Combining convolutional neural network and self-adaptive algorithm to defeat synthetic multi-digit text-based CAPTCHA

Ye Wang*, Yuanjiang Huang†, Wu Zheng†, Zhi Zhou†, Debin Liu†, Mi Lu*
*Texas A&M University
wangye0523@tamu.edu, mlu@ece.tamu.edu
†SocialCredits, Ltd
{yuanjiang.huang, wu.zheng, zhi.zhou, debin.liu}@socialcredits.cn

***Abstract*—We always use CAPTCHA(Completely Automated Public Turing test to Tell Computers and Humans Apart) to prevent automated bot for data entry. Although there are various kinds of CAPTCHAs, text-based scheme is still applied most widely, because it is one of the most convenient and user-friendly way for daily user [1]. The fact is that segmentations of different types of CAPTCHAs are not always the same, which means one of CAPTCHA's bottleneck is the segmentation. Once we could accurately split the character, the problem could be solved much easier. Unfortunately, the best way to divide them is still case by case, which is to say there is no universal way to achieve it. In this paper, we present a novel algorithm to achieve state-of-the-art performance, what was more, we also constructed a new convolutional neural network as an add-on recognition part to stabilize our state-of-the-art performance of the whole CAPTCHA system. The CAPTCHA datasets we are using is from the State Administration for Industry& Commerce of the People's Republic of China. In this datasets, there are totally 33 entrances of CAPTCHAs. In this experiments, we assume that each of the entrance is known. Results are provided showing how our algorithms work well towards these CAPTCHAs.**

***Index Terms*—CAPTCHA, convolutional neural network, segmentation, reverse Turing test, clustering**

I. Introduction

We rely on CAPTCHA increasingly heavily in distinguishing between human beings and computer programs automatically. Because of the combination of distorting characters and obfuscation techniques which can be recognized by people but may be hard for automated bots [2] [3], and text-based CAPTCHAs are most widely used by both companies and individuals.

We divide text-based CAPTCHA defeating into three parts: denoising, segmentation and recognition. They are equally vital, regarding preprocessing step. The purpose is to decrease the noise influence for ensuring the correct segmentation which can increase the recognition accuracy rate. Actually, there exist a lot of methods, including the combination of image processing and artificial intelligence algorithms, like median filter [4], neighborhood filter, wavelet threshold, universal denies, K-nearest neighbors algorithm, support vector machine and so on. However, how to choose those candidate ways is the key to perform the right denoising [1] [5].

Fig. 1. Online samples in the datasets

Actually, if we could directly utilize OCR, segmentation is not an essential step. While after preprocessing, most of our CAPTCHAs cannot be recognized by OCR. Thus, we should use the segmentation to divide the image into characters for better performance. Towards the segmentation, it heavily depends on the feature of the images. Image intensity histogram and color clustering are the most frequently used techniques [6]. For some special CAPTCHAs to be further discussed later, we present our novel adaptive length of characters to implement the segmentation. Recognition is the last step to get output. Three ways we used here will be introduced. First of all, Optical Character Recognition(OCR) is the way to convert words in pictures or other language related in images to the typed text [7]. It is widely used as a form of data entry from the printed records, whether documents like bank statements, contracts, receipts, even business or credit cards and so on. Currently, the task to identify machine typed text has already been absolutely solved. Many business commercial applications are now in the markets. While the methods failed to CAPTCHAs due to various reasons. Some applications work well on black-and-white images, some based on free-noise text, while majority of the CAPTCHAs we will discuss in this paper are against the rules - they are random noised, colored and even rotated [5] [1]. Moreover, our CAPTCHAs images suffers from inconsistent lighting conditions, and dif-

978-1-5090-5320-9/17/$31.00 ©2017 IEEE

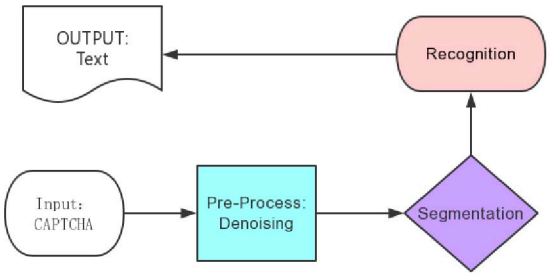

Fig. 2. Basic Flow Chart to defeat the CAPTCHA

ferent fonts, directions, and other distortions. The combination of those complicated problems can not be easily solved by employing only OCR. The second potential way is Template Matching, before OCR, this is the most original artificial intelligence method to execute recognition [8]. The review of TM is considerably straightforward and convenient. At the mean time, for some rotated CAPTCHAs, it can get kind of higher accuracy rate than OCR. We need to figure out that with the development of increasingly complex circumstances, including heavily rotation, overlapping and twisted, CNN(convolutional neural network) overwhelmingly works better than any others in terms of accuracy and time complexity [9]. Every coin has two sides, so does the CNN. It need to be pre-trained with a lot of well prepared samples, which is the essential contribution part of this CNN [10]. Our CNN conquer this problem with a lot of human being efforts aiming at data collection. So that, we finally achieve state-of-the-art performance to defeat the CAPTCHAs.

The rest of this paper is organized as follows: Section II describes the background on the three steps in terms of denoising, segmentation and recognition. Section III proposes the scheme of adaptive segmentation and how we construct our convolutional neural network. Section IV shows the experiment. The paper concludes in Section V.

II. Background

In reality, different kinds of CAPTCHAs can be found in a lot of popular websites [11] [3] [12]. Each of the existing CAPTCHA should be defeated individually because of their unique encoding algorithms. Correspondingly, a lot of deCAPTCHA algorithms have been proposed for most famous companies such as Google, Microsoft, and Facebook. A low-cost attack on a Microsoft CAPTCHA was presented for solving the segmentation resistant task, which achieved a segmentation success rate of higher than 90% [13]. Moreover, another projection-based segmentation algorithm for breaking MSN and YAHOO CAPTCHAs proves to be effective, which doubled the corrected segmentation rate of the traditional method [14]. Besides, an algorithm using ellipse-shaped blobs detection for breaking Facebook CAPTCHA also presents to be useful [15].

As can be seen, Fig. 1 presents some examples of text-based CAPTCHAs which will be discussed later. Fig. 2 shows the basic flow chart to defeat the CAPTCHAs. In this section, we will demonstrate the whole process.

A. Prepossessing Step

There are several potential techniques to denoise the images as we have introduced above. According to the difference of each type, we should correspondingly choose the appropriate method. I basically followed three ways in terms of image processing and artificial intelligence. Median filter is a quite common way for nonlinear digital method to filter the noise, which can improve the performance for next processing. Affine transformation acts an important role in preprocessing. An affine transformation does not necessarily preserve angles between lines or distances between points, though it does preserve ratios of distances between points lying on a straight line [16]. after an affine transformation, the sets of parallel lines still remain parallel, which obviously preserves points, lines and planes.

B. Segmentation

Segmentation is the most vital step due to heavily influence about recognition. The most common way for segmentation analysis is to divide the CAPTCHAs' image into multiple single part [6]. By detecting the CAPTCHAs' boundaries like lines or curves is our goal of segmentation. With consideration of the rotation, noise and even twisted characters, we should break the whole flow of CAPTCHAs into single character for better analysis.

Histogram-based and K-means clustering are two simple but effective segmentation methods. For histogram-based method, which is referred to as computing the number of pixels in each row or column. The basic algorithm would be illustrated as y_0, y_1, ... y_n, where y_i is the number of pixels in the image with gray-level i, and n is the maximum gray-level attained. We can easily imagine that if the histogram between each character is very far, it would be apparent to set the start and end point for each character [18]. K-means clustering is a widely used method in clustering analysis. We focus on partitioning n observation into K clusters to find out which centroid they belong to with the intended nearest mean. Actually, segmentation is a very dependent method since the CAPTCHAs varies considerably. So far, there aren't any useful algorithms to segment towards the affixed, bended, even twisted characters. Besides, the sophisticated combination of histograms, K-means plus the affine transformation are our best weapons to defeat the CAPTCHAs. In the next section, we will focus on an algorithm which contains a lot of CAPTCHAs types by using adaptive length segmentation.

C. Recognition

After segmentation, the last step is to recognize the character, getting text from the images. Here we will introduce our ways. We mainly implement three ways for recognition, Optical character recognition(OCR), Template Matching(TM)

and Convolutional Neural Network(CNN). For OCR, we choose Tesseract by Google [7] [20], which is an open source software. If the format are rigid, the performance is the best. While not all the cases are ruled, for some others that are irregular, TM may be an option, which is a pattern-oriented method to find the best candidate characters. The theory is to slide the template image over the input image and get the similarity of each other to obtain the highest possible label.

Convolutional Neural Network is a more advanced artificial intelligent technique rather than other two ways [21] [9]. Since it needs much more data to train the model for a more robust system in terms of higher accuracy and less time complexity. One of the CNN limitation is certainly the size of datasets, since we need to manually denoise the image and then extract the datasets, and segment them into individual character, thus the working procedure is quite complicated and heavy.

III. The Scheme

A. Modified adaptive algorithm

CAPTCHAs are with various kinds of fonts as Fig.3 shows. All of them come from the same entrance which means we have to deal with them simultaneously. Totally, there exist 8 font types in terms of traditional and simplified Chinese character, arabic numbers, symbol operator and character operator. The first thing for us to do is the right segmentation for each type. Then the rest we need to resolve is to identify which font type we just mentioned they belong to.

One attractive feature of this matrix representation is that we can use it to factor a complex transform into a set of simpler transform. Here we only utilize the rotation transformation, thus the following rotation matrix is essential: $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$, where θ is an angle of counterclockwise rotation around the origin.

After carefully observation, we find out that the angel is fixed, which means that every time when we load an image we first need to do the regular affine transformation to make it vertical. The rest thing is to segment it as the Fig. 6 exhibits. Another issue we need to cope with is the segmentation. As we have introduced before, there are about eight font types, we need to check each of them character by character.

We finally optimize the whole flows, following is the basic idea which we will explain in details. Once the image comes into our system, we should check its first character, if it is an arabic number, we go to the third character, to check whether it is a digit or Chinese character. If it is a digits, we go back to check the second character to see whether it is an algebraic symbol or Chinese character operator. Otherwise, we go ahead to find the fourth character. If the first character is not a digit, we would like to check the second character to check the font type. If the second one is an algebraic symbol, the third character must be the Chinese character, then we can ignore the fourth one. If the second is not an algebraic symbol, the simplest way is to directly see the fourth one whether it is equal to one particular ideograph. Fig.4 shows the basic flow chart of our adaptive segmentation.

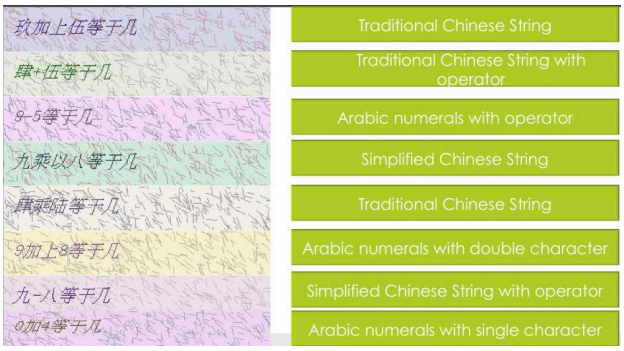

Fig. 3. The example of adaptive segmentation

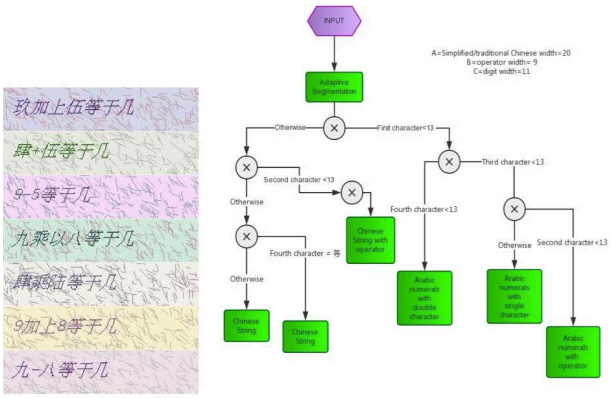

Fig. 4. The detailed flow chart of adaptive segmentation

B. Construct the convolutional neural network

1) The architecture: The architecture is showed in Figure 7, which is one of most classic LeNet-5 model [22] [23]. The top is our image input layer and the size is fixed 32x32. The net contains six layers with weights; the first four are convolutional and the remaining two are fully-connected The output of the last fully-connected layers is fed to a 82-way(10 for digits 0-9, 26 of each for a-z and A-Z, 10 of each for simplified and traditional chinese digits) softmax which produces a distribution over the 82 class labels.

2) Pooling layer: Pooling layers in CNNs summarize the outputs of neighboring groups of neurons in the same kernel map. Traditionally, the neighborhoods summarized by adjacent pooling units do not overlap [24]. To be more precise, a pooling layer can be thought of as consisting of a grid of pooling units spaced s pixels apart, each summarizing a neighborhood of size z x z centered at the location of the pooling unit. Regarding the shape, typical values are 2x2 or no max-pooling. Very large input images may warrant 4x4 pooling in the lower-layers. Keep in mind however, that this will reduce the dimension of the signal by a factor of 16, and may result in throwing away too much information. [9] [21]

3) Hyper-parameters: CNNs are especially tricky to train, as it contains more hyper-parameters than a standard MLP(Multilayer perceptron). While the usual rules of thumb for learning rates and regularization constants still apply, the

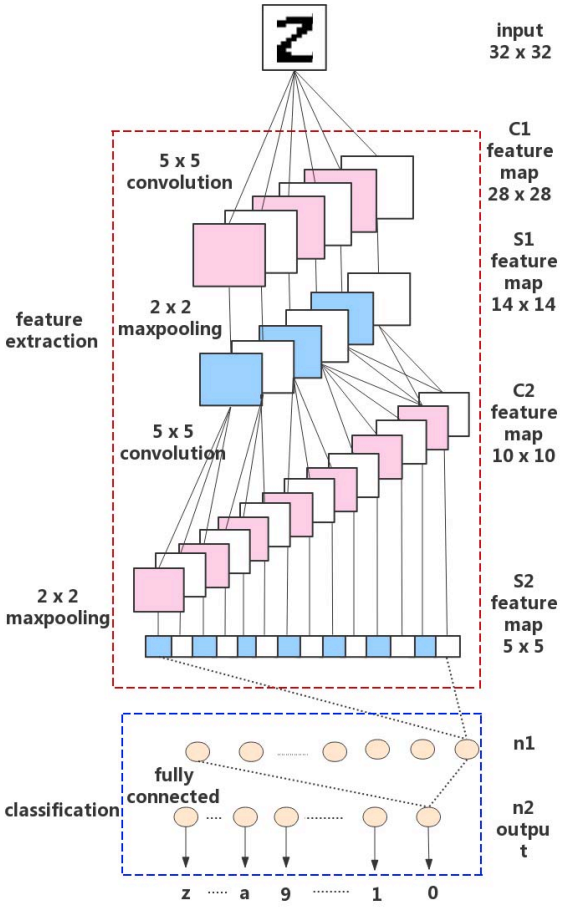

Fig. 5. Neural network structure

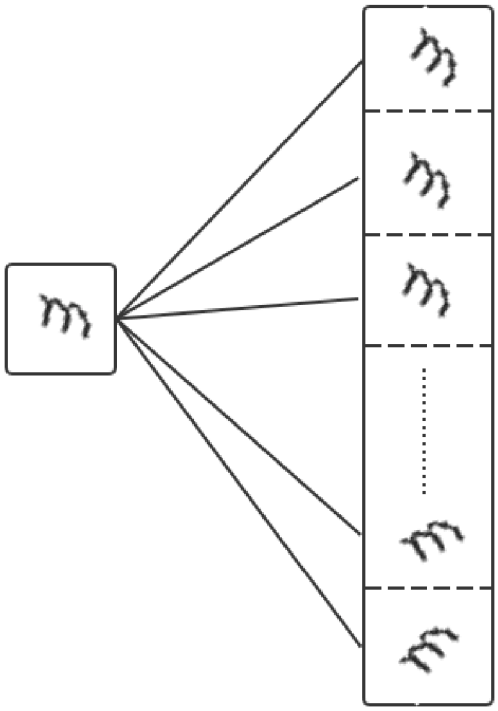

Fig. 6. Rotate the image to diversify training data

following should be kept in mind when optimizing CNNs. [24] Since feature map size decreases with depth, layers near the input layer will tend to have fewer filters while layers higher up can have much more. In fact, to equalize computation at each layer, the product of the number of features and the number of pixel positions is typically picked to be roughly constant across layers. To preserve the information about the input would require keeping the total number of activations (number of feature maps times number of pixel positions) to be non-decreasing from one layer to the next (of course we could hope to get away with less when we are doing supervised learning). The number of feature maps directly controls capacity and so that depends on the number of available examples and the complexity of the task. [25] [9]

IV. Experiment

This paper selects more than 7000 testing samples from the 33 entrances. Due to the similarity, we should first combine the similar CAPTCHAs. Nevertheless, We can observe that the outcome in Fig.8 is very clear. CNN is much more robust than OCR/TM. In the other side, we cannot deny the fact that, the way to get the training data is quite painful. Because we need to manually decompose them into characters. For more data, here we just rotate them from $-50°$ to $50°$ in every $5°$ as Fig.6 shows. So that our data size is ten times bigger than before. Besides, since rotation also brings more circumstances for potential test data, which proved to be more useful.

Furthermore, by comparing the results in Fig. 8 , we can see that some rotated CAPTCHAs usually play worse recognition rate than the others in TM/OCR ways. The best rotated recognition rate is 71.6% in the sixth row. However, the lowest accuracy rate of rotated fonts is as low as 33%. We can clearly see that with the increasing rotated angle, the accuracy rate in TM/OCR is correspondingly decreasing. That is to say, the rotation hinders the accurate segmentation and recognition. By contrast, we can see that the CNN plays much more robust than TM/OCR ways especially in rotation fonts. The average performance in CNN is 10% better than in TM/OCR.

In fact, in some simple cases like first row shows, both TM/OCR and CNN perform similar. With the rotation increases, the more advantages appears in CNN ways. Fig.9 shows the accuracy rate much more clear. Mostly, our CAPTCHA is constructed by four characters, if and only if all of them have been correctly recognized, we count a successful defeat. More precisely, Fig.10 shows the single character recognition accuracy curve chart, as we can see, the

Fig. 7. The full process of defeating the CAPTCHA

The type of CAPTCHA	Accuracy rate	
	TM/OCR	CNN
	288/300=96.0%	296/300=98.6%
	264/300=88.0%	289/300=96.3%
	244/300=81.3%	273/300=91.2%
	241/300=80.3%	260/300=86.7%
	235/300=78.3%	255/300=85%
	215/300=71.6%	263/300=87.6%
	166/300=55.3%	192/300=64%
	162/300=54.0%	195/300=65%
	100/300=33.3%	162/300=54%
	105/300=35.0%	166/300=55.3%
	192/300=64%	200/300=66.6%

Fig. 8. The accuracy rate for each CAPTCHA

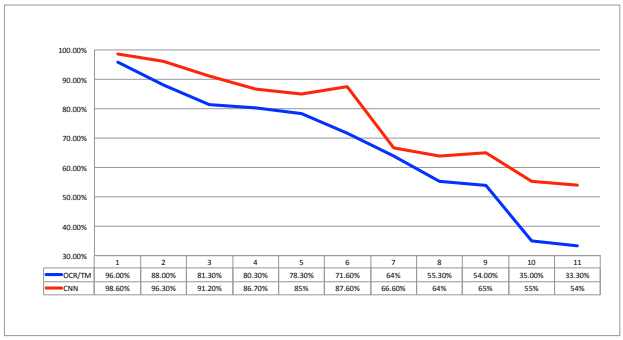

Fig. 9. Curve chart of accuracy rate

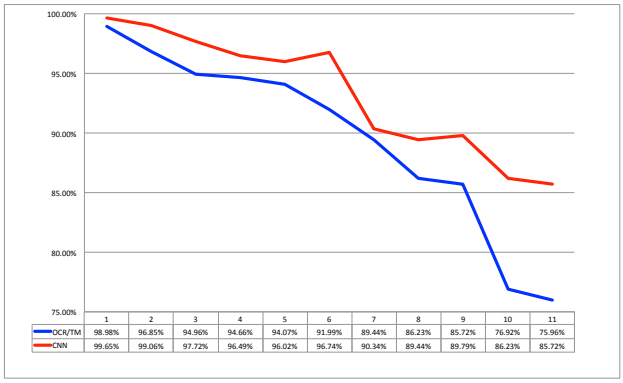

Fig. 10. Accuracy rate of individual character

lowest accuracy rate is 75.96% in TM/OCR and 85.72% in CNN.

V. Conclusions

Defeating the CAPTCHAs is the most effective way to increase its own safety by finding the deficiency. Regarding the defeating performance, in ideal condition, if we could use OCR directly, the effect wouldn't be bad. While the situation is much more complicated, CNN acts as a more robust and useful method. All in all, CNN cannot perform well without good training data and appropriate training structure. Collecting the data in first early period is quite painstaking but worth. This paper is a contribution towards defeating the systematic CAPTCHAs, manually collecting approximately 300,000 training data, evaluating various methods into practice, and identifying the optimal algorithms that we have achieved state-of-the-art performance in this datasets.

Acknowledgement

I want to take this chance to thank my advisor—-Dr Mi Lu. In the process of composing this paper, she gives me many academic and constructive advices, and helps me to correct my paper. At the same time, I would like to appreciate Dr Debin Liu and his research team of CnBizCheck, LLC, who give me a lot of useful literature knowledge and information in this paper.

References

[1] 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*. ACM, 2011, pp. 125–138.

[2] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Building segmentation based human-friendly human interaction proofs (hips)," in *Human Interactive Proofs*. Springer, 2005, pp. 1–26.

[3] X. Ling-Zi and Z. Yi-Chun, "A case study of text-based captcha attacks," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2012 International Conference on*. IEEE, 2012, pp. 121–124.

[4] S. Perreault and P. Hébert, "Median filtering in constant time," *Image Processing, IEEE Transactions on*, pp. 2389–2394, 2007.

[5] Y. Wang and M. Lu, "A self-adaptive algorithm to defeat text-based captcha," in *2016 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 2016, pp. 720–725.

[6] S.-Y. Huang, Y.-K. Lee, G. Bell, and Z.-h. Ou, "An efficient segmentation algorithm for captchas with line cluttering and character warping," *Multimedia Tools and Applications*, vol. 48, no. 2, pp. 267–289, 2010.

[7] R. Smith, "An overview of the tesseract ocr engine," in *icdar*. IEEE, 2007, pp. 629–633.

[8] J. P. Lewis, "Fast template matching," in *Vision interface*, vol. 95, no. 120123, 1995, pp. 15–19.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE transactions on neural networks*, vol. 8, no. 1, pp. 98–113, 1997.

[11] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, "Captcha: Using hard ai problems for security," in *Advances in Cryptology—EUROCRYPT 2003*. Springer, 2003, pp. 294–311.

[12] N. Roshanbin and J. Miller, "A survey and analysis of current captcha approaches," *Journal of Web Engineering*, vol. 12, no. 1-2, pp. 1–40, 2013.

[13] J. Yan and A. S. El Ahmad, "A low-cost attack on a microsoft captcha," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 543–554.

[14] S.-Y. Huang, Y.-K. Lee, G. Bell, and Z.-h. Ou, "A projection-based segmentation algorithm for breaking msn and yahoo captchas," in *Proc. of the international Conference of Signal and Image Engineering*. Citeseer, 2008.

[15] P. Liu, J. Shi, L. Wang, and L. Guo, "An efficient ellipse-shaped blobs detection algorithm for breaking facebook captcha," in *Trustworthy Computing and Services*. Springer, 2013, pp. 420–428.

[16] C. C. Stearns and K. Kannappan, "Method for 2-d affine transformation of images," Dec. 12 1995, uS Patent 5,475,803.

[17] H. S. Baird and T. P. Riopka, "Scattertype: a reading captcha resistant to segmentation attack," in *Electronic Imaging 2005*. International Society for Optics and Photonics, 2005, pp. 197–207.

[18] C. A. Glasbey, "An analysis of histogram-based thresholding algorithms," *CVGIP: Graphical models and image processing*, pp. 532–537, 1993.

[19] A. Singh, K. Bacchuwar, and A. Bhasin, "A survey of ocr applications," *International Journal of Machine Learning and Computing*, vol. 2, no. 3, pp. 314–318, 2012.

[20] R. Smith, D. Antonova, and D.-S. Lee, "Adapting the tesseract open source ocr engine for multilingual ocr," in *Proceedings of the International Workshop on Multilingual OCR*. ACM, 2009, p. 1.

[21] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," *arXiv preprint arXiv:1312.6082*, 2013.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[23] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[24] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Synthetic data and artificial neural networks for natural scene text recognition," *arXiv preprint arXiv:1406.2227*, 2014.

[25] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.