ISSN: 1803-3814

EVALUATION OF RANDOMLY GENERATED FONTS FOR BUBBLE CAPTCHA

Ondrej Bostik, Karel Horak, Jan Klecka

Brno University of Technology
Department of Control and Instrumentation
Technicka 12, 61600 Brno
Czech Republic
bostik@feec.vutbr.cz

Abstract: *A Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA), is the wide-spread concept of systems suited to secure the web services from automated SPAM scripts. The most common CAPTCHA systems benefit from imperfections of Optical Character Recognition algorithms.*

This paper presents our ongoing work focused on the development of a new CAPTCHA scheme based on a human perception. The goal of this work is to evaluate the usability of randomly generated fonts used in Bubble Captcha scheme with both humans and OCR classifiers.

Keywords: *OCR, CAPTCHA, Neural Networks, k-NN, Decision trees, SVM, Bubble Captcha*

1 Introduction

Widely spread CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) systems are the important part of almost every web service. The reason was to utilize a simple technique which differentiates the human user from the automated system and collect data only from humans. This way was meant to guarantee only relevant inputs to the public online pools, e-shops discussion forums.

CAPTCHA is defined as a general task that must be very easy for the humans to solve, but it must be enormously difficult to create an autonomous machine to solve the task both for the computing resources and for the algorithm complexity [1].

The goal of this work is to assess the usability of randomly generated font consist of bubbles to form abstract characters previously presented in [5] and later improved in [4]. We compare two main aspects. The first one is the human ability to recognize the characters and the second one is the success rate of machine learning algorithms on this kind of letters.

1.1 Text-based CAPTCHA

Common most used approach to Captcha implementation for web services is based on OCR (Optical Character Recognition) problem. Current OCR algorithms can be very robust, but they have some weakness. This imperfection limits the usage of this algorithms but can be utilized for Captcha purposes with great advantage. The server sends an image (as can be seen in Fig. 1) with a sequence of characters to the client side. This image is prepared in the way that uses known OCR issues against the computers. At the same time, people who try algorithmically solve this kind of Captcha challenges helps to improve OCR algorithms ([12]).

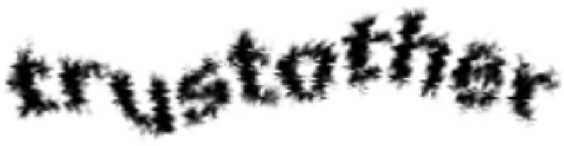

Figure 1: Wikipedia Captcha, example of ordinary Captcha scheme, (taken from [8])

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 shown us in [8], this kind of system can overcome almost any possible Captcha scheme with high success rate.

Previous study [8] recommend several techniques to improve the security of Captcha schemes. The first to consider is to utilize some kind of anti-segmentation technique. Many systems use lines crossing the letters, but the common mistake is to use long lines (longer than the size of a letter), that can be filtered with Hough transformation. The most straightforward technique is to use variable keyword length, that makes more difficult to guess the position of individual letters.

Using complex background is not suggested, because it does not help distinguish man from machine [8]. If letters are properly hidden in the background, humans, and machines both need to overcome the same difficulties.

The next stage of security is at the level of single characters. It is good practice to apply characters of different fonts types, sizes, and rotations. On the other hand, it is not recommended to use random noise because the current algorithms are better suited to handle the noise than human brains. A large set of characters is not recommended too because for example problems with the distinction of number 0, letter O and big D is common for computers and humans [8].

A very interesting idea called reCaptcha was described in [1]. The further development and improvements were later held in Google. The rough estimations published in [1] indicated about 100 million Captcha challenges solved every day with various time from 5 to 20 seconds. This leads us to a situation when humanity wasted dozens of years every day solving Captcha schemes.

Professor Ahn and his team ask a question, how to utilize this time to help humanity. The solution is simple. Archives contain a great number of documents which are not digitalized and so not available for further processing. Original system reCaptcha (see fig. 2) consist of two parts. The preparation stage utilizes two OCR algorithms, that tries to transcribe submitted document independently. Outputs are then compared. Matched parts are then marked as correctly solved. Any disagreement in outputs is used to create Captcha challenge [1].

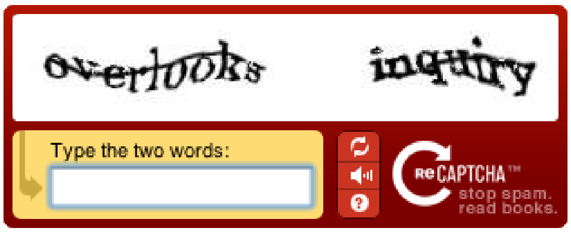

Figure 2: Sample of Google reCaptcha text scheme, (taken from [9])

Every reCaptcha scheme consists of two separate words. One of them is the word with uncertain meaning. The second one is control word which was previously successfully solved in the reCaptcha system, but on which one or both OCR algorithm failed. Both words are enhanced with some anti-segmentation techniques and randomly arranged in the image [1].

As can be anticipated, using a scanned document as input helps human to overcome reCaptcha challenge. The reason is obvious, scanned documents are mainly written in English and use the English word. Human brain then significantly speed up the process and helps to distinguish the letters based on the meaning.

Result evaluation is then based primarily on the successful transcription of the control word. If the answer to the known word is correct, then we can expect the second word is also transcribed correctly. The value of control word is determined by the virtual pool between the two OCR algorithm and answers submitted by users. Human transcription is valued as one vote, the output of OCR algorithm is count as half a vote. The correct value needs at least 2.5 votes to be declared as successfully recognized. This word is then added to the control set [1].

Results from the authors of system state that this OCR system works with a success rate of 99.1%, which is very close to the result achieved by transcription from two human professionals. To compare, standard OCR programs can gain 83.5% success rate [1]. The final system also produce correctly transcribed documents as a bi-product.

2 Bubble Captcha concept

In previous work [5] we developed elementary bi-color Captcha scheme with randomly positioned circles/bubbles forming the font. The main reason was to prepare the framework to generate galleries to test and assess weaknesses of current OCR algorithms and compare results with success rate of other people. The completed system will be also used to generate single Captcha challenge for security purposes on web pages.

2.1 Initial work

A two-dimension array representing binary grid for every character used is one of the key elements entering the algorithm. Generative algorithm randomly selects characters from used character set and position them bubble by bubble to generate our Captcha scheme alongside with bubbles of different color. Every bubble is randomly positioned approximately to its correct position in the grid and randomly scaled in size within predefined limits. The three different variants (denoted as *a*, *b* and *c* hereinafter) of Bubble Captcha are generated in this way as an input dataset of the supervised machine learning algorithms [5].

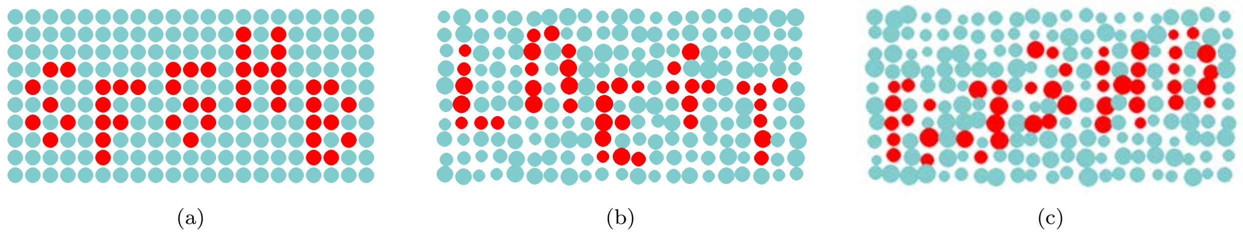

Figure 3: Initial Bubble Captcha variants overview with (a) no displacement - answer 6F5HB (b) medium displacement - answer LNE4T (c) big displacement - answer RJ3HN (from our previous work [5])

Implemented Captcha scheme is shown in the Fig. 3 displaying 3 main levels of randomness in the picture. Differences between the three variants are the rate of randomness in bubble diameters and in bubble placements. The simplest variant (a) presented in Fig. 3(a) contains every bubble in precise centers of the grid with uniform diameters for each bubble.

Variant (b) and (c) presented on Fig. 3(b) and 3(c) respectively contains various rate of randomness in positioning and bubble diameters. All the parameters of all three variants introduced in [5] are presented in table 1.

Variant $[-]$	Buble diameter $[px]$	Buble variation $[\%]$	Grid dimension $[px]$	Grid variation $[\%]$	Number of letters $[-]$
a	30	0 %	35	0 %	5 - 7
b	30	10 %	35	20 %	5 - 7
c	30	15 %	35	30 %	5 - 7

Table 1: Buble Captcha parameters for each variant (from our previous work [5])

2.2 Randomly generated fonts

In security analysis of text-based Captchas [8] was presented, that one of the key points to strengthen the Captcha scheme as a whole is to use multiple fonts. In our previous research [4] we presented the algorithm that further extends this idea.

Initial Bubble Captcha operates with one fixed font, that was inspired by the digital font used on electronic 7-segment displays. Our initial font has several downsizes that lower the usability of the whole Bubble Captcha. For example, humans were hard to differentiate numbers 5 and 6 because of the similarity. On the other hand, despite the variations of bubbles diameters and displacement, the grid was so firm, that several machine learning techniques were able to differentiate the letters with high success rate even with only simple feature extraction (further details in [6]).

Proposed algorithm utilize specific degradation of common computer fonts shown in Fig. 4. The process starts with a blank canvas. The letter of selected font is placed on the canvas and then the canvas is cropped accordingly the utilized area to speed up the process. The image is then tilted, the angle is randomly chosen in the range of ± 18.

The crucial part of the algorithm is to downscale the resolution of the image. The main reason is to further randomize the image with some errors made during downscale and following binary thresholding of the image. This process creates the binary matrix which serves as the main input to the Bubble Captcha PHP generator algorithm.

2.3 Input gallery overview

The main goal of this work is to compare the different fonts, asses their usability and evaluate the whole idea of random fonts generator for Bubble Captcha. To fulfill this goal, the input gallery consists of the three major groups of Bubble Captcha characters - one for every of three used fonts. The first is the initial digital font, the

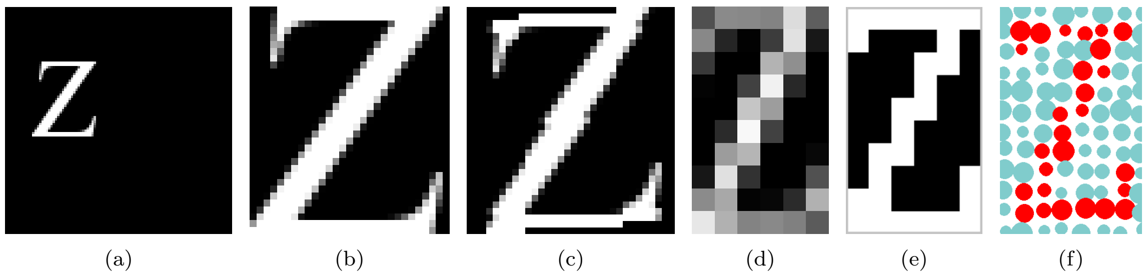

Figure 4: Process of generation new font previously described in [4] (a) initial step with a letter of selected font drawn into blank image (b) cropped image without a blank space (c) tilted image (d) rescaled image to 6x10px (e) binary image (f) resulted Bubble captcha letter

second is Arial-based font and the last is based on Times New Roman. For every font, we generated 5 examples for each combination of 3 level of difficulties presented in table 1 and for every of 33 used characters. That means that for every font we generated 495 samples. In total, we prepared the gallery of 1485 images.

Comparison of all tree used fonts is depicted in Fig. 5.

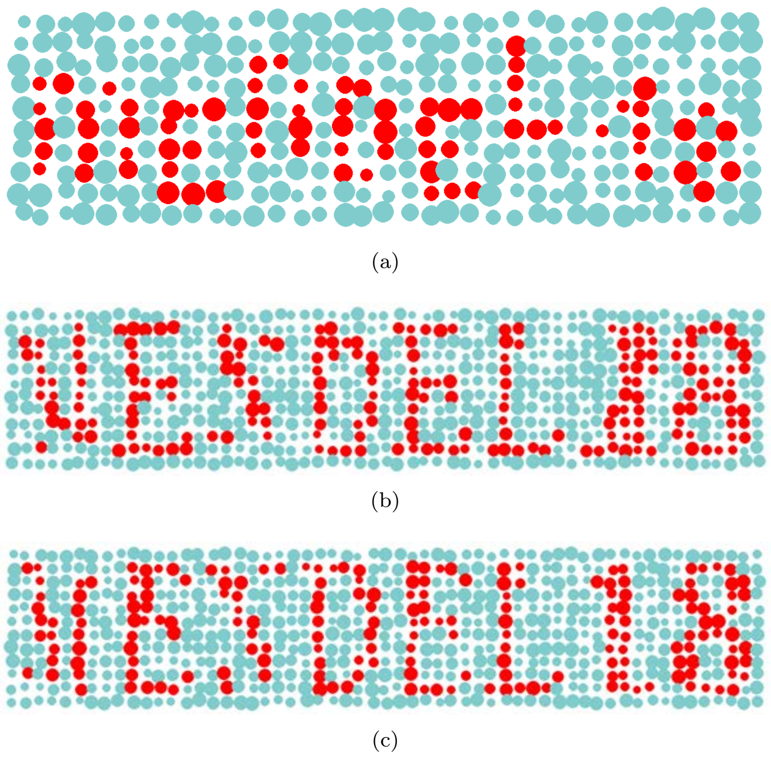

Figure 5: Comparison of various fonts for Bubble Captcha with difficulty level 3 stating "MENDEL218" (a) Initial digital font (b) Arial-based font (c) Times New Roman-based font

3 Experiments

Our experimental work consists of two major parts. The first one is to evaluate the usability of different fonts with human participants. The second part is to employ the machine learning algorithms in a task of classification the letters.

3.1 Evaluation of human characters classification

Our testing platform previously explained in [5] was extended with web annotating tool. The purpose of this platform was to evaluate the success rate of real people over the prepared gallery.

The annotating platform contains raw images of individual characters in a separate folder on the server and also contains true answers of every character in MySQL database table alongside with the information about font type and level of randomness used to generate the font. With every user response, a new entry is stored in the separate table with the information of provided user nickname, the user response and link to the entry in the first table. The report is then sent to the user with the information of correctness of the answer.

Annotating platform chooses randomly from unannotated files. When all of the files are answered once, the application starts over and select from all files. This lead to the even distribution of annotated characters for each level of randomness and font type during the whole testing.

For a common person, every answer took about 3-5 seconds from loading the page to send out the result. We employ 11 people in total, the success rate of individuals was from 70% to 89%.

First evaluated criterion was the success rate of human based on the level of randomness in presented Bubble Captcha characters. The results are shown in table 2. Different levels of randomness display similar success rate. The maximum difference is only 2.6%.

Table 2: Results of human trial for each level of difficulty

Level of randomness	Total answers	Correct answers	Incorrect answers	Success rate
Level 1	1172	1009	163	86.09%
Level 2	1200	1035	165	86.25%
Level 3	1155	966	189	83.64%
Total	3527	3010	517	85.34%

The second parameter to evaluate was person success rate based on the type of font of the presented character. The differences in success rates are based on the font type are shown in table 3. Initial digital font shows a very good success rate of 93.10%. The reason is that the font was carefully designed by humans and users with little training get used to it. On the other hand, computer-generated fonts show some imperfections which slightly limits the usability. The low success rate of 80.71% and 82.07% are based on problems with generating the characters such as M, V, U, I and 1. The main reason is that during the generating of the font the Bubble Captcha character is normalized to the dimension of 6x10 bubbles and some of the characters are degraded beyond the readability limit.

Table 3: Results of human trial for each font type

Font type	Total answers	Correct answers	Incorrect answers	Success rate
Initial digital font	1189	1107	82	93.10%
Arial-based font	1161	937	224	80.71%
Times New Roman-based font	1177	966	211	82.07%
Total	3527	3010	517	85.34%

For the further font evaluation, in table 4 we present the most misclassified characters for every difficulty level, type of font and the total statistics with their success rates. The table reflects our previous conclusion about some letters, that are poorly generated. The table reflects some frequently misclassified letters, like letter M. This letter M drawn with the Arial-based font was never correctly transcribed (red highlighted value).

Table 4: The most misclassified characters with their success rate

Type	Letter 1	Letter 2	Letter 3	Letter 4	Letter 5
Total statistics	V (34.69%)	M (39.81%)	I (55.45%)	1 (56.07%)	U (62.26%)
Difficulty level 1	V (31.25%)	M (42.86%)	1 (51.35%)	I (54.05%)	F (61.76%)
Difficulty level 2	V (36.36%)	M (41.18%)	I (53.85%)	U (57.89%)	1 (63.89%)
Difficulty level 3	M (35.29%)	V (36.36%)	1 (52.94%)	I (58.82%)	U (61.76%)
Initial digital font	6 (55.56%)	Z (68.42%)	Q (76.47%)	K (77.50%)	V (84.38%)
Arial-based font	M (0.00%)	V (12.12%)	1 (17.14%)	F (28.57%)	U (41.67%)
Times New Roman-based font	V (9.09%)	I (17.14%)	M (20.00%)	U (45.71%)	1 (48.57%)

3.2 Machine learning algorithm evaluation

The first step of our experiment is feature extraction. We want to concentrate on properties of the used classifiers. From that reason, we utilize only the most simple feature extraction depicted in Fig. 6. The red channel of the image is first converted into a binary image of normalized dimensions 102px by 172px and then rearranged into a linear vector of 17544 values. This feature extraction, the following comparative experiments, and final evaluation is done in MATLAB computing environment with Statistics and Machine Learning Toolbox and Optimization Toolbox.

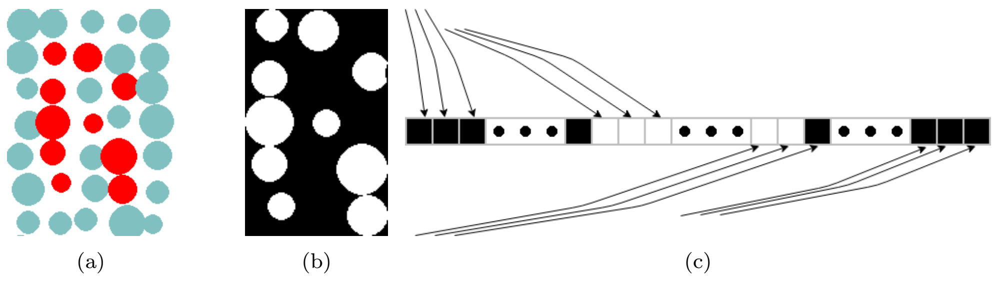

Figure 6: Color based feature extraction (a) original segmented letter (b) binary 2D image 102x172px (c) input vector with lenght of 17544 for machine learning algorithms (from our previous work [6])

The whole input gallery is split into testing and validating subsets in the ratio of 3:2 respectively. In each gallery, there is a uniform representation of all characters, font types and difficulty levels. For all of used machine learning algorithm, we train the classifier firstly on all used fonts and then we learn the classifiers on each of the generated font types independently. For each of this experiment, we evaluate the overall success rate of classifier firstly on all used font types and then on each of the generated fonts. For each of the classifiers, we obtain a 4-by-4 matrix of success rate values representing the ability of knowledge abstraction.

First classification algorithm used was K-Nearest Neighbors [2]. Standard MATLAB setting was used, output was based on exactly one closest element. When more than one stored element has the same distance from the classified item, the algorithm chooses the one with the lowest index (numbers and letters are sorted alphabetically with the numbers before letters). Euclidean distance with no weighting was used as metric for algorithm evaluation. Computation of distances uses an unoptimized exhaustive algorithm, that means all distances are evaluated. No score transformation is utilized.

The result of the experiment is depicted in table 5. When trained on all subsets, the result for individual subsets and all subsets combined are close to 100%. During the evaluation of each individual subset, the results for that subsets are also close to 100%. Other success rates reflect the close similarities of Arial-based font and Times New roman-based fonts. The Arial-based font has also the best results when classifying other fonts, even the Initial digital font.

Table 5: Success rate evaluation of k-Nearest Neighbor algorithm for different validating subsets

Training subset	Success rate for All subsets	Success rate for Initial digital font	Success rate for Arial-based font	Success rate for Times New Roman-based font
All subsets	**99.7%**	99.0%	100.0%	100.0%
Initial digital font	46.6%	**99.0%**	18.2%	22.7%
Arial-based font	65.2%	30.3%	**100.0%**	65.2%
Times New Roman-based font	63.6%	24.2%	66.7%	**100.0%**

The experiment previously described was repeated with the Pattern Recognition Neural Network which is a type of feed-forward neural network [3] implemented in MATLAB computer environment. This neural network consists of an input layer, two hidden layers, and output layer. Input layer with 17544 neurons only propagates signal into the network. Two hidden layers are almost identical - booth utilizes Nguyen-Widrow initialization method, the booth has a transfer function of hyperbolic tangent sigmoid, neurons in booth layers use the summation of weight and biases as an input function. Fist hidden layer consist of 200 neurons and second hidden layer consist of 50 neurons. Output layer has 33 neurons with transfer function Soft Maximum. Remaining parameters of output layer are the same as with the hidden layers. Performance is evaluated based on cross-entropy. The learning of the network was based on Scaled Conjugate Gradient Backpropagation as described in [13]. The maximum number of the epoch was set to 1000 and minimal gradient to 1e-6.).

The result of the experiment for Pattern Recognition Neural Network is shown in table 6. Almost all success rates are lower when compare to k-NN. We believe that this is mainly based on the huge differences between training set for which Neural Network with this kind of feature extraction is not very suitable.

Table 6: Success rate evaluation of Pattern Recognition Neural Network algorithm for different validating subsets

Training subset	Success rate for All subsets	Initial digital font	Arial-based font	Times New Roman-based font
All subsets	**98.8%**	96.5%	100.0%	100.0%
Initial digital font	44.4%	**99.5%**	14.5%	19.2%
Arial-based font	56.6%	14.6%	**100.0%**	55.1%
Times New Roman-based font	57.4%	17.7%	54.5%	**100.0%**

The third method used was Binary Regression Decision Tree created with standard CART algorithm [14] and with Gini's Diversity Index used as the split criterion [7]. The decision tree uses no score transformation. No pruning was applied to the final tree, but all the child nodes with the risk greater or equal to the risk of parent nodes are merged.

The results are depicted in table 7. The results indicate that this classifier is the weakest of all the compared algorithm. Even when training on all subsets the result are below 90%. With this kind of feature extraction, Decision Trees are not suitable classifier for this task.

Table 7: Success rate evaluation of Decision Trees algorithm for different validating subsets

Training subset	Success rate for All subsets	Initial digital font	Arial-based font	Times New Roman-based font
All subsets	**84.5%**	75.3%	88.9%	89.4%
Initial digital font	31.6%	**86.4%**	4.0%	4.5%
Arial-based font	39.6%	4.5%	**85.4%**	28.8%
Times New Roman-based font	36.7%	5.6%	17.2%	**87.4%**

The fourth and the last classification algorithm used in this experiment was Support Vector Machines [10]. The learning phase was based on Error-Correcting Output Codes (ECOC) algorithm [11]. This approach extends the SVM binary classifier to be used for multi-class classification. This particular implementation uses one-to-one coding design, which means that for every combination of two classes within the learning set, one is declared as positive, the second one is declared as negative and the rest of the classes are omitted. The number of resulting binary learners for K classes is $K(K-1)/2$. For evaluation performance of each learner, the binary loss is calculated with Hinge function.

The result for Support Vector Machines is shown in table 8. The relations in between resulting success rates are similar as was in previous three experiments. But the individual scores were slightly higher than was in k-NN. The main advantage of this algorithm is during the classification the unknown font, for which the obtained classifier was not learned. The classifier trained on the initial digital font was about 10% more successful with the two other fonts that in the case of the k-NN classifier.

Table 8: Success rate evaluation of Support Vector Machines algorithm for different validating subsets

Training subset	Success rate for All subsets	Initial digital font	Arial-based font	Times New Roman-based font
All subsets	**98.8%**	96.5%	100.0%	100.0%
Initial digital font	54.7%	**99.5%**	30.3%	34.3%
Arial-based font	65.5%	30.8%	**100.0%**	65.7%
Times New Roman-based font	63.1%	24.7%	64.6%	**100.0%**

4 Conclusion

This work evaluates the usability of randomly generated fonts for our Bubble Captcha scheme. We compare the success rate of character transcription made by a common person in the first experiment. We found out that people are able to recognize character with high success rate. The result for each font type evaluated apart looks promising. On the first look randomly generated fonts has a lower success rate that initial digital font. The reason is based on some letters that random generative algorithm degrade so much that untrained person cannot differentiate them. In the following work, we want to upgrade this algorithm so the generated characters are not degraded beyond the certain level.

On the other hand, machine learning algorithm can differentiate the characters with a high level of success rate. With this feature extraction technique, the best result was achieved by Support Vector Machines algorithm. Support Vector Machine is also the best of these algorithm to work with an unknown font. The k-Nearest Neighbor algorithm and Pattern Recognition Neural Network has only slightly worse success rate that is comparable to the winner. The Decision trees, on the other hand, gained the worst success rates and we do not recommend it for this task without some modification at least with some other feature extraction technique.

In the future work, we want to improve our algorithm for random font generation. We want also utilize some technique which will classify the generated characters before they are utilized in Captcha scheme. We want to improve the security of Captcha algorithm from automated attacks but we also want to check characters if they are readable by a human. Our next plan is to deeply examine all usable features which can be used in OCR problem and propose the measures to suppress them in the way they cannot be used to successfully break the Captcha scheme.

Acknowledgement: The completion of this paper was made possible by the grant No. FEKT-S-17-4234 - Industry 4.0 in automation and cybernetics financially supported by the Internal science fund of Brno University of Technology.

References

[1] von Ahn, L., Maurer, B., Mcmillen, C., Abraham, D., Blum, M.: reCAPTCHA: Human-Based Character Recognition via Web Security Measures. Science (80-.). **321**(12 September 2008), 1465–1468 (2008). DOI 10.1126/science.1160379. URL http://www.ncbi.nlm.nih.gov/pubmed/18703711

[2] Altman, N.S.: An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. Am. Stat. **46**(3), 175–185 (1992). DOI 10.1080/00031305.1992.10475879. URL http://www.jstor.org/stable/2685209

[3] Bishop, C.M.: Pattern recognition and machine learning. Springer (2006). URL https://www.springer.com/gp/book/9780387310732

[4] Bostik, O.: Creating a Random Characters for Captcha Schemes From Existing Fonts. In: V. Novák (ed.) Proc. 24nd Conf. STUDENT EEICT 2018, p. 5. Brno University of Technology, Brno (2018)

[5] Bostik, O., Horak, K., Klecka, J., Davidek, D.: Bubble CAPTCHA - A Start of the New Direction of Text CAPTCHA Scheme Development. In: Mendel, *23*, vol. 23, pp. 57–64. Brno University of Technology (2017). URL http://www.mendel-conference.org/

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

[7] Breiman, L., Friedman, J., Stone, C.J., Olshen, R.: Classification and regression trees. Chapman & Hall (1993). URL https://www.taylorfrancis.com/books/9781351460491

[8] Bursztein, E., Martin, M., Mitchell, J.C.: Text-based CAPTCHA strengths and weaknesses. Proc. 18th ACM Conf. Comput. Commun. Secur. **2011**, 125–138 (2011). DOI 10.1145/2046707.2046724. URL https://dl.acm.org/citation.cfm?id=2046724

[9] Carnegie Mellon University: The Official CAPTCHA Site (2010). URL http://www.captcha.net/

[10] Cortes, C., Vapnik, V.N.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). DOI 10.1007/BF00994018. URL http://link.springer.com/10.1007/BF00994018

[11] Dietterich, T.G., Bakiri, G.: Solving Multiclass Learning Problems via Error-Correcting Output Codes. J. Artif. Intell. Res. (1994). URL http://arxiv.org/abs/cs/9501101

[12] Kaur, K., Behal, S.: Designing a Secure Text-based CAPTCHA. In: Procedia Comput. Sci., vol. 57, pp. 122–125. Elsevier (2015). DOI 10.1016/j.procs.2015.07.381

[13] Møller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. NEURAL NETWORKS **6**(4), 525—-533 (1993). URL http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.38.3391

[14] Rokach, L., Maimon, O.: Data Mining with Decision Trees, 2 edn. Series in Machine Perception and Artificial Intelligence. World Scientific (2014). DOI 10.1142/9097. URL http://www.worldscientific.com/worldscibooks/10.1142/9097