using Keras library; in addition, an end-to-end neural network model is constructed for the complex CAPTCHA with high adhesion and more interfering pixels. The test results on three different CAPTCHA datasets show that the proposed method has certain advantages over the traditional methods and has higher recognition rate, robustness, and good generalization ability. In the future, we will study more types of CAPTCHA recognition.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (61976198), the Natural Science Research Key Project for Colleges and University of Anhui Province (KJ2019A0726), High-Level Scientific Research Foundation for the Introduction of Talent of Hefei Normal University (2020rcjj44), and the Anhui Province Key Laboratory of Big Data Analysis and Application Open Project.

References

- [1] B. B. Zhu, J. Yan, G. Guanbo Bao, M. Maowei Yang, and N. Ning Xu, "CAPTCHA as graphical passwords—a new security primitive based on hard AI problems," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 6, pp. 891–904, 2014.
- [2] M. A. Kouritzin, F. Newton, and B. Wu, "On random field completely automated public turing test to tell computers and humans apart generation," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1656–1666, 2013.
- [3] L. Lizhao, L. Jian, D. Yaomei, X. Huarong, Y. Huayi, and Z. Shunzhi, "Design and implementation of verification code identification based on anisotropic heat kernel," *China Communications*, vol. 13, no. 1, pp. 100–112, 2016.
- [4] P. Wang, H. Gao, Z. Shi, Z. Yuan, and J. Hu, "Simple and easy: transfer learning-based attacks to text CAPTCHA," *IEEE Access*, vol. 8, pp. 59044–59058, 2020.
- [5] P. Wang, H. Gao, Q. Rao, S. Luo, Z. Yuan, and Z. Shi, "A security analysis of CAPTCHAS with large character sets," *IEEE Transactions on Dependable and Secure Computing*, p. 1, 2020.
- [6] Y. H. Shen, R. G. Ji, D. L. Cao, and M. Wang, "Hacking Chinese touclick CAPTCHA by multiscale corner structure model with fast pattern matching," *Proceedings of the ACM International Conference on Multimedia*, pp. 853–856, 2014.
- [7] M. Belk, C. Fidas, P. Germanakos, and G. Samaras, "Do human cognitive differences in information processing affect preference and performance of CAPTCHA?" *International Journal of Human-Computer Studies*, vol. 84, pp. 1–18, 2015.
- [8] V. P. Singh and P. Pal, "Survey of different types of CAPTCHA," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 2, pp. 2242–2245, 2014.
- [9] P. Lupkowski and M. . Urbanski, "SemCAPTCHA—user-friendly alternative for OCR-based CAPTCHA systems," in *Proceedings of the 2008 International Multiconference on Computer Science & Information Technology*, pp. 325–329, IEEE, Wisia, Poland, October 2008.
- [10] P. Golle and N. Ducheneaut, "Keeping bots out of online games," in *Proceedings of the International Conference on*

- Advances in Computer Entertainment Technology*, pp. 262–265, DBLP, Valencia, Spain, June 2005.
- [11] K. Chellapilla and P. Y. Simard, “Using machine learning to break visual human interaction proofs (HIPs),” *Advances in Neural Information Processing Systems*, pp. 265–272, 2004.
- [12] J. Yan and A. S. El Ahmad, “Breaking visual CAPTCHAs with naive pattern recognition algorithms,” in *Proceedings of the 23rd Annual Computer Security Applications Conference*, pp. 279–291, Miami Beach, FL, USA, December 2007.
- [13] J. Yan and A. S. El Ahmad, “A low-cost attack on a Microsoft CAPTCHA,” in *Proceedings of the ACM Conference on Computer & Communications Security*, pp. 543–544, Alexandria, Virginia USA, October 2008.
- [14] E. Bursztein, M. Martin, and J. Mitchell, “Text-based CAPTCHA strengths and weaknesses,” in *Proceedings of the 18th ACM conference on Computer and communications security*, pp. 125–138, Chicago, IL, USA, October 2011.
- [15] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, “The robustness of hollow CAPTCHAs,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1075–1086, Berlin, Germany, November 2013.
- [16] H. Gao, J. Yan, F. Cao et al., “A simple generic attack on text captchas,” in *Proceedings of the Network & Distributed System Security Symposium*, pp. 220–232, San Diego, CA, USA, February 2016.
- [17] K. Qing and R. Zhang, “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)*, pp. 1313–1317, IEEE Computer Society, Kyoto, Japan, November 2017.
- [18] B. Shi, X. Bai, and C. Yao, “An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 11, pp. 2298–2304, 2016.
- [19] F.-L. Du, J.-X. Li, Z. Yang, P. Chen, B. Wang, and J. Zhang, “CAPTCHA recognition based on faster R-CNN,” *Intelligent Computing Theories and Application*, pp. 597–605, 2017.
- [20] D. Lin, F. Lin, Y. Lv, F. Cai, and D. Cao, “Chinese character CAPTCHA recognition and performance estimation via deep neural network,” *Neurocomputing*, vol. 288, pp. 11–19, 2018.
- [21] F. H. Alqahtani and F. A. Alsulaiman, “Is image-based CAPTCHA secure against attacks based on machine learning? an experimental study,” *Computers & Security*, vol. 88, 2019.
- [22] H. Yu, S. Xiao, Z. Yu, Y. Li, and Y. Zhang, “ImCAPTCHA: imperceptible CAPTCHA based on cursor trajectories,” *IEEE Consumer Electronics Magazine*, vol. 9, no. 1, pp. 74–82, 2020.
- [23] J. Wang, J. Qin, J. Qin, X. Xiang, Y. Tan, and N. Pan, “CAPTCHA recognition based on deep convolutional neural network,” *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 5851–5861, 2019.
- [24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.
- [25] C. Funk and Y. Liu, “Symmetry reCAPTCHA,” in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5165–5174, IEEE, Las Vegas, NV, USA, June 2016.
- [26] M. Osadchy, J. Hernandez-Castro, S. Gibson, O. Dunkelman, and D. Pérez-Cabo, “No bot expects the deep CAPTCHA! introducing immutable adversarial examples, with applications to CAPTCHA generation,” *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2640–2653, 2017.
- [27] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, “Selective learning confusion class for text-based CAPTCHA recognition,” *IEEE Access*, vol. 7, pp. 22246–22259, 2019.
- [28] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 99, pp. 2999–3007, 2017.