SEMI-SUPERVISED DEEP LEARNING APPROACH FOR BREAKING GEOCACHING CAPTCHAS

Ondrej Bostik
Doctoral Degree Programme (4), FEEC BUT
E-mail: bostik@feec.vutbr.cz

Supervised by: Karel Horak
E-mail: horak@feec.vutbr.cz

Abstract:

For nearly two decades, a substantial part of developed anti-abuse and anti-spam systems for web applications called CAPTCHA is based on imperfections in OCR (Optical Character Recognition) algorithms. But with improvements in Deep Learning in OCR, these systems are now obsolete. More and more systems can now break various text Captchas with great accuracy. Now with sufficient training dataset, almost every text-based Captcha scheme can be broken.

The focus of this work is to present an idea of a semi-supervised method for reading text-based Captcha which needs only a small initial dataset. The main part of this article is dealing with the problem of training a deep learning system with only a small sample of target Captcha scheme via transfer learning.

Keywords: OCR, CAPTCHA, Deep learning, semi-supervised learning, MATLAB

1 INTRODUCTION

Common most used approach to Captcha (Completely Automated Public Turing Test to tell Computers and Humans Apart) implementation for web services is based on OCR (Optical Character Recognition) problem. Current OCR algorithms can be very robust, but they have some weaknesses. This imperfection limits the usage of these algorithms but can be utilized for Captcha purposes with great advantage. The server sends an image with a sequence of characters to the client-side. This image is prepared in a way that uses known OCR issues against the artificial solver (computer). At the same time, as the Captchas become more and more robust, people who try algorithmically solve this kind of Captcha challenge helps to improve the OCR algorithm [1].

This kind an iteration process helps both sides, but development advanced so far, that current Captcha schemes are very complex for humans and the computers have a significantly higher success rate than humans. Many current Captcha challenges are so complicated, that humans cannot solve them, but machines can. Automated versatile systems for cracking Captcha can beat many schemes without any kind of human interaction. Some of these systems can be tweaked to learn new unknown Captcha challenge. As previous research has shown, this kind of system can overcome almost any possible Captcha scheme with a high success rate [2, 3, 4].

Most recent attacks use convolutional neural networks (CNN) in combination with other techniques. For example in [5] Gao et al utilized CNNs in the first phase for feature extraction and Long short-term memory for actual recognition. Another similar work was presented in [6], where 2 deep nets were used, one estimating the length of the text, the other using this information to get the supposed correct answer. The disadvantage in both works are in the need to build a large annotated dataset for initial system learning.

This disadvantage is bypassed by the system in [7]. The system use a small annotated sample (about 500pcs) to learn the generator of synthetic Captcha codes of the same style. The generator is based on a Generative adversarial network. The generated data is then used to train the basic version of CNN and the original data is used to fine-tune the network.

2 PROPOSED SOLUTION

The main idea of this paper is to use the target web-page for generating sufficient enough dataset without annotation a huge number of data. The process benefits from the fact, that the target web-page uses the Captcha to validate the input data. The target web-page needs to generate a Captcha image for every request and validate data received from the user. The attacker can easily make an autonomous system, which downloads the Captcha image, tries to guess the answer (with a random answer or with a Captcha breaking system) and send it to the server. The attacker is automatically notified by the server, whether the sent answer is correct or not. With the use of distributed attacks and the great number of attempts, the attacker can obtain a great number of testing samples without annotating a single Captcha image.

In this paper, the method will be tested on the annotated dataset, rather than the online test to lower the network traffic. To speed the initial phase up, the small sample of 30 images was used to pre-train the classification network as a starting point for this experiment.

2.1 INPUT DATASET

The dataset used in this experiment was created by downloading 16731 Captcha images from the page `http://geocheck.org` via Selenium Webdriver. This Captcha scheme used on this page consists only of numbers of the constant length of 5 digits. The implementation has a great weakness. The verification process is located at the client-side, e.g. the correct answer is present on the web page send to a user computer. The only security measure used is insecure MD5 hash. The automated Python script can download the Captcha image and break MD5 hash to obtain the right answer in less than 0.25s. This makes this Captcha scheme perfect for rapid dataset creation. The sample image from the dataset is presented in figure 1.

Figure 1: Sample of input dataset used for the experiments

2.2 IMPLEMENTATION

The entire experiment was done in MATLAB computational environment using Deep Learning Toolbox and Computer Vision Toolbox. Experiment setup and data flow is described on figure 2.

In the first stage, the 30 Captcha images was labeled via *Image Labeler* to create precise annotation of character location. Extracted regions was then used to train segmentation deep neural network [8] with layers shown on table 1. Training used algorithm called stochastic gradient descent with momentum [9], initial learning rate set to 0.001 and mini batch size of 10 for total number of 100 epochs. The class weights were set according the ratio of the classes in training data.

With this segmentation network, the initial batch of learning images was created. This RGB image contains one letter each and each was resized to 28x28px size and saved to disk with the correct label.

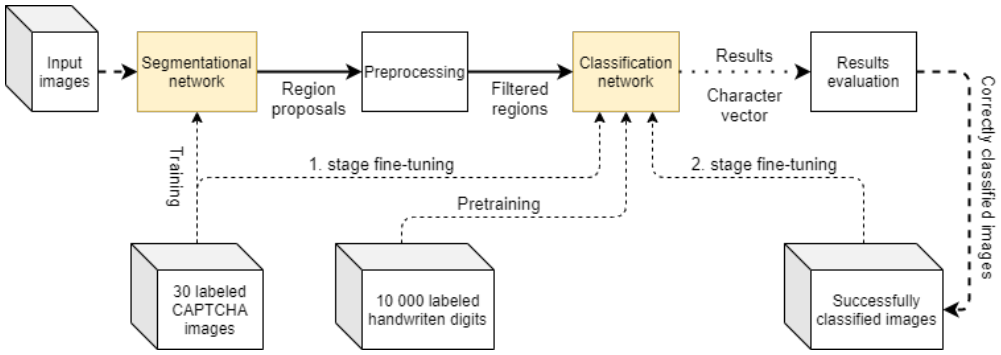

Figure 2: Experiment setup diagram

	Layer type	Layer description
1	Image Input	60x213x3 images with 'zerocenter' normalization
2	Convolution	32pcs 3x3x3 convolutions with stride 1 and padding 1
3	ReLU	ReLU
4	Max Pooling	2x2 max pooling with stride 2 and padding 0
5	Convolution	32pcs 3x3x32 convolutions with stride 1 and padding 1
6	ReLU	ReLU
7	Max Pooling	2x2 max pooling with stride 2 and padding 0
8	Transposed Convolution	32pcs 4x4x32 transposed convolutions with stride 2 and cropping 1
9	ReLU	ReLU
10	Transposed Convolution	32pcs 4x4x32 transposed convolutions with stride 2 and cropping 1
11	ReLU	ReLU
12	Convolution	2pcs 1x1x32 convolutions with stride 1 and padding 0
13	Softmax	Soft-max
14	Pixel Classification Layer	Class weighted cross-entropy loss with 2 classes

Table 1: Segmentation deep network used to classify pixels between background and foreground

The classification network was pre-train on 10000 handwritten digits dataset (1000 per digit) from Matlab demo datasets, 7500 images were used for training, 2500 for validation. The original dataset contains binary images, but for the test purposes, random color for character and background was selected and every image was colored. The structure of the network is shown in table 2. Training also uses an algorithm called stochastic gradient descent with momentum [9], an initial learning rate set to 0.01 for the total number of 4 epochs. This pre-trained network was saved as a starting point for further fine-tuning.

The pre-trained classification network was then fine-tuned using a labeled segment from the segmentation network. During the first stage of fine-tuning, 100 segments (10 per digit) were used for training, 50 for validation. The training uses the same parameters, but the number of epochs was set to 20 to make more emphasis on real data.

With both deep neural networks learned, the whole system was tested on the dataset of 16731 images. As can be seen in figure 2, every input image was segmented using the first network, preprocessed and every segment was classified. The resulted digits were compared with the ground truth. To make proper simulation of server response, the simulation only outputs true or false for the entire response, not for each digit. The correctly answered Captchas are then used to create a new dataset for next stage fine-tuning. The classification network fine-tuned on the new data and the process is repeated.

	Layer type	Layer description
1	Image Input	28x28x3 images with 'zerocenter' normalization
2	Convolution	8pcs 3x3x3 convolutions with stride 1 and padding 'same'
3	Batch Normalization	Batch normalization with 8 channels
4	ReLU	ReLU
5	Max Pooling	2x2 max pooling with stride 2 and padding 0
6	Convolution	16pcs 3x3x8 convolutions with stride 1 and padding 'same'
7	Batch Normalization	Batch normalization with 16 channels
8	ReLU	ReLU
9	Max Pooling	2x2 max pooling with stride 2 and padding 1
10	Convolution	32pcs 3x3x16 convolutions with stride 1 and padding 'same'
11	Batch Normalization	Batch normalization with 32 channels
12	ReLU	ReLU
13	Fully Connected	10 fully connected layer
14	Softmax	Soft-max
15	Classification Output	crossentropyex with '0' and 9 other classes

Table 2: Classification deep network used to translate image regions into corresponding characters

3 RESULTS OF EXPERIMENT

All experiments was realised on a laptop computer with Intel Core i5-6300HQ with 4 cores and 2.3GHz frequency. The graphic card used was NVIDIA GeForce GTX 950M. The segmentation network was trained on 30 images for 300 iterations divided into 100 epochs. The entire training time was 1 minute and 50 seconds and resulted accuracy was 94.61%.

Training of the classification net was done in iterations. The statistics of the whole process is in table 3. In the pre-training phase, the dataset was trained on a different dataset to prepare the network working on similar image data. Then the pre-trained network was saved and then the learning continues on a tiny sample of annotated data (Iteration 1). Then the correctly classified images were used to enlarge the training data and were fine-tuned the network again twice. It is depicted in the table 3, that the number of epochs was updated in every iteration to prevent over-learning.

Iteration	Training count	Test count	Test accuracy	Total epochs	Total time
Pre-training	7 500	2 500	92.16	4	1m 26s
Iteration 1	100	50	78.00	20	5s
Iteration 2	1 190	1380	95.93	10	35s
Iteration 3	15 530	5 635	99.61	4	2m 55s

Table 3: Training results per iteration

After each training iteration, the validation accuracy was evaluated to see the progress and to enlarge the training dataset. The main parameter in the table 4 is image accuracy. With every phase (and with more train data) the success rate per image rises significantly.

4 CONCLUSION

This paper presents an idea of a semi-supervised method for text-based Captcha which needs only a very small initial dataset. Previous works need thousands of samples to learn the target scheme and break it. The presented experiment presents a semi-supervised method for training the Captcha

Iteration	Image count	Segment accuracy	Image accuracy	Successfully classified images	Total time
Iteration 1	16 731	48.62%	3.07%	514	15m 40s
Iteration 2	16 731	74.58%	25.30%	4 233	20m 21s
Iteration 3	16 731	83.15%	43.71%	7 313	25m 12s

Table 4: Validation results per iteration

breaking algorithm with only 30 annotated images.

The following work will focus on a more general method to break Captcha schemes with more security measures, like a wider character set and various lengths.

ACKNOWLEDGEMENT

The completion of this paper was made possible by the grant No. FEKT-S-20-6205 - "Research in Automation, Cybernetics and Artificial Intelligence within Industry 4.0" financially supported by the Internal science fund of Brno University of Technology.

REFERENCES

[1] K. Kaur and S. Behal, "Designing a Secure Text-based CAPTCHA," in *Procedia Comput. Sci.*, vol. 57, pp. 122–125, Elsevier, 2015.

[2] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The End is Nigh: Generic Solving of Text-based CAPTCHAs," 2014.

[3] H. Gao, J. Yan, F. Cao, Z. Zhang, L. Lei, M. Tang, P. Zhang, X. Zhou, X. Wang, and J. Li, "A Simple Generic Attack on Text Captchas," in *Netw. Distrib. Syst. Secur. Symp. (NDSS 2016)*, 2016.

[4] O. Bostik and J. Klecka, "Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms," in *15th IFAC Conf. Program. Devices Embed. Syst. PDES 2018*, (Ostrava), p. 6, IFAC-PapersOnLine, 2018.

[5] Y. Zi, H. Gao, Z. Cheng, and Y. Liu, "An End-to-End Attack on Text CAPTCHAs," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 753–766, 2020.

[6] M. Tang, H. Gao, Y. Zhang, Y. Liu, P. Zhang, and P. Wang, "Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, pp. 2522–2537, oct 2018.

[7] G. Ye, Z. Tang, D. Fang, Z. Zhu, Y. Feng, P. Xu, X. Chen, and Z. Wang, "Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach," in *Proc. 2018 ACM SIGSAC Conf. Comput. Commun. Secur.*, CCS '18, (New York, NY, USA), pp. 332–348, Association for Computing Machinery, 2018.

[8] K. Horak and R. Sablatnig, "Deep learning concepts and datasets for image recognition: overview 2019," No. March 2016, p. 100, 2019.

[9] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, vol. 27. Cambridge, MA: The MIT Press, 1. edition ed., 2012.