Recognizing Character-Matching CAPTCHA Using Convolutional Neural Networks with Triple Loss

Junfeng Hu, Wenchao Ma, Aamir Khan, and Li Liu(✉)

School of Big Data and Software Engineering, Chongqing University, Chongqing, China
dcsliuli@cqu.edu.cn

Abstract. Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) is a widely used type of challenge-response test to determine whether or not the user is human in many web applications. The traditional CAPTCHAs with English and Chinese characters can be automatically recognized with high accuracy. Yet current methods are limited in recognizing new CAPTCHAs such as character-matching CAPTCHA. We present an approach that combines convolution neural network with triple loss to solve character-matching CAPTCHA. We evaluate our approach on five types of CAPTCHAs including character-matching CAPTCHA. The experimental results show that our approach outperforms other four common recognition methods in the aspects of both accuracy and convergence speed.

Keywords: CAPTCHA · Convolutional neural network · Triple loss Recognition

1 Introduction

This century witnesses the expansion of information, inferable from which, an extensive number of data is put away on Cloud services. Furthermore, there are data interchange between people and network constantly in our daily life. However, the wide use of web spiders has brought about the main problem that a large number of users' data is stolen, which exerts a pernicious influence on people's life. Therefore, what should be taken into consideration is how to protect users' privacy and data from being stolen. CAPTCHA [1] is a test to distinguish computers and humans automatically. CAPTCHA is adopted for protecting service and the data from being abused or crawled by automatic program. Currently, a vast number of internet service providers will ask users to recognize CAPTCHA correctly when registering or operating other action.

As computer vision develops, regular CAPTCHA like Fig. 1 is easy enough for computers to accurately recognize, which urges people to develop a more complex CAPTCHA. One of the novel CAPTCHA systems based on images is designed by Datta et al. [5], which asks users to match the given images to their introductions in 30 s. Meanwhile, in order to prevent computers recognizing images, some semitransparent color lumps are added to blend them up. With respect to these novel CAPTCHAs, a complex CAPTCHA (Character-Matching CAPTCHA) that based on Chinese characters tends to be used, which requires users to match four characters to the given pictures in proper order like Fig. 6(e). There are some difficulties to recognize this CAPTCHA: 1. Due to more complicated stroke and structure, CAPTCHA based on Chinese characters is quite different from CAPTCHA based on regular English characters. Furthermore, it will be difficult for computer to recognize when resizing characters and adding interference lines on image. 2. Unlike the regular CAPTCHAs shown in figure two (a)(b)(c)(d), whose each character corresponds to a signal label, Character-Matching CAPTCHA cannot predict character directly for there are more than three thousands Chinese characters and the total amount of Chinese characters tends to be up to ten thousands, which is much more than English characters. Though in [13], there are some Chinese characters have been recognized, the data set just comprises of two hundred Chinese characters which can lessen the trouble of recognition. In this paper, we develop a method based on convolutional neural network to solve this novel CAPTCHA and evaluate the performance of our method.

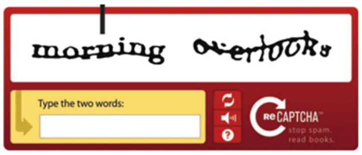

Fig. 1. Google's RECAPTCHA [20] include a distorted word and a word scanned from a book, it also adds interference lines. Users need to enter two words together would they be able to approve.

2 Related Work

Generally, there exists three kinds of CAPTCHAs: text-based [20], image-based [5], and audio-based [2,6,21] CAPTCHAs. We mainly focus on text-based CAPTCHAs in this paper.

For early CAPTCHAs, the procedure to the recognition CAPTCHAs is like [10]. For the most CAPTCHAs, people separated it into two sections: First, recognize the area that the individual character occupies and then divide it [14]. Second, recognize each character individually [17]. For example, the CAPTCHAs problem can be solved by means of separating individual character and recognizing them by Chellapilla et al. [4]. After that, LeCun et al. [12] asserted that convolutional neural network (called LeNet network) is adopted for recognizing handwritten text and obtained high accuracy. However, when Mori et al. [15] pointed out in 2003 that text-based CAPTCHAs could be recognized with a high recognition rate, all the CAPTCHAs have been designed to a more complex format. Ligature and torsion resistance are used for some CAPTCHAs, which lead to the failure of recognizing by separating them into single character. LeNet is unable to solve the more complicated CAPTCHAs above. Stark et al. [19] achieved a high accuracy by recognizing English characters directly on small data set rather than separating them. Bursztein et al. [3] researched and developed Decaptcha and recognized CAPTCHAs of 13 websites successfully, including Google. In 2014, Goodfellow et al. proposed a recognition method of multi-character text based on deep CNN with localization, segmentation, and recognition. Yunhang Shen et al. [8] proposed a Multi-Scale Corner Structure Model for Chinese Touclick CAPTCHA recognition.

The above research has obtained good results in CAPTCHAs recognition. However, as CAPTCHAs develops, more novel CAPTCHAs tend to be designed, that is, the regular CAPTCHAs recognition loses its efficiency in these novel CAPTCHAs. These novel CAPTCHAs are applied to many famous websites, including Alibaba and ebay. In this paper, we develop a new method to recognize one of these novel CAPTCHAs. As shown in Table 1, we compare several models to find whether they can finish five tasks of data sets.

Table 1. We consider the model can complete the task if it achieves a accuracy of 95% or higher. LeNet can achieve a high success rate in task one only. The structure of Lin et al. can achieve high success rate in both Chinese and English CAPTCHAs but not in the new matching CAPTCHAs. The structure of Stark F et al. can achieve task one and task three only. However, we can achieve all five tasks with high success rate.

	Task one	Task two	Task three	Task four	Task five
LeNET	√	×	×	×	×
Lin et al.	√	√	√	√	×
Shark F et al.	√	×	√	×	×
OURS	√	√	√	√	√

3 Methods

3.1 Binary Option

For text based CAPTCHAs, color image tends to be less effective to improve the accuracy of recognition. The binary option can significantly reduce the amount of computation required without reducing accuracy. We first process graying operation on image:

$$Gray = R \times 0.299 + G \times 0.587 + B \times 0.114, \tag{1}$$

where R, G, and B denotes three channel of a color image. Then we will process binary operation. Nowadays, the best method of image binary option is called

Maximum Between-Class Variance from Nobuyuki Otsu [9] and its abbreviation is Otsu. Otsu is a self-adaptive threshold determination method. It divides the image into two parts by threshold value T based on the gray feature of the image. These two parts of image are called front image and back image. When we get the best threshold, these two parts should be the most different. In Otsu algorithm, the metric to measure the difference is Maximum Between-Class Variance. The probability of being wrongly divided attain minimum when the segmentation of threshold values makes the largest variance. It is assumed that T means the threshold of front image and back image. Meanwhile, the ratio of front and back images to total image is W_0, W_1 and average gray value is U_0, U_1. The average variance of front image and back image is U:

$$U = W_0 \times U_0 + W_1 \times U_1 \tag{2}$$

$$g = W_0 \times (U_0 - U)^2 + W_1 \times (U_1 - U)^2 \tag{3}$$

yielding:

$$g = W_0 \times W_1 \times (U_0 - U_1)^2 \tag{4}$$

when g gets maximum, the difference between front and back image is the largest and the threshold value T is the best. Figure 2 shows examples after binary operation.

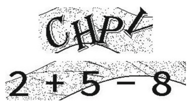

Fig. 2. Examples of a binary operated image with background noise and rotational deformation.

3.2 Image Noise Reduction

Image noise has a significant influence on the accuracy of CAPTCHA recognition, it is important to reduce image noise before recognition. Compared to other linear filtering like average filtering, median filtering is a non-linear image noise reduction method, which can remove spot and spiced salt noise effectively and protect image edges. Although, Median filtering is a domain operation which is similar to convolution, it is not a weighted sum. Median filtering sorts the pixels in order by grayscale and then selects the median as the output of pixels. Median filtering is defined as follows:

$$\mathrm{g}(x, y) = \underset{(i,j)\in W}{median}\{f(x \pm i, y \pm j)\}, \tag{5}$$

where g(x, y) is the output of the pixel gray scale, $f(x \pm i, y \pm j)$ is the input of the pixel gray scale, and W indicates template window that can be Line font, rhombus, or rectangle and so on. As shown in Fig. 3, a result of image noise reduction can be obtained by selecting a specific window in accordance with the images in five tasks. After image noise reduction, the image still has interference lines that make it difficult for our model to recognize characters. Compared to characters, the interference line has smaller width. The equation is written as:

$$n = \underset{(i,j)\in N^*}{sum}(Sgn(f(x \pm i, y \pm j))), \tag{6}$$

where i, j are positive integer and f(x−i, y−i) indicates input pixel. After selecting the specific threshold value, f(x, y) is isolated and interference lines should be removed when n is smaller than threshold value. Figure 4 shows the image after removing interference lines.

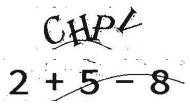

Fig. 3. It is assumed that W is square and the size of window W can be set according to the task. Apart from interference lines, spiced salt noise has been removed effectively after median filtering noise reduction.

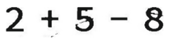

Fig. 4. Interference line has been removed.

3.3 Segmentation

Since Arithmetic CAPTCHA has diverse characters number for different examples and the method we use to recognize character-matching CAPTCHA, we need to segment characters on these two CAPTCHAs. We use a method called X-axis Pixel Projection. The process of the method is that we sum the number of black spots on each vertical axis. Obviously, if one vertical line is background, the number of the black spots should be close to zero. Otherwise, if one vertical line passes a character, there are more black dots on this vertical line. For two standalone characters, there must be at least one vertical line that has no black spot. However, if the characters are cohesive, we cannot find a vertical line that has no black spot. In these cases, we find a vertical line having a minimum of black spot between two cohesive characters can be a connection between two characters. Threshold value can be set in accordance with different situations. When black spot for one vertical line is less than threshold value, we segment character on this vertical line. Figure 5 shows an image after segmentation.

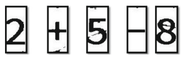

Fig. 5. Image has been segmented into single characters.

4 Experiment

In experiment section, we outline two experiments to solve five undertakings we have depicted previously. Experiment one is based on regular CAPTCHA from task one to task four. While experiment two focuses on novel CAPTCHA (character-matching CAPTCHA). From this, our model's performance can be evaluated. All experiments have been executed using Tensorflow 1.5 on a NVIDA GeForce 1080 Ti GPU.

4.1 Data Sets

The data set that provided by In spur Technologies Co., Ltd. is used for our model. The data set mainly has five parts: I. Arithmetic CAPTCHA, which contains arithmetic. As shown in Fig. 6(a), results can be obtained by arithmetic. II. English Alphabet and Digital CAPTCHA, which contains five characters. As shown in Fig. 6(b), results can be obtained by typing all the characters, which should be converted to uppercase. III. English Alphabet and Digital CAPTCHA, which is as same as type two but has four characters like Fig. 6(c). IV. Chinese rotational characters CAPTCHA. which contains four Chinese characters. Users are requested to select one of the four characters that is rotated by 90° like Fig. 6(d). V. character-matching CAPTCHA, which contains an image with four Chinese characters (called verification image) and nine images (called matching image) including a single character. As shown in Fig. 6(e), results are the order number of matching images in conformity with the verification image.

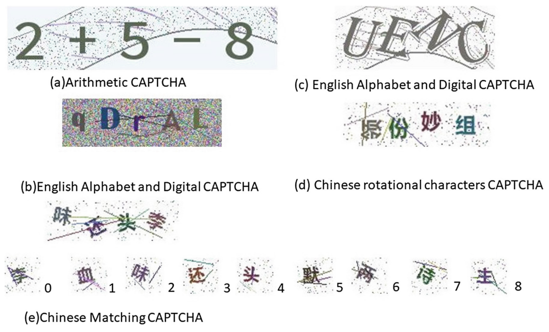
(a)Arithmetic CAPTCHA
(c) English Alphabet and Digital CAPTCHA
(b)English Alphabet and Digital CAPTCHA
(d) Chinese rotational characters CAPTCHA
0
1
2
3
4
5
6
7
8
(e)Chinese Matching CAPTCHA

Fig. 6. Examples for different CAPTCHAS in our data set.

4.2 Network Design

Based on LeNet structure, we add two convolutional layers. Figure 7 shows the structure of our CNN network. Our CNN consists of four convolutional layers, which has a size of 16, 32, 64, 128 and have a kernel of 3×3. To make it sure that after every convolutional operation the feature maps have the same size, we use Zero-Padding. After each convolutional layer, there are a max pooling layers that each filter has a stride of 2×2 and size of 2×2. Then, the network has one fully connected layer with a size of 512 and the final output layer is designed depending on the task. We use ReLu function to activate feature map and a method called dropout [18] is used to prevent overfitting.

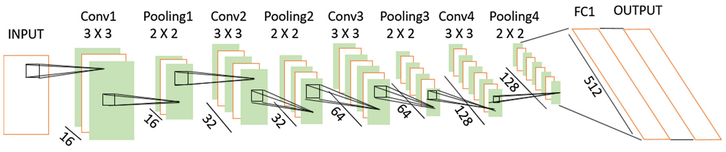

Fig. 7. Convolutional neural network structure.

4.3 Experiment I

For task one Arithmetic CAPTCHA, we define a bisection $\Theta_1(x)$ that maps a character $x \in \{'0'...'9','+','-','\times'\}$ to a positive integer :

$$\Theta_1(x) = \begin{cases} 0...9, & if x = '0'...'9' \\ 10,11,12 & if x = '+','-','\times' \end{cases} \qquad (7)$$

For task two and task three, apart from uppercase and lowercase of English character, we define a bisection $\Theta_2(x)$ that maps a character $l \in \{'0',...'11'\}$ to a positive integer $l \in \{'0',...'35'\}$:

$$\Theta_2(x) = \begin{cases} 0...9, & if x = '0'...'9' \\ 10...35 & if x = 'a/A'...'z/Z' \end{cases} \qquad (8)$$

Softmax loss function can be used in task one and four while sigmoid cross entropy loss function in task two and three. Image segmentation is adapted for task one and five. Due to adherence of characters, it is difficult to segment in task three. Conversely, task two and three can achieve a high accuracy without image segmentation. For task two and three, we assign the first 36 output neurons to the first character of the sequence, the second 36 neurons to the second character, and so on. Hence, for the i-th character, the neuron n is calculated as follows:

$$n = 36 \times i + \Theta_2(x_i), \qquad (9)$$

where i $\in \{0, 1, 2, 3, 4\}$ (for task two) or i $\in \{0, 1, 2, 3\}\{0, 1, 2, 3, 4\}$(for task three). For task two, the output layer has $36 \times 5 = 180$ neurons and for task three, the output layer has $36 \times 3 = 144$ neurons. Figure 8 shows an instance of a network

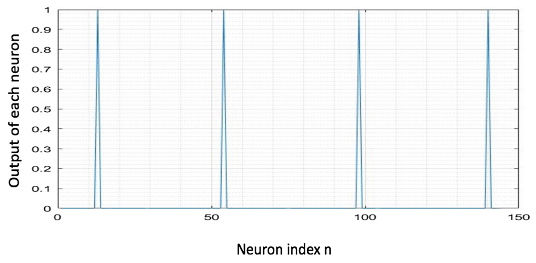

Fig. 8. Output of the network for the CAPTCHA 'CHPV'.

output for task three. The predicted index for the first character is 12, so the character is 'C'. Xavier initialization [7] is adopted for training the model. Then the model is optimized by Adam [11] algorithm, with batch size of 64. The learning rates for task one to four are 0.0001, 0.00005, 0.00005, and 0.0001.

The performances of our model with other common machine learning methods including LeNet are shown in Table 2. On account that task one is easy to recognize, our model doesn't reveal enormous advantage when compared to other methods. When it comes to task two, three, and four, the accuracy of our model is much higher than other methods. As shown in Fig. 9, when compared to LeNet, our model has a faster convergence in all tasks and has a higher accuracy in task two,three, and four. We get high accuracy which is close to 100% within a short 50 iterations in task one. An accuracy of 99.0% can be obtained in task four after 300 iterations while an accuracy of 99.9% can be obtained in task two after 500 iterations. Our model has a slower convergence and in task two than that of other tasks. Due to the problem of uppercase, lowercase and more noises, it is difficult to recognize in task two. As indicated in Fig. 9, with the increase of iterations, the accuracy of LeNet doesn't develop in task two, that is, LeNet cannot obtain effective features in task two.

Table 2. The performance of different machine learning methods.

Machine learning method	Task one	Task two	Task three	Task four
Our model	1.000	0.999	1.000	0.998
LeNet	1.000	0.182	0.881	0.932
SVM	1.000	0.000	0.000	0.776
Decision tree	0.980	0.040	0.011	0.572
KNN	0.982	0.100	0.000	0.495

4.4 Experiment II

It is hard to achieve high accuracy with a limited data set in task five, inferable from which, there are around three thousands basic Chinese characters and a

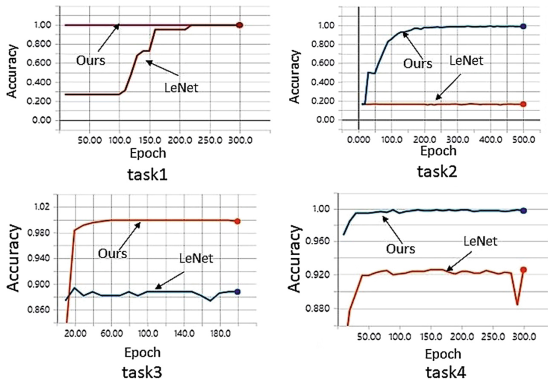

Fig. 9. In all four tasks, our model tends to be faster convergence and higher accuracy.

large number of Chinese characters in total. Inspired by Schroff et al. [16] from Google, we use triple loss in our task five.

As indicated in Fig. 10, a triple can be defined as follows: first, after image segmentation, we select a sample from data set that is called Anchor, and then selected two samples, one belongs to the same class as anchor (called Positive) and the other one belongs to the different class (called Negative). With respect to each element in the triple, we project the sample onto a single point in the embedding space by a parameters-shared network, which is written as $f(x_a^i)$, $f(x_p^i)$, and $f(x_n^i)$. It is preferred that the distance between an Anchor and a Positive is as close as possible and the distance between the Anchor and a Negative is far. Meanwhile, there should be a margin that indicates the distance between $||f(x_i^a) - f(x_i^p)||_2^2$ and $||f(x_i^a) - f(x_i^n)||_2^2$, yielding:

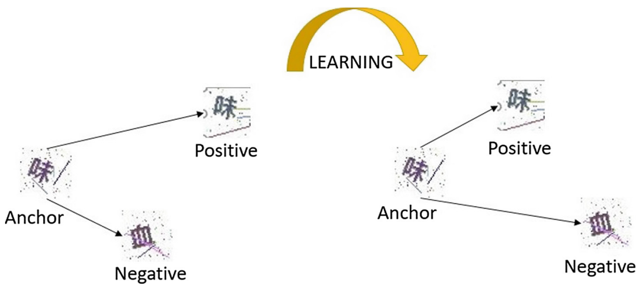

Fig. 10. Every triple can be classified into three components, Anchor, Positive, and Negative. We want CNN to learn to minimize the distance between an Anchor and a Positive and maximize the distance between the Anchor and a Negative.

$$||f(x_i^a) - f(x_i^p)||_2^2 + a < ||f(x_i^a) - f(x_i^n)||_2^2 \tag{10}$$

The triple loss is defined as:

$$\sum_i^N [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + a]_+ \tag{11}$$

The difficulty of using triple loss to train CNN is that triple loss is hard to be convergent to the minimum. Thus, The Anchor's distance between the positive should be as far as possible, while the distance between the Negative should be as close as possible. However, our amount of triples is $10,000 \times C_4^1 \times C_8^1 = 320,000$ which is much smaller than the data set in [16]. Therefore, we will not select triples manually. With the increase of iteration, the maximum, minimum, and average distance of Anchor-Positive and the Anchor-Negative are shown in Fig. 11. During training, the maximum, minimum, and average distance keep increasing but the average distance attains 9.954 after 20 iterations, that is, our CNN structure learns base difference between different samples rapidly. After 180 iterations, the difference between Anchor-Positive average distance and Anchor-Negative average distance attain 14.05, that is, our model could distinguish the otherness between different samples and can get high accuracy. However, the Anchor-Positive max distance is still larger than Anchor-Negative min distance, that is, there are still some samples that our model cannot distinguish well. When testing our model, we first project the sample onto a single point in the embedding space that we describe above. Then, we calculate the L2 distance between verification images and matching images. We select a verification image and matching image that have minimum L2 distance, which means that this verification matches the matching image. After this, we remove the verification image and the matching image. Then, repeat the process until all the verification

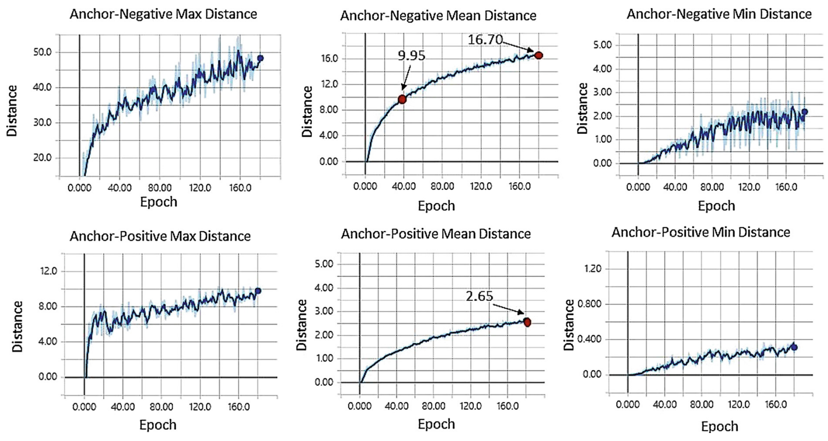

Fig. 11. The maximum, minimum, and average distance between the Anchor and Positive and between the Anchor and Negative with the increase of iteration.

images match matching images. Finally, the accuracy of out model tends to be 97.9% and the LeNet is 46.6%, that is, our model can finish the task better.

5 Conclusion

Keeping in mind the goal to tackle novel CAPTCHAs, we advance a model based on convolutional neural network, which can get high accuracy on common CAPTCHAs as well as on novel CAPTCHAs. With respect to character-matching CAPTCHA, the accuracy rate tends to be 97.9% by using convolutional neural network with triple loss. Meanwhile, compared with LeNet, our model can get higher accuracy on all five tasks.

Acknowledge. This work was supported by grants from the Fundamental Research Funds for the Key Research Programm of Chongqing Science & Technology Commission (grant no. cstc2017rgzn-zdyf0064), the Chongqing Provincial Human Resource and Social Security Department (grant no. cx2017092), the Central Universities in China (grant nos. CQU0225001104447).

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: using hard AI problems for security. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 294–311. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_18
2. Bursztein, E., Beauxis, R., Paskov, H., Perito, D.: The failure of noise-based non-continuous audio captchas. In: Security and Privacy, pp. 19–31 (2011)
3. Bursztein, E., Martin, M., Mitchell, J.: Text-based CAPTCHA strengths and weaknesses. In: ACM Conference on Computer and Communications Security, pp. 125–138 (2011)
4. Chellapilla, K., Simard, P.Y.: Using machine learning to break visual human interaction proofs (HIPS). In: Advances in Neural Information Processing Systems, pp. 265–272 (2004)
5. Datta, R., Li, J., Wang, J.Z.: IMAGINATION: a robust image-based CAPTCHA generation system. In: ACM International Conference on Multimedia, November, Singapore, pp. 331–334 (2005)
6. Gao, H., Liu, H., Yao, D., Liu, X., Aickelin, U.: An audio CAPTCHA to distinguish humans from computers, pp. 265–269. Social Science Electronic Publishing (2010)
7. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. J. Mach. Learn. Res. **9**, 249–256 (2010)
8. Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. Comput. Sci. (2013)
9. IEEE: IEEE xplore abstract - a threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. (1979)
10. Jaderberg, M., Vedaldi, A., Zisserman, A.: Deep features for text spotting. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8692, pp. 512–528. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10593-2_34

11. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. Comput. Sci. (2014)
12. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
13. Lin, D., Lin, F., Lv, Y., Cai, F., Cao, D.: Chinese character CAPTCHA recognition and performance estimation via deep neural network. Neurocomputing **288**, 11–19 (2018)
14. Lu, Y.: Machine printed character segmentation; an overview. Pattern Recognit. **28**(1), 67–80 (1995)
15. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. In: Proceedings of CVPR, vol. 1, p. 134 (2003)
16. Schroff, F., Kalenichenko, D., Philbin, J.: FaceNet: a unified embedding for face recognition and clustering. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 815–823 (2015)
17. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks (2003)
18. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
19. Stark, F., Hazirbas, C., Triebel, R., Cremers, D.: CAPTCHA recognition with active deep learning. In: German Conference on Pattern Recognition Workshop (2015)
20. Von, A.L., Maurer, B., Mcmillen, C., Abraham, D., Blum, M.: reCAPTCHA: human-based character recognition via web security measures. Science **321**(5895), 1465–1468 (2008)
21. Wu, Y., Lin, J.J.: An improved adaptive noise estimation in Kalman filtering. J. East China Univ. Sci. Technol. (2004)