Semi-supervised deep learning approach to break common CAPTCHAs

BOŠTÍK, O.; HORÁK, K.; KRATOCHVÍLA, L.; ZEMČÍK, T.; BILÍK, Š.

Neural Computing and Applications
2021, vol. 33, iss. 20, Pages 13333-13343

ISSN 0941-0643

DOI: http://dx.doi.org/10.1007/s00521-021-05957-0

Accepted manuscript

Neural Computing and Applications manuscript No.
(will be inserted by the editor)

Semi-supervised deep learning approach to break common CAPTCHAs

Ondrej Bostik · Karel Horak · Lukas Kratochvila · Tomas Zemcik · Simon Bilik

Received: date / Accepted: date

Abstract Manual data annotation is a time consuming activity. A novel strategy for automatic training of the CAPTCHA breaking system with no manual dataset creation is presented in this paper. We demonstrate the feasibility of the attack against a text-based CAPTCHA scheme utilizing similar network infrastructure used for Denial of Service attacks. The main goal of our research is to present a possible vulnerability in CAPTCHA systems when combining the brute-force attack with transfer learning. The classification step utilizes a simple convolutional neural network with 15 layers. Training stage uses automatically prepared dataset created without any human intervention and transfer learning for fine-tuning the deep neural network classifier. The designed system for breaking text-based CAPTCHAs achieved 80% classification accuracy after 6 fine-tuning steps for a 5 digit text-based CAPTCHA system. The results presented in this paper suggest, that even the simple attack with a large number of attacking computers can be an effective alternative to current CAPTCHA breaking systems.

Keywords CAPTCHA · Semi-supervised learning · Convolutional Neural Networks

1 Introduction

The expansion of web services in recent decades created a demand for an automated system that can automatically interact and process data contained within these systems. Substituting humans with computer programs in monotonous interaction with web services can be beneficial for both sides. Automated services, as described in [8], can make a profit on stock markets, send unsolicited advertisement, and respond to a new condition in a matter of moment without any human interaction.

Sometimes this exploit of public resources can be harmful. This situation leads to the creation of an automatic system to differentiate the human user from a machine. The resulting test was called CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) in [1]. It is defined as a general task that must be very easy for humans to solve, but it must be enormously difficult to create an autonomous system to solve the task both for the computing resources and for the algorithm complexity.

During the last two decades, a constant struggle of improvements between system providers and users introduce various forms of CAPTCHAs in almost every shape possible. And for every shape sooner or later an automated CAPTCHA breaking solution was created (see [4, 13, 14]). A lot of effort from both sides was invested in gaining an advantage over the other side driven by the opportunity to earn money .

No other field was as affected by this kind of evolution and so synonymous with CAPTCHA as the field of OCR (Optical Character Recognition). Text-based CAPTCHA systems were one of the first widely spread forms of CAPTCHA. Current OCR algorithms can be very robust, but they have some weaknesses. This imperfection limits the usage of these algorithms but it 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 CAPTCHA become more and more robust, people who try to algorithmically solve this kind of CAPTCHA challenge help to improve the OCR algorithm, as for example in [20].

This kind an iterative process helps both sides, but development advanced so far, that current CAPTCHA schemes are

O. Bostik
Faculty of Electrical Engineering and Communication
Brno University of Technology
Technicka 12
Brno 61200
Czech Republic E-mail: bostik@feec.vutbr.cz

very complex for humans and the computers have a significantly higher success rate than humans. 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 in [5,18] has shown, this kind of system can overcome almost any possible CAPTCHA scheme with a high success rate.

In some situations, authors even admit their systems can be overcome by automated solvers. Their goal is to discourage most of the generic attacks while maintaining user experience and solving accuracy for humans. In the end, every CAPTCHA systems must be easily solvable for humans by design, as presented in [10].

In this paper, we want to present a new way to overcome CAPTCHA systems. Until now, every CAPTCHA solving system focuses on single computer attacks utilizing one strong solver. This kind of attack can be easily blocked from the service provider by blocking the individual IP address or range of IP addresses. Nevertheless, the attackers can easily hijack a huge amount of computers and distribute the attack over a vast number of computers worldwide. Even a very simple attack based on classifier trained without a huge dataset can be successful (success rate over 1% as presented in [9]). And with every successfully classified CAPTCHA, attackers can automatically expand the training set and augment the classifier accuracy. Even more, the attacker can train multiple classifiers and make them compete between instances to make the most accurate prediction or even better, use these classifiers combined in the style of the AdaBoost method (see [15]).

Having this in mind, we would like to present a new way to train the CAPTCHA solvers with none or at least minimal human interaction. Our goal is not to break any CAPTCHA service in particular, but we would like to understand and describe the process to further improve the security of CAPTCHA services.

2 Related work

The main CAPTCHA security features of all prominent websites rely mostly on anti-segmentation techniques like negative-kerning, occluding lines, character overlapping, and variable length. All these security measures are in place because the standard approach to overcome CAPTCHA for the two decades is to first segment the image into individual regions per assumed character. The following step commonly utilized machine learning to classify the segmented regions as an individual character. This approach is called segment-then-recognize (see [24,12,27]) and can also be used for audio CAPTCHA as presented in [7,6].

The reason for this can be found in [11], where authors demonstrated that machines outperform humans in the task of single character recognition. This work set the base ground for the assumption, that the difficulty of CAPTCHA recognition lies mainly in anti-segmentation techniques and was improved in [9].

Most of the works discussing CAPTCHA solving around the year 2015 are using ad-hoc methods fine-tuned to a particular CAPTCHA scheme. One example can be seen in [29], where the presented approach achieved 82% on reCAPTCHA version 2011 and 95.5% on reCAPTCHA version 2012. This high success rate is due to the precisely tuned algorithm custom fit on these kinds of CAPTCHAs and this approach must be extensively modified to other CAPTCHA schemes of this class.

Similarly, in [32] authors used a simple method utilizing pixel counting combined with some heuristics to overcome CAPTCHAs provided by Captchaservice.org. In [17] authors presented CAPTCHA solving method dealing with hollow CAPTCHAs with a success rate from 36% to 89% on five hollow CAPTCHA schemes, but this approach was not effective with non-hollow CAPTCHA schemes. Two-layer CAPTCHA provided by Microsoft was also overcome in [16] by the custom made attack which relies on splitting the CAPTCHA into two separate layers and solving them independently.

The next stage in the evolution of CAPTCHA solving methods uses toolbox design. One instance of this is Decaptcha in [9], which was used by its creators to overcome 13 out of 15 schemes with success rates between 5%-66%. The presented Decaptcha system consists of five general steps that can be individually hand-tuned to a particular CAPTCHA scheme.

In the last years, generic methods are considered as the state of the art CAPTCHA solving techniques. These methods aim to create end-to-end enclosed solution, which can be operated without any previous fine-tuning on attacked CAPTCHA scheme.

In [5] authors proposed a cutting method which slices each CAPTCHA image in all possible ways and scores all possible combination. The most probable answer is then chosen. Despite the interesting title, authors must utilize many optimizations to lower the computational complexity as the computation cost increases exponentially with the length of CAPTCHA. This results in a lower success rate raging from 5% to 55%. However, the presented attack is the first method not strictly utilizing the segment-than-recognize approach.

In [18] authors presented CAPTCHA solver based on dissecting every CAPTCHA image into a great number of sub-segments. These partial segments are then combined with adjacent sub-segments and recognized. Recognition rate can be up to 77% on some CAPTCHA schemes. However, this method failed when dealing with a complicated background or a large amount of noise.

Most recent attacks use convolutional neural networks (CNN) in combination with other techniques. For example in [36] authors utilized CNNs for feature extraction and Long Short-Term Memory (LSTM) for actual recognition. Another similar work was presented in [30], where two deep networks 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 is the need to build a large annotated dataset for initial system learning.

This disadvantage is almost bypassed by the system in [34]. The system uses a small annotated sample (about 500pcs per CAPTCHA scheme) to learn the generator of synthetic CAPTCHA codes of the same style. The generator is based on a Generative adversarial network (GAN). The generated data is then used to train the basic version of CNN and the original data is used to fine-tune the network. Creating GAN to create large dataset needed for training CAPTCHA solver is a very time consuming task. Training such a network increases the computational complexity and the presented network heavily relied on the noise removal stage. GAN structure described in [35] partially deals with noise removal by introducing GAN-based Background Denoiser.

The method described in [31] overcomes some of these issues by transfer learning. First, the base solver is trained on synthetic CAPTCHA images generated without any noise. Then 500 CAPTCHA images of the targeted CAPTCHA scheme is used for fine tuning the base solver. The paper presented, that before the fine tuning, the average success rate on CAPTCHAs is close to 0%, but after fine tuning on 500 CAPTCHAS, the success rates increases to 50%-97% for latin alphabet based CAPTCHA schemes. The presented CNN is an end-to-end solution and is based on ResNet with RNN attention mechanism.

The main disadvantage of the last two papers is the requirement of the 500 annotated image samples of targeted CAPTCHA scheme. With the average of around 4 CAPTCHA images annotated per minute, the required time is 2 hour of intensive human labor.

The goal of this paper is therefore to introduce a new approach that will require no or absolutely minimal involvement of a person to annotate the training data. Every study discussed in previous paragraphs needed some annotated data as an input. We want to prove, that this part of all previous research papers in this field is obsolete and CAPTCHA breaking system can be trained without any human interaction.

3 Proposed CAPTCHA breaking system

In this paper, we want to simulate a hypothetical CAPTCHA breaking attack. Assume that an attacker has a large group of controlled computers (either on site or hacked computers) to create large bot network. The large number of involved machines is needed, because every attack on common

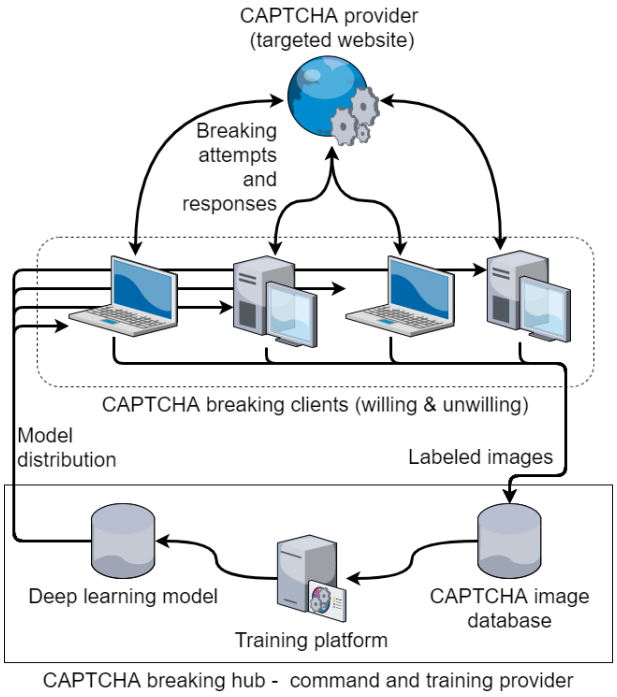


Fig. 1: Proposed CAPTCHA breaking system

CAPTCHA system can be easilly discovered [3,2] and the attacker can be blocked. With the usage of large bot network, a large number of breaking attempts can be done in short time without any chance of detection. The potential attacker also has a central unit at his disposal for a suitable classifier training and management of controlled computers.

The central unit can order its clients to automatically download the CAPTCHA image, use the supplied classifier to acquire predicted answers, and send it back to the web-page. Attacked web-service cannot distinguish the machine from a human without evaluating the sent answer. The attacked web-page will respond and either stops on the current page or continues to the next page. The attacking unit will obtain results in binary yes (and the image with its label is saved) or no (and the image is discarded). Correctly labeled images are then sent to the central unit to fine-tune the classifier.

At first, the model used with all clients is very general and does not have much success. But with each translated CAPTCHA, the central unit has more and more training data to adjust the model accordingly. The central unit then distributes a fine-tuned model to all its clients further improving the data collection success rate.

The above-explained hypothetical system is presented in diagram in Fig. 1.

In these times, huge Denial of Service (DoS) attacks are widely used from time to time. Presented attack on web-service implementing text-based CAPTCHA as a security measure is very similar in its nature to DoS attack and can utilize the same infrastructure.

3.1 Comparison with the previous work

Obtaining a sufficiently large annotated dataset for training CAPTCHA breakers is currently the main problem of all published methods. Hundreds of samples are needed for successful training of standard machine learning-based classifiers [5, 18], and tens of thousands of annotated samples are needed for CNN-based classifiers [28,36]. For the rapid deployment of CAPTCHA breakers on new CAPTCHA system, novel methods are published and each presents an optimal way to reduce the required amount of human labor to a minimum.

Of the several methods published in recent years, it is necessary to highlight the method published in [34]. The GAN-based system generates CAPTCHA challenges similar to the original ones. First, a generic generator is created that is fine-tuned based on 500 images to match the target system. The data generated from this system are then used to train a suitable classifier.

Another interesting way is the use of transfer-learning as in [31]. The general solver is trained on the model of the general CAPTCHA system, then the transfer-learning is used and the presented CAPTCHA breaking system is fine-tuned on 500 real samples so the CAPTCHA breaking system can break the targeted CAPTCHA system with a high accuracy.

We observe, that the common denominator of a large number of methods from recent years is the number of 500 samples of the targeted CAPTCHA system annotated by a human. So far, no method has decreased the number of required manually hand-annotated CAPTCHA images below this limit. The goal of our efforts in this paper is to break this boundary, to use a targeted CAPTCHA system against itself so that it is not necessary to manually annotate a single image.

In Table 1, we compare the number of annotated CAPTCHA samples needed to create a successful classifier for several methods published in recent years with our proposed approach.

4 Experimental setup

The proposed attack on CAPTCHA systems is programmatically simple but requires extensive computational resources and usage of those illegal activities is not even allowed on the academic ground. Without the use of the bot network system presented in Fig. 1, we are not able to directly implement the proposed attack and thus obtain a sufficiently large number of annotated CAPTCHA images to successfully verify the functionality of the system. Our entire workplace is on a uniform range of IP addresses, which the real CAPTCHA system would detect and immediately permanently block whole IP range. Therefore in this paper, we only simulate proposed attack on CAPTCHA system from the machine-learning perspective. The goal is to train a simple deep learning classifiers

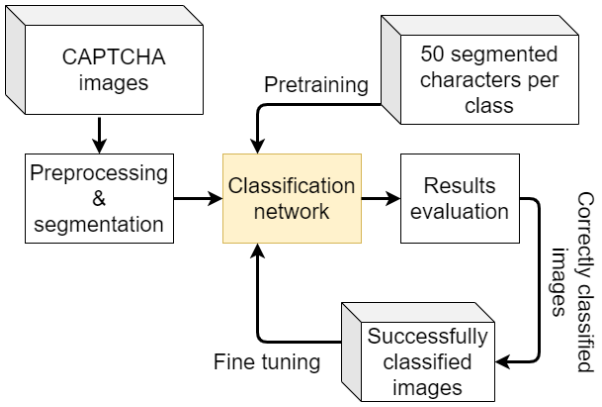


Fig. 2: Experiment overview

(see [19]) sufficient enough to prove the concept. Our current goal presented in this paper is not to create versatile CAPTCHA breaking system like in [30,34,36].

The system used in this article is described in Fig. 2. Every input image is preprocessed and segmented into a number of regions. Every individual region is then fed into a classification neural network. To simulate the response from the targeted server, we compare outputs from all segments (e.g. the assumed character) to the correct answer per single CAPTCHA image. If only one segment (one character) is misclassified, the whole CAPTCHA image is marked as misclassified and the image is discarded in this phase. Only the segments from the images with completely correct answers are used for further fine-tuning. The whole processing pipeline is described bellow in more detail.

The entire experiment was implemented in MATLAB computational environment using mainly Deep Learning Toolbox and Computer Vision Toolbox.

4.1 Input dataset

Dataset of CAPTCHA images used in this article was set up by python library ”Lepture/Captcha” for generating text-based and audio-based CAPTCHA challenges. The source code of this library is available via GitHub from [33].

The main parameters for dataset generation were the length of CAPTCHA text and the character set. We generate CAPTCHA challenges with 5, 7, or 10 characters per image. As for the character set, we want to compare text-based CAPTCHA with:

- numbers 0-9
- lowercase alphabet a-z
- non confusable numbers and letters (0-9, a-z except 0, 1, 9, g, l and o)

In this experiment, we used all nine combinations of these features to generate nine individual groups of data. Each

Table 1: Selected CAPTCHA breaking studies

Year	Author	Focus	Number of annotated samples for training
2021	**Presented approach**	**Simple segmentation, CNN classifier**	**0 manually labeled CAPTCHAs**
2021	Zhang et al. [35]	GAN-based de-noising + CNN based recognition	500 manually labeled CAPTCHAs
2020	Wang et al. [31]	ResNet+RNN attention mechanism, transfer learning	500 manually labeled CAPTCHAs
2020	Noury and Rezaei [28]	End-to-end CNN, fixed size CAPTCHA	50.000 sythetic CAPTCHAs from [33]
2020	Zi et al. [36]	End-to-end CNN and LSTM	200,000 manually labeled CAPTCHAs
2018	Ye et al. [34]	GAN-based generator + CNN based solver	500 manually labeled CAPTCHAs
2016	Gao et al. [18]	Component creating, graph building, k-NN	500 manually labeled CAPTCHAs
2014	Bursztein et al. [5]	Custom segmentation with k-NN scorer and ensemble arbiter	1.000 manually labeled CAPTCHAs

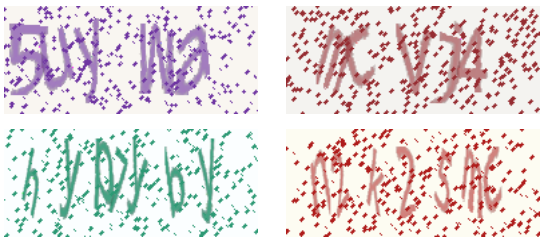


Fig. 3: Sample CAPTCHA images (correct answers: 5uywa, mcvj4, hyp7yby and n2k25mc)

group consists of 50 000 individual text-based CAPTCHA challenges. Sample CAPTCHA images from this dataset are displayed in Fig. 3. The size of all CAPTCHA images is 180x90px.

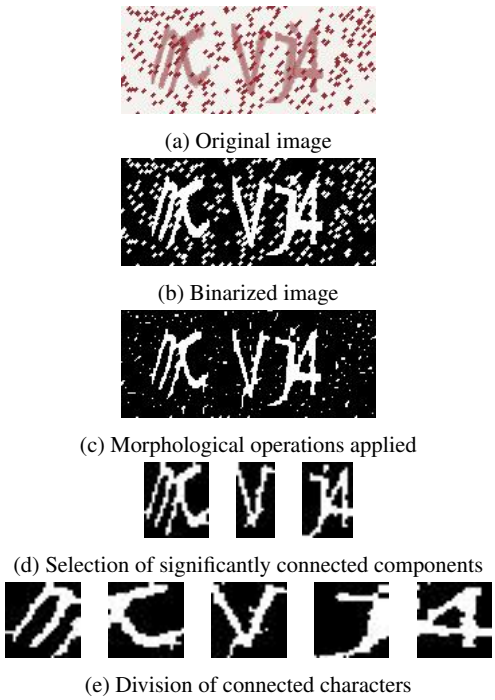
4.2 Segmentation stage

We chose only basic segmentation techniques for this experiment. The aim of this paper is not to repeat the segmentation experiments presented in section 2 (for example [9,5,29,30]), but to demonstrate how to train the classification part without manually creating any dataset. The segmentation method used in this paper can be replaced by one of these methods to increase the success rate.

Every input image in our case is therefore only binarized by the Otsu method. Subsequently, morphological operations are used to remove noise. Small regions are then also excluded from further processing. If there is an overlap of characters, such overlap is detected and non-compliant regions are divided. This process is illustrated in Fig. 4.

4.3 Recognition stage

The main idea of this paper is to use transfer learning to train the CAPTCHA character classifier without any manual dataset creation. This is achieved by generating a small training dataset of single synthetic characters. This dataset contains images of various fonts, sizes and with various distortions. During the fine-tuning stages, the classification model



(a) Original image

(b) Binarized image

(c) Morphological operations applied

(d) Selection of significantly connected components

(e) Division of connected characters

Fig. 4: Segmentation process overview

for character recognition is slightly changed with the new training data and thus adapting more and more to the targetted CAPTCHA scheme. The training process is illustrated in Fig. 2.

We describe the learning algorithm only for text-based CAPTCHAs with digits, the rest of the experiments was performed similarly but with a greater number of characters.

The classification network was firstly pre-trained on 500 synthetic images (50 per class) containing one distorted letter per image. From these 500 images, 75% was used for training, 25% for validation. This general training dataset in real attack can be augmented by a small sample of targetted characters (or from any similar CAPTCHA) to speed up the learning stage. The training used the Stochastic Gradient Descent with Momentum algorithm introduced by [25]. The training process was run for 10 epochs with an initial learning rate set

Table 2: Convolutional deep neural network used for classification

	Layer type	Layer description
1	Image Input	28x28x1 images
2	Convolution	8pcs 3x3x1 convolutions
3	Batch Normalization	8 channels
4	ReLU	-
5	Max Pooling	2x2 max pooling
6	Convolution	16pcs 3x3x8 convolutions
7	Batch Normalization	16 channels
8	ReLU	-
9	Max Pooling	2x2 max pooling
10	Convolution	32pcs 3x3x16 convolutions
11	Batch Normalization	32 channels
12	ReLU	-
13	Fully Connected	10 outputs
14	Soft-maximum	-
15	Classification Output	Cross Entropy Function for 10 Mutually Exclusive Classes

to 0.01. The exact topology of the convolutional deep neural network used in this experiment is shown in Table 2.

This weak classifier was then used to classify all images from the previously described dataset. To simulate the server response, only when all segments from tested CAPTCHA image are classified correctly, the answer is registered and the CAPTCHA image is stored.

After the whole dataset is processed, the stored CAPTCHA images are segmented again with the same segmentation algorithm and the new training dataset is created. The fine-tuning step utilized the same parameters as the first training stages, but only for 5 epochs.

5 Experimental results

All experiments were computed on a desktop computer with AMD FX-6350 with 6 core processor and 3.9 GHz frequency, 24GB RAM and GeForce GT 630 graphics card.

5.1 Segmentation stage results

The above-described approach is applicable to the simple text CAPTCHA used in this experiment. Sample output from this stage is depicted in Fig. 5. The described segmentation technique is successful in 22%-80% depending on CAPTCHA length (see Table 3).

The presented segmentation stage is also suitable to some other schemes that do not intentionally use character overlap, but rather the characters are randomly distributed around the image and resulting overlaps are small. For example about half of the text-based CAPTCHA schemes attacked in [9] and one-third of CAPTCHA schemes attacked in [36] can be successfully preprocessed in this way.

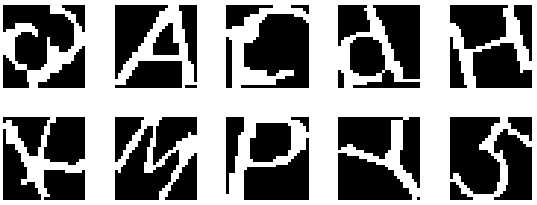


Fig. 5: Sample output from segmentation stage (characters a, A, C, d, H, k, M, P, Y and 5)

Table 3: Segmentation stage results per feature combination

	CAPTCHA text length		
	5	7	10
Numbers 0-9	80.02%	59.47%	21.88%
Lowercase alphabet a-z	68.23%	50.79%	22.34%
Non-confusable lowercase characters and letters	71.11%	52.54%	22.29%

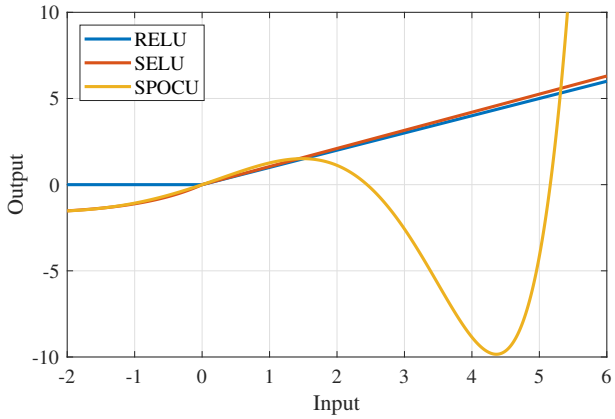


Fig. 6: Graph comparison of activation functions RELU, SELU and SPOCU

On the other hand, the presented segmentation stage is not versatile enough to broad usage and is the most serious drawback of the presented experiment.

5.2 Network optimization

The important parts of the network are activation functions. Their main function is to increase the network ability. In this manner we tried to upgrade the network performance with two different activation functions Scaled Exponential Linear Unit (SELU) [23] and Scaled Polynomial Constant Unit (SPOCU) [22]. The interesting ability of these activation functions is to create a self normalizing network. In the Fig. 6 is shown the comparison of these activation functions with the Rectifier Linear Unit (RELU) [26].

The experimental setup was as follows. We create an updated architecture with SPOCU activation function instead of RELU activation function, the rest of the network remains

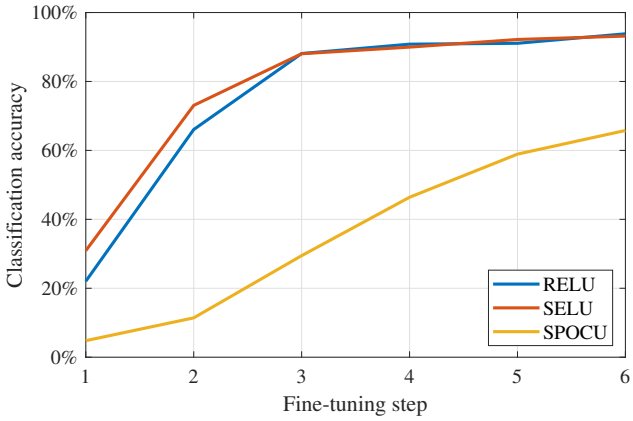


Fig. 7: Classification accuracy per fine-tuning step for 5 character length CAPTCHA with different activation functions

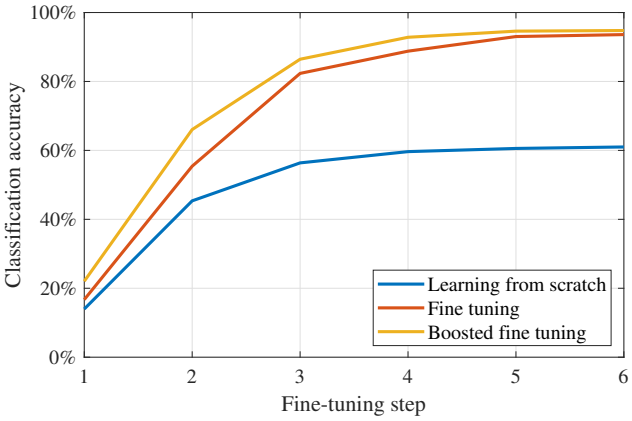


Fig. 8: Comparison of classification accuracy per fine-tuning step for different training strategies (text-based CAPTCHA with length of 5 digits)

the same as in the table 2. For the performance we transform the inputs as in [22] into interval $[-\beta\gamma,(1-\beta)\gamma]$ by eq.

$$x_{norm} = \gamma\frac{x - min(x)}{max(x) - min(x)} - \beta\gamma. \quad (1)$$

The SPOCU parameters [22,21] which we tried are $\alpha = 3.0937$, $\beta = 0.6653$, $\gamma = 4.437$ and $c = \infty$. For the SELU activation function we also replace the RELU activation function with SELU activation function. The parameters used for SELU were $\alpha = 1.67326$ and $\gamma = 1.0507$. Both activation functions were built with the MATLAB custom layer.

From the Fig. 7 can be concluded that RELU activation function surmount the SPOCU activation function and results are comparable to the SELU activation function. The MATLAB implementation is optimized and thus faster than custom user implementation. From the time comparison on the validation dataset the RELU is 50% faster than SELU and 70% faster than SPOCU. Therefore we use RELU in the rest of experiments. However, it should be noted that the strength of the SPOCU auto-normalization layer would be manifested when used in a different network architecture, which makes this comparison impossible.

5.3 Different training strategies for classification stage

In this paper, we compare three different strategies to train the classification network. The comparison of all these strategies is depicted in Fig. 8. The proposed strategy is described in following section.

The first training strategy is to create an individual dataset for every fine-tuning step. We call this strategy "learning from scratch". In every fine-tuning step, we train a fresh new classifier from the data acquired with the last classifier. This strategy works with some limitations for short CAPTCHA schemes with small character sets (for example digits-only CAPTCHA). The main drawback of this approach is in the swift shift from the original character features in every fine-tuning step. This results in misclassified characters in later stages, that have been correctly recognized in the first steps.

The second strategy is to fine-tune the classifier instead of training a fully new classifier with every new data from targeted CAPTCHA system. The training dataset is created from all correctly classified CAPTCHA at every stage while omitting the pre-train data. This approach has the main potential for CAPTCHA schemes with a small character set. For the recognition of a bigger character set like all lowercase letters, this approach has a tendency to omit some characters which significantly reduce the accuracy and learning speed of this method.

5.4 Proposed approach

The proposed approach is based on previous learning strategy. The main difference is that we added data from the pre-training step to every class. In this way, we guarantee that the classifier does not forget some characters. During the first fine-tuning steps some characters were recognized but were not added to the training data because one or more characters from evaluated CAPTCHA image was not been recognized. As the classification of this previously unrecognized character is growing, the probability of adding more characters to training classes increases as well.

The proposed semi-supervised approach can be successfully used to break text-based CAPTCHA of different length, but the length of CAPTCHA significantly slow down the training process. After approximately 4 fine-tuning stages we can observe an almost exponential reduction in the training accuracy with the length of the CAPTCHA. This is illustrated in Fig. 9. It is important to highlight, that the accuracy for CAPTCHA schemes with a length of 7 and 10 is still

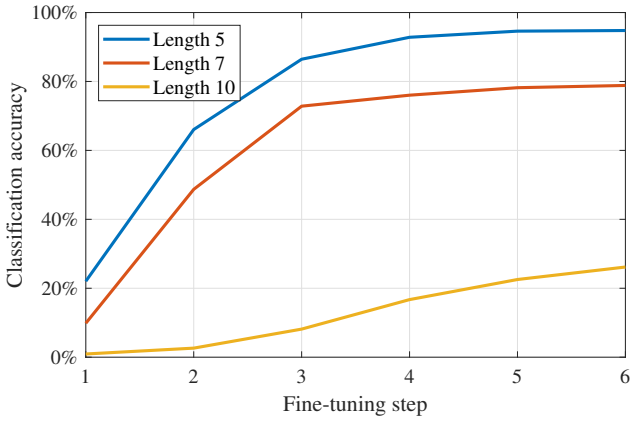


Fig. 9: Classification accuracy per fine-tuning step for digits CAPTCHA

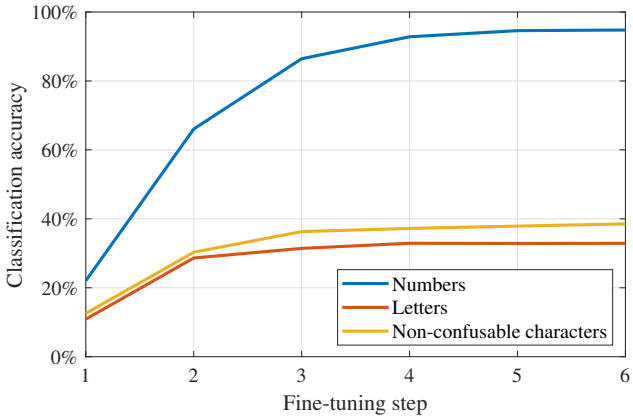


Fig. 10: Classification accuracy per fine-tuning step for 5 character length CAPTCHA with different character set

increasing significantly even in fine-tuning step 6 when our evaluation finished.

As for the different character set used for training, the classification accuracy is significantly lower when using longer character set (see Fig. 10). After 6 fine-tuning steps, the classification accuracy for letters is almost 2,5 times lower than in the case of digits, which is almost the same ratio as in the case of characters length. The classification accuracy for the non-confusable letters and digits is slightly higher because the characters are easier to recognize than in the case of some letters.

All results are summarized in Table 4. In all cases, the accuracy is greater than previously mentioned 1% threshold and therefore we can state, that the CAPTCHA scheme used in this experiment is broken.

The quantification of the improvement in classification between the pre-trained classifier and the fine-tuned classifier is shown in Fig. 11. The value in the figure is defined as the ratio between the number of correctly classified CAPTCHAs after the first and sixth learning steps. There is an exponential

Table 4: Classification accuracy after 6 fine-tuning steps

	CAPTCHA text length		
	5	7	10
Numbers 0-9	94.78%	78.84%	26.16%
Lowercase alphabet a-z	32.89%	20.39%	1.13%
Non-confusable lowercase characters and letters	38.54%	19.86%	2.54%

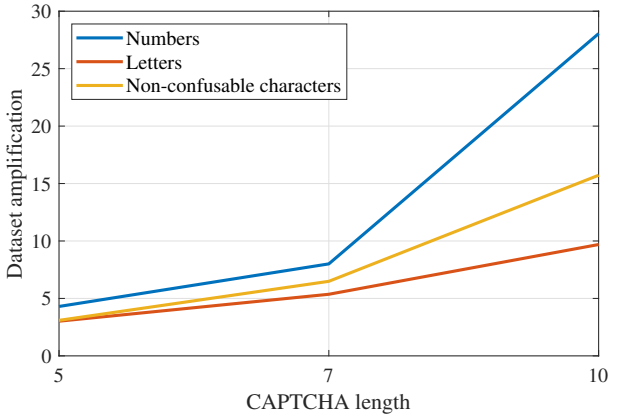


Fig. 11: Amplification of training dataset between first and last fine-tuning steps

increase with the increase of CAPTCHA text length. This is caused by the fact that in the first steps there is a greater probability of error in at least one character and thus there is the slow down in the fine-tuning process.

5.5 Character accuracy

Our simulation has an advantage in knowledge of the solution for every CAPTCHA in the dataset. Although we cannot use this advantage in the simulation (because we would deny the meaning of the experiment), we can use this information for the evaluation of the classification accuracy per single character. Fig. 12 clearly illustrates that single character accuracy is significantly higher in the first fine-tuning steps. In later learning steps, the differences decrease and the CAPTCHA recognition system as a whole reaches the limit at the level of accuracy of individual character recognition.

6 Conclusion

In this paper, we have presented a novel strategy for automatic training of text-based CAPTCHA without previous dataset creation. We have demonstrated, that the CAPTCHA system can be broken via transfer learning. The proposed attack would utilize similar network infrastructure as the infrastructure used for Denial of Service attacks.

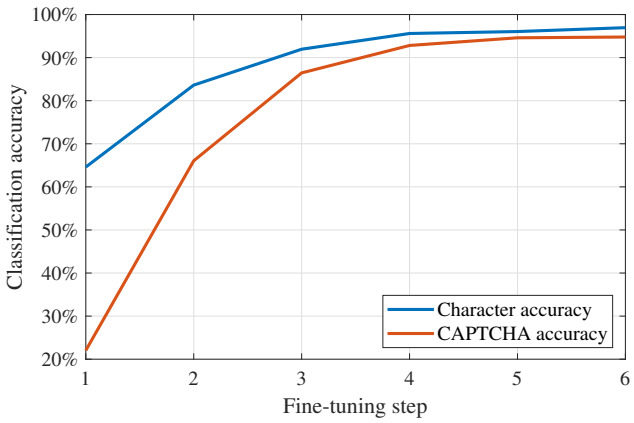


Fig. 12: Classification accuracy for 5 digits numerical CAPTCHA

The simulation of this attack was in particular successful on shorter 5-character CAPTCHAs with only digits. The classification accuracy on this CAPTCHA scheme was 94.78% after 6 fine-tuning steps with 50000 attacks per step. With the length of CAPTCHA text and with the complex character set, the classification accuracy decrease rapidly, but the accuracy for all tested combinations was greater than 1%.

The main disadvantage of the presented approach is in the segmentation process, which is fine-tuned for this particular dataset.

Our future goal is to combine our training approach with the previous work in this field of research to create a general end-to-end solution that requires no training data.

The obtained results confirm that the proposed CAPTCHA breaking technique is feasible and the security measures must be ready on this kind of attack.

Acknowledgements 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.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: CAPTCHA: Using Hard AI Problems for Security. In: Lecture Notes in Computer Science, pp. 294–311. Springer, Berlin, Heidelberg (2003). DOI 10.1007/3-540-39200-9_18. URL `http://link.springer.com/10.1007/3-540-39200-9{\_}18`
2. Arai, T., Okabe, Y., Matsumoto, Y.: Precursory Analysis of Attack-Log Time Series by Machine Learning for Detecting Bots in CAPTCHA. In: 2021 International Conference on Information Networking (ICOIN), pp. 295–300 (2021). DOI 10.1109/ICOIN50884.2021.9333881
3. Arai, T., Okabe, Y., Matsumoto, Y., Kawamura, K.: Detection of Bots in CAPTCHA as a Cloud Service Utilizing Machine Learning. In: 2020 International Conference on Information Networking (ICOIN), pp. 584–589 (2020). DOI 10.1109/ICOIN48656.2020.9016522
4. Athanasopoulos, E., Antonatos, S.: Enhanced CAPTCHAs : Using Animation to Tell Humans and Computers Apart. Ifip International Federation For Information Processing **4237**, 97–108 (2006). DOI 10.1007/11909033_9. URL `http://link.springer.com/chapter/10.1007/11909033{\_}9`
5. Bursztein, E., Aigrain, J., Moscicki, A., Mitchell, J.C.: The End is Nigh: Generic Solving of Text-based CAPTCHAs (2014). URL `http://portal.acm.org/citation.cfm?id=2671296`
6. Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C., Mitchell, J.: The failure of noise-based non-continuous audio captchas. In: Proceedings - IEEE Symposium on Security and Privacy, pp. 19–31 (2011). DOI 10.1109/SP.2011.14
7. Bursztein, E., Bethard, S.: Decaptcha: breaking 75% of eBay audio CAPTCHAs. Proceedings of the 3rd USENIX conference on Offensive technologies **1**(8), 1–7 (2009)
8. Bursztein, E., Bethard, S., Fabry, C., Mitchell, J.C., Jurafsky, D.: How good are humans at solving CAPTCHAs? A large scale evaluation. In: Proceedings - IEEE Symposium on Security and Privacy, pp. 399–413. IEEE (2010). DOI 10.1109/SP.2010.31. URL `http://ieeexplore.ieee.org/document/5504799/`
9. Bursztein, E., Martin, M., Mitchell, J.C.: Text-based CAPTCHA strengths and weaknesses. In: Proceedings of the ACM Conference on Computer and Communications Security, pp. 125–138 (2011). DOI 10.1145/2046707.2046724
10. Bursztein, E., Moscicki, A., Fabry, C., Bethard, S., Mitchell, J.C., Jurafasky, D.: Easy Does It: More Usable CAPTCHAs. In: CHI '14 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 2637–2646. 1600 Amphitheatre Pkwy (2014). URL `https://www.elie.net/publication/easy-does-it-more-usable-captchas`
11. Chellapilla, K., Larson, K., Simard, P., Czerwinski, M.: Computers beat humans at single character recognition in reading based human interaction proofs (HIPs). In: 2nd Conference on Email and Anti-Spam, pp. 1–8. Conference on Email and Anti-Spam, CEAS (2005)
12. Chellapilla, K., Simard, P.: Using Machine Learning to Break Visual Human Interaction Proofs (HIPs). In: L. Saul, Y. Weiss, L. Bottou (eds.) Advances in Neural Information Processing Systems, vol. 17, pp. 265–272. MIT Press, Vancouver (2005). URL `https://proceedings.neurips.cc/paper/2004/file/283085d30e10513624c8cece7993f4de-Paper.pdf`
13. Chow, Y.W., Susilo, W.: AniCAP: An Animated 3D CAPTCHA Scheme Based on Motion Parallax. In: D. Lin, G. Tsudik, X. Wang (eds.) Cryptology and Network Security: 10th International Conference, CANS 2011, Sanya, China, December 10-12, 2011. Proceedings, pp. 255–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). DOI 10.1007/978-3-642-25513-7_18. URL `http://link.springer.com/10.1007/978-3-642-25513-7{\_}18`
14. Desai, A., Patadia, P.: Drag and Drop: A Better Approach to CAPTCHA. In: 2009 Annual IEEE India Conference, pp. 1–4 (2009). DOI 10.1109/INDCON.2009.5409359
15. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. Journal of Computer and System Sciences **55**(1), 119–139 (1997). DOI https://doi.org/10.1006/jcss.1997.1504. URL `http://www.sciencedirect.com/science/article/pii/S002200009791504X`
16. Gao, H., Tang, M., Liu, Y., Zhang, P., Liu, X.: Research on the Security of Microsoft's Two-Layer Captcha. IEEE Transactions on Information Forensics and Security **12**(7), 1671–1685 (2017). DOI 10.1109/TIFS.2017.2682704
17. Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., Yan, J.: The robustness of hollow CAPTCHAs. In: Proceedings of the ACM Confer-

ence on Computer and Communications Security, pp. 1075–1086 (2013). DOI 10.1145/2508859.2516732
18. Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., Zhang, P., Zhou, X., Wang, X., Li, J.: A Simple Generic Attack on Text Captchas. In: Network and Distributed System Security Symposium (NDSS 2016), pp. 1–26 (2016). DOI 10.14722/ndss.2016.23154
19. Horak, K., Sablatnig, R.: Deep learning concepts and datasets for image recognition: overview 2019. In: Eleventh International Conference on Digital Image Processing (ICDIP 2019), 11179, pp. 484–491. SPIE (2019). DOI 10.1117/12.2539806. URL https://doi.org/10.1117/12.2539806
20. Kaur, K., Behal, S.: Designing a Secure Text-based CAPTCHA. In: Procedia Computer Science, vol. 57, pp. 122–125. Elsevier (2015). DOI 10.1016/j.procs.2015.07.381
21. Kiseľák, J., Lu, Y., Švihra, J., Szépe, P., Stehlík, M.: Correction to: "SPOCU": scaled polynomial constant unit activation function. Neural Computing and Applications (2020). DOI 10.1007/s00521-020-05412-6. URL https://doi.org/10.1007/s00521-020-05412-6
22. Kisel'ák, J., Lu, Y., Švihra, J., Szépe, P., Stehl\'\ik, M.: "SPOCU": scaled polynomial constant unit activation function. Neural Computing and Applications pp. 1–17 (2020)
23. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-Normalizing Neural Networks. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30 (NIPS 2017), vol. 30, pp. 971–980. Curran Associates, Inc. (2017). URL https://proceedings.neurips.cc/paper/2017/file/5d44ee6f2c3f71b73125876103c8f6c4-Paper.pdf
24. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: breaking a visual CAPTCHA. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings. **1**, I–I (2003)
25. Murphy, K.P.: Machine Learning: A Probabilistic Perspective, 1. edition edn. The MIT Press, Cambridge, MA (2012)
26. Nair, V., Hinton, G.: Rectified Linear Units Improve Restricted Boltzmann Machines Vinod Nair. In: Proceedings of ICML, vol. 27, pp. 807–814 (2010)
27. Nguyen, V.D., Chow, Y.W., Susilo, W.: A CAPTCHA scheme based on the identification of character locations. In: X. Huang, J. Zhou (eds.) Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 8434 LNCS, pp. 60–74. Springer International Publishing, Cham (2014). DOI 10.1007/978-3-319-06320-1_6. URL https://doi.org/10.1007/978-3-319-06320-1{_}6
28. Noury, Z., Rezaei, M.: Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment (2020). URL http://arxiv.org/abs/2006.08296
29. Starostenko, O., Cruz-Perez, C., Uceda-Ponga, F., Alarcon-Aquino, V.: Breaking text-based CAPTCHAs with variable word and character orientation. Pattern Recognition **48**(4) (2015). DOI 10.1016/j.patcog.2014.09.006
30. Tang, M., Gao, H., Zhang, Y., Liu, Y., Zhang, P., Wang, P.: Research on Deep Learning Techniques in Breaking Text-Based Captchas and Designing Image-Based Captcha. IEEE Transactions on Information Forensics and Security **13**(10), 2522–2537 (2018). DOI 10.1109/TIFS.2018.2821096. URL https://ieeexplore.ieee.org/document/8327894/
31. Wang, P., Gao, H., Shi, Z., Yuan, Z., Hu, J.: Simple and Easy: Transfer Learning-Based Attacks to Text CAPTCHA. IEEE Access **8**, 59044–59058 (2020). DOI 10.1109/ACCESS.2020.2982945
32. Yan, J., Ahmad, A.S.E.: Breaking Visual CAPTCHAs with Naive Pattern Recognition Algorithms. In: Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007), pp. 279–297 (2008). DOI 10.1109/acsac.2007.4412996
33. Yang, H.: GitHub - lepture/captcha: A CAPTCHA library that generates audio and image CAPTCHAs. (2020). URL https://github.com/lepture/captcha/
34. Ye, G., Tang, Z., Fang, D., Zhu, Z., Feng, Y., Xu, P., Chen, X., Wang, Z.: Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, pp. 332–348. Association for Computing Machinery, New York, NY, USA (2018). DOI 10.1145/3243734.3243754. URL https://doi.org/10.1145/3243734.3243754
35. Zhang, N., Ebrahimi, M., Li, W., Chen, H.: A Generative Adversarial Learning Framework for Breaking Text-Based CAPTCHA in the Dark Web. In: 2020 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 1–6 (2020). DOI 10.1109/ISI49825.2020.9280537
36. Zi, Y., Gao, H., Cheng, Z., Liu, Y.: An End-to-End Attack on Text CAPTCHAs. IEEE Transactions on Information Forensics and Security **15**, 753–766 (2020). DOI 10.1109/TIFS.2019.2928622