Breaking CAPTCHA using Neural Network

Juan De Dios Santos Rivera
Department of Information Technology
Uppsala University
Uppsala, Sweden
Email: Juan_De_Dios.Santos_Rivera.3313@student.uu.se

Sigit Adinugroho
Department of Information Technology
Uppsala University
Uppsala, Sweden
Email: Sigit.Adinugroho.6004@student.uu.se

***Abstract*—CAPTCHAs has been used widely to determine human users from computer ones. In order to secure a system, a CAPTCHA must hold the property of being hard to read by a computer. In this paper we tried to encode a set of simple CAPTCHAs automatically by employing preprocessing and segmentation methods followed by a recognition phase. Experiment results show that a simple multilayer perceptron network with few hidden nodes can recognize a simplified CAPTCHA with very high accuracy. This result might help a CAPTCHA designer to add more complicated noise and features to make the segmentation hard to achieve.**

***Keywords*—CAPTCHA recognition, multilayer perceptron, image analysis**

I. Introduction

Artificial neural networks [1] are highly used in the area of image analysis; these have been used for tasks going from handwriting recognition to computer vision. From these applications and for the flexibility of ANN, we got our motivation to work on a project whose purpose it trying to break the well-known challenge-response authentication, CAPTCHA. A CAPTCHA, which stands for Completely Automated Public Turing Test to Tell Computers and Humans Apart is a test given to users to determine whether the user is a real person or a computer [2]. Breaking a CAPTCHA has been a benchmark for different computer science areas, which span from Computer Security to Artificial Intelligence. For our project, we intend to break a simple version of a CAPTCHA, which consists on a fixed size image where several non-overlapping digits are in a center of it, surrounded by dots and lines (noise). Our plan was to start with a basic and simple case and from there, depending on how our system works, increase the difficulty of the CAPTCHA by increasing the amount of noise, letters and the length of the string. Our contribution to this problem was to investigate the performance of an artificial neural network to solve the recognition problem while finding an optimal architecture and parameters.

A. Related Work

As mentioned in the introduction, the CAPTCHA recognition problem is a benchmark in the field of Artificial Intelligence, as a result, different research has been done before by a large number of people. In the paper done by Chellapilla and Simard[3] they developed a framework of breaking the CAPTCHA consisting of three parts: Image preprocessing and noise removal, segmentation and recognition.

The first of these steps, image preprocessing and noise removal involves both techniques called binarization and erosion[3]. Binarization is a simple conversion from grayscale image into binary image. The most common way of binarization is by introducing a threshold to decide which part of image is background or foreground. All pixels that have grayscale value larger than the threshold will be replaced by value 1 and all others will be replaced by 0 [4].

There are some useful methods to remove noise. One of these is erosion, which is done by moving a structuring element to all pixels and check whether it fits the set [5]. Shrinking is a useful way to remove noise, especially lines. It is done by removing every pixel which has neighbor pixels less than a specified threshold. A pixel is deleted as well if it has no text pixel neighbor. In order to differentiate text pixels from noise and background, some methods like edge detection, thresholding, and fill flooding can be applied [6].

The following step, segmentation, denotes the process of dividing an image into several regions of interest. In case of the CAPTCHA problem, segmentation means chopping the image into pieces that contain single characters. One way to segment a CAPTCHA is by extracting connected component and if the resulted component is too large, it will be split into two or three components [3]. A method implemented by Yan and Ahmad, states to segment the image vertically by utilizing the histogram of the image. An image will be broken into several components based on the location of gaps in the histogram and each component will be given a different label by using a color-filling method [7].

Another approach to segment images is by using vertical and horizontal projection. Vertical and horizontal projections are used to find left, right, top, and bottom border of a segment. This method fails to detect two or more overlapping characters, since they are segmented as one segment. To overcome this problem, a feedback method is involved to separate those characters after segmentation. Based on the result of segmentation, a standard width of a character is calculated as the average width of characters multiplied by 1.5. If the width of a segment is greater than the standard width, it will be segmented again until its weight is lower than the standard one [8].

A clustering method can also be implemented to segment a CAPTCHA image. By using K-means clustering, centroids are spread equally to the width of the images. The number of centroids are equal to the number of characters in a CAPTCHA. This method works properly if the characters in the image are spaced proportionally [9]. There are some drawbacks of K-Means segmentation. It is unstable and sometimes it needs to be run several times to produce a satisfactory result. Also,

several characters such as M,G,i,l often deceive the segmenter [6].

The segmentation part is considered to be the most difficult and resource consuming step during overall process since the CAPTCHA is designed to resist segmentation [10], [7].

Prior to recognition phase, several features that represent the problem need to be extracted. Chandavale et al used the following four features [11]:

1) Number of holes. Some characters have 1 hole for instance a,b,d,o.
2) Height of characters. Each character is categorized as small or big by a threshold of its height.
3) Maximum number of white-black transitions in the vertical direction.
4) Relative position of the vertical stroke feature that cross a hole in a character.

A simpler feature was introduced by Zhang and Wang by using 4x4 grid that represents the class of a character. Each cell in the grid contains the number of zero value pixels, which denote object, of a character that cross the cell. The 16 attributes determines the class of a character C=0..9,A..Z [8].

Recognition is the last step of the whole task, which can be done easily by using a classification model. A typical way to work with this problem is by using a variation of neural network, called convolutional neural network or CNN [7]. A CNN is a feed-forward neural network focused on extracting the relevant features of the input data by creating local connectivity patterns between neurons of adjacent layers and restricting each layer to the same parametrization.

A previous work by Yan and Ahmad shows that a convolutional neural network can recognize single character from CAPTCHA with high accuracy, even beat the accuracy of a human being [7]. A training set for the network can be populated automatically by generating image characters using several different fonts, from 0 - 9 and A - Z. In order to recognize rotated characters, each character has 12 rotational copy, ranging from -30 to 30 with 5 degree step. Then each character is given label $0 \ldots 35$ [12].

The paper by Bouchain [13] focuses on the difference in performance of a multilayer perceptron against a CNN for character recognition. On his experiment, both networks were tested using a database containing 50,000 hand-writing digits. The CNN showed an error rate of 0.95% for centered and normalized digits on the image and a lower error of 0.8% for distorted versions of the images. On the other hand, a multilayer perceptron architecture with one hidden layer showed an error rate of 4.5%. With two hidden layers, the error rate decreased to 3.05%; for this case, the first hidden layer consisted of 300 neurons and the second hidden layer of 100 neurons. By increasing the numbers of neurons to 1000 for the first hidden layer and 150 for the second hidden layer, an error rate of 2.45% was achieved [13].

In the experiments by Chellapilla et al, 2004 [3], a CNN was used as the recognizer of their problem. 2500 HIPs were used and distributed as follow: 1600 HIPs served as the training set, 200 as the validation set and 200 were used for testing. These HIPS were used for the recognition. 500 additional HIPs were used for testing the segmentation. The CNN employed was trained with these grayscale HIPs where the character was centered.

In a following experiment by Chellapilla et al from 2005 [10], a CNN was also implemented. 110,000 random characters were sampled a distortion and clutter method consisting of rendering the characters with some sort of translation and rotation, adding arcs and applying local and global warps.

The challenge of breaking CAPTCHA, or more generally, the task of character recognition has been applied to the field of Deep Learning. Deep Learning is a field in machine learning that focuses on multi-layered architectures that features a hierarchy of layers that focus on more complex questions or tasks the deeper they are in the network. For example, for the task of determining whether an image shows a car or not, it can be broken into several questions such as ”Does it have a wheel” or ”Does it have four tires?”. Using this same approach, the latest question can be divided into several sub-problems: ”Are the tires divided into sets of two?”, ”Are these sets parallel to each other?”. If all of these questions (in this particular case) are true, then our image probably shows a car.

In the paper by Bastien et al [14], they attempted the problem of handwritten character recognition employing a Deep Learning algorithm. In their experiment, they built a training set made of the MNIST dataset and CAPTCHAs, and reported results of an error rate 17% on tests involving character classes and error rates from 1% to 5% on tests involving just digits.

II. CAPTCHA RECOGNITION ALGORITHM BASED ON A MULTILAYER PERCEPTRON

A. Overview

In the previous parts, we presented various methods and examples of approaches implemented by others to solve the problem of CAPTCHA recognition. Methods that ranges from neural networks, convolutional networks and deep learning. Using some of these ideas, we managed to create our own methodology to attempt the problem.

Our method is divided in three sections - that will be discussed in the following sections. These are: preprocessing of data, segmentation, and recognition. The first and second steps from the sequence make use of some image analysis methods to provide a noise-clean image for the recognition part, which uses neural network to recognize the pattern. Using this sequence of steps, we want to find if it is possible to break a CAPTCHA using multilayer perceptron and along with it, an optimal architecture for the network.

B. Characteristics of the CAPTCHA

In the experiments conducted, a simplified version of CAPTCHA was used, instead of the well-known CAPTCHA seen on sites, such as Outlook.com. Each CAPTCHA is made of three random digits, which use the same font type, ranging from 0 to 2. These digits are not warped, in other words, the shape of the number is not modified. The digits are placed randomly with arbitrary rotational angle from -30 to 30 degree.

The size of each digit is between 40 and 60 measured in point where 1 point = $\frac{1}{72}$ inch.

Besides the digits, each one of the CAPTCHAs is also composed by two different type of noise. The first noise is based on circular dots with random size in the range of 0 to 5 that are spread randomly over the image. The other type of noise is a 0.5 point width straight line. The lines are laid randomly on the image with arbitrary length and rotational angle.

Both digits and noises are merged and saved as Portable Network Graphics (PNG) file. The size of the image is 253 x 152 pixels with 75 Dots per Inch (DPI) resolution. Under low resolution image, the digits will have rippled edge and the big size dots will appear as plus sign.

Fig. 1. An example of simplified CAPTCHA

C. Data Preprocessing

The main purpose of preprocessing is to remove the noise from the images. The processed images will be passed to the segmentation process that assume the input is noise free. In the first step of preprocessing, the image will be converted to binary format, which has only 0 and 1 value where 0 denotes black or background and 1 denotes white or foreground. Although all CAPTCHAs used in this paper have black and white format, they are stored in RGB format where each pixel has red, green, and blue channel. Each channel has a value ranging from 0 to 255. A pixel will have value of 1 if its original luminance is greater than a threshold which is computed using Otsu algorithm [15].

A morphological image processing, called dilation, is employed at the first stage of noise removal. The operation fills a pixel x from a background X if the origin of the structuring element SE = x. A structuring element is a set of pixels in certain shape to probe an object under study [5]. This operation may differ from common image processing which uses erosion to remove noise since in this experiment the digits have black color and that means background in MATLAB. The size of the structuring element must be considered carefully because large structuring element will erase large amount of noise easily, as well as remove several part of the digits. On the other hand, a small structuring element will erase noise less efficiently but preserving the digits better.

After the dilation operation, thin lines and small dots should be completely removed, while the size of large dots should be reduced. In order to remove remaining small dots, the median filter is used. This filter replaces the value of a pixel by the median of the value of pixels in its neighborhood.

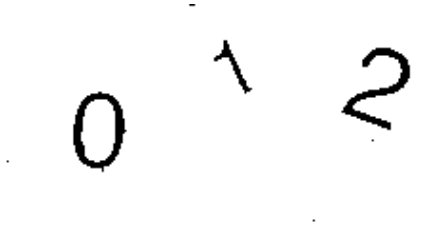

Fig. 2. A CAPTCHA after dilation

The size and shape of the neighborhood area are defined by user [5]. The bigger the neighborhood area, the bigger noise can be removed. However, large neighborhood also distort the main object. Median filter implementation in MATLAB has a side effect where pixels in the corner of the image will be replaced by black because the borders of the images are padded by 0s [15]. This problem can be fixed easily by applying imclearborder function to the complement of preprocessed image.

D. Segmentation

Segmentation is the process of dividing an image into several region of interest. For the experiment, we employed segmentation to break up the image into three regions where each one contains a single digit. In order to do that, each connected component in the image is labelled with a different integer value, where 0 denotes the background, 1 denotes the leftmost digit, 2 denotes the middle digit, and 3 denotes the rightmost digit. Based on those labels, a bounding box for each digit can be derived to produce sub images that contain every digit. This procedure always assume that input image is free from any noise. In the worst case scenario, where the preprocessing cannot completely remove part of the noise, the segmentation will produce more than four connected components. There is also the possibility that some of the digits are overlapping. In this case the number of connected components are fewer than four. For both cases where segmentation produces other than four components, no further action is taken and decoding for a CAPTCHA fails.Once the segmentation process is completed, the final images will be resized to 41 x 27 and flattened to produce a single dimension array containing 1107 elements.

Fig. 3. A success segmentation. Different colors illustrates different connected component

Fig. 4. Segmentation fails to separate two overlapping digits

E. Network Structure and Training Algorithm

For the experiment, a multilayer perceptron network was employed for the recognition of the images. It features 1107 input nodes, a single layer of hidden nodes and three output nodes, corresponding to the number of digits. Since one of the purpose of our experiment is to obtain the most optimal configuration for the network, we do not have a specific number of hidden nodes. Instead, we will conduct several tests using a different number of hidden nodes. The training algorithm used was batch resilient backpropagation (RPROP) and the error function to minimize was the mean square error (MSE).

III. Experiments and Analysis

A. Experiment dataset and parameters

The network was trained over a training set of 39 non-noisy images of the digits 0, 1, 2. The digits, follow the same characteristics as the CAPTCHAs discussed in the above section. All the digits are rotated from -30 degrees to 30 degrees, same font and non-warped.

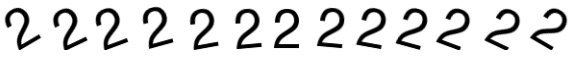

Fig. 5. A training image for digit 2

For the testing part, we built a dataset of 1000 CAPTCHAs with the characteristics described in the above section. Then they were sent to the network one by one, producing an output consisting of three numbers. This result from the recognition phase is then compared with the predefined label to measure the accuracy.

In order to figure out the most appropriate number of hidden nodes, our program was run with 2 hidden nodes and the number of hidden nodes was increased iteratively to find as low error as possible.

B. Analysis of results

During the preprocessing and segmentation, we identified 16 images that were not segmented properly, i.e producing fewer or more than 4 connected components. This is caused by remaining noise after noise removal and overlapping digits that cannot be segmented by the segmentation algorithm. Because the preprocessing and segmentation are deterministic process, their error rates are equal for every run.

TABLE I. Testing Accuracy

Number of Hidden Nodes	Accuracy
2	58.2%
3	98.1%
4	97.4%
5	98.4%
6	98.4%
7	98.4%
8	98.4%
9	98.4%
10	98.4%

Table I shows correlation between the number of hidden nodes and testing accuracy. After using 5 hidden nodes, the accuracy was converged to 98.4%. This means the neural network can recognized all patterns correctly since the 1.6% error was contributed by the preprocessing and segmentation error.

IV. Conclusion

From the entire experiment, it was shown that our version of CAPTCHAs can be recognized easily by using neural network approach. However, better preprocessing and segmentation algorithms should be used since most error came from those process. Watershed algorithm is a promising segmentation method to try since it has a capability to draw separation lines between two overlapping objects. Even with that, since our CAPTCHAs were very simple, most of the problem was solve during the preprocessing phase, turning the problem into a trivial one for the neural network. For further testing and in order to understand whether neural network can be used to recognize more complicated CAPTCHA, a noisy and warped digits should be tested to a network which was trained using clean and non-warped digits.

References

[1] A. P. Engelbrecht, *Computational Intelligence: An Introduction*, 2nd ed. Wiley Publishing, 2007.

[2] L. von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Commun. ACM*, vol. 47, no. 2, pp. 56–60, Feb. 2004. [Online]. Available: http://doi.acm.org/10.1145/966389.966390

[3] K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (hips)." in *NIPS*, 2004. [Online]. Available: http://dblp.uni-trier.de/db/conf/nips/nips2004.htmlChellapillaS04

[4] Z. Chi and K. Wong, "A two-stage binarization approach for document images," in *Intelligent Multimedia, Video and Speech Processing, 2001. Proceedings of 2001 International Symposium on*, 2001, pp. 275–278.

[5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing (3rd Edition)*, 3rd ed. Prentice Hall, 2009.

[6] A. Hindle, M. Godfrey, and R. Holt, "Reverse engineering captchas," in *Reverse Engineering, 2008. WCRE '08. 15th Working Conference on*, Oct 2008, pp. 59–68.

[7] 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*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 543–554. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455839

[8] J. Zhang and X. Wang, "Breaking internet banking captcha based on instance learning," in *Computational Intelligence and Design (ISCID), 2010 International Symposium on*, vol. 1, Oct 2010, pp. 39–43.

[9] A. Caine and U. Hengartner, "The ai hardness of captchas does not imply robust network security," in *Trust Management*, ser. IFIP International Federation for Information Processing, S. Etalle and S. Marsh, Eds., vol. 238. Springer US, 2007, pp. 367–382. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-73655-6_24

[10] K. Chellapilla, K. Larson, P. Y. Simard, and M. Czerwinski, "Computers beat humans at single character recognition in reading based human interaction proofs (hips)," in *CEAS'05*, 2005, pp. –1–1.

[11] A. Chandavale, A. Sapkal, and R. M. Jalnekar, "Algorithm to break visual captcha," in *Emerging Trends in Engineering and Technology (ICETET), 2009 2nd International Conference on*, Dec 2009, pp. 258–262.

[12] G. Lv, "Recognition of multi-fontstyle characters based on convolutional neural network," in *Computational Intelligence and Design (ISCID), 2011 Fourth International Symposium on*, vol. 2, Oct 2011, pp. 223–225.

[13] D. Bouchain, "Character recognition using convolutional neural networks," *Institute for Neural Information Processing*, vol. 2007, 2006.

[14] F. Bastien, Y. Bengio, A. Bergeron, N. Boulanger-Lewandowski, T. M. Breuel, Y. Chherawala, M. Ciss, M. Ct, D. Erhan, J. Eustache, X. Glorot, X. Muller, S. P. Lebeuf, R. Pascanu, S. Rifai, F. Savard, and G. Sicard, "Deep self-taught learning for handwritten character recognition," *CoRR*, vol. abs/1009.3589, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1009.htmlabs-1009-3589

[15] MathWorks, *Image Processing Toolbox Reference*. Natick, Massachusetts: The MathWorks Inc., 2014.