Adversarial AEA-CAPTCHA

Alireza Khodaie, Arman Torikoglu, Muhammed Esad Simitcioglu

SUMMARY

CAPTCHA is used to make decisions if the user is a human or a computer on the website. There are many different types of CAPTCHAs used today such as text-based, image-based, audio-based, etc. Recently, the asked test that enables this authentication system to distinguish between humans and computers can be easily solved using machine learning models. CAPTCHA designers have tried their best to improve security but most schemes deployed on the websites have been cracked [15].

This project demonstrates two things. First, the state-of-the-art CAPTCHA solvers' accuracy rate and our CAPTCHA solver's accuracy rate. Second, we will create unsolvable CAPTCHA with some cybersecurity techniques such as Evasion attacks and Carlini-Wagner Attacks. Then we will use these state-of-the-art solvers and our CAPTCHA solver to solve our newly created "unsolvable" CAPTCHA to demonstrate that it's possible to improve the robustness of text-based CAPTCHAs.

MOTIVATION AND PROBLEM STATEMENT

A good CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) should be simple for people to solve yet difficult for computers to understand. It's challenging to strike this balance between usability and security. CAPTCHA is used to prevent automated registration, spam, or malicious bot programs[1]. It is a Turing Test, which uses activities that are simple for people to do but difficult for computers in order to tell humans and computers apart. The CAPTCHA can be regarded as effective if it has a human success rate of 90% or above and a computer program success rate of less than 1% [2].

A larger number of early simple text-based schemes deployed in the wild have been broken, such as Google and Microsoft[2]. CAPTCHA designers try different techniques to improve the robustness of the CAPTCHA such as crowding characters together, noise arcs, two-layer structure, etc. However, all of these resistance mechanisms seem to be ineffective, as studies [3,4] have demonstrated. These concerns opened a new topic in the CAPTCHA studies. Are we really safe on the websites and if we are not, can cybersecurity solve this problem?

The solution we are proposing in this project will enhance our understanding of attacks performed on ML models, namely evasion attacks. In terms of evasion attacks, we will use state-of-art security techniques and this will keep us up to date with the current literature.

PROPOSED APPROACH

CAPTCHA Solver

The text-based CAPTCHA's lack of security has been covered in great detail in the pertinent literature. There is no denying that text-based CAPTCHA remains the most basic type. Unfortunately, there still exist websites that use text-based CAPTCHA. It's easier to solve text-based CAPTCHA than the others. Currently, there are lots of CAPTCHA solver implementations with different architectures. Most CAPTCHAs are classification tasks, and the most recent automatic recognition was carried out using neural networks. Shao et al. [6] have used different methods for solving CAPTCHAs. For shallow models, they have used SVM [7], KNN [8], and random forest classifiers [9]. For the deep neural networks, they have used various models such as LeNet [10], AlexNet [11], VGG11, VGG13, VGG16, VGG19 [12], GoogLeNet [13], and ResNets [14].

All of these state-of-art algorithms have significant accuracy rates. First, we will implement our own CAPTCHA solver. We will use Convolutional Neural Networks (CNN) as the main architecture to implement our CAPTCHA solver. After implementing the solver, we will analyze its accuracy on test data. Since adversarial examples can fool neural networks, we will come up with an adversarial CAPTCHA generator to beat the solver that we propose.

Adversarial Example

Some researchers have found that some state-of-the-art deep neural network classifiers can be fooled by adding some small but intentionally worst-case perturbations that people cannot notice in an original image. These disturbed images are called adversarial examples [5]. Szegedy et al. made the initial argument for the idea of an adversarial example in 2014 [5]. They discovered a number of machine learning models, including cutting-edge neural networks, that are susceptible to very slight changes in the original images. In other words, these machine learning models incorrectly identify cases that are a little different from those that are correctly classified. There are different types of attacks to construct this kind of attack such as evasion attacks and Carlini-Wagner attacks

Evasion Attack

Attacks during testing known as "evasion attacks" tries to generate undesirable outputs in the machine learning system by manipulating the input data. CNN models are especially vulnerable to Evasion Attacks [16]. Therefore, this technique is relevant to attack CAPTCHA solvers as CNN models are a prominent part of the current CAPTCHA solvers. The technique is to find a minimum perturbation to be added to the test sample so that it "fools the classifier" [16]. In order to find these optimized perturbations, there are a few optimization techniques to employ. Carli-Wagner Attack is one of them and we are planning to use it in our Evasion Attack.

Carlini-Wagner Attack

Carlini-Wagner Attack solves this optimization problem using advanced mathematical equations and proofs. The initial formula of the problem is MIN(D(x, x+ δ)): such that $C(x + \delta) = t$ and $x + \delta \in [0,1]^n$, and this formula is later simplified to be solvable by Adam optimizer [17].

Threats

There are critical points that we want to avoid while creating unsolvable CAPTCHAs. One of them is overestimating the random noise. As opposed to humans, computer vision-based algorithms today are more precise and cunning in avoiding noise. Additionally, characters that are difficult for both a machine and a person to distinguish, such as the number "0" and the letter "O", and the number "l" and the letter "I", might be good examples to fool the model.

Implementation

In this project, we will use mainly Python programming language for the scripts and tests. The following Python modules will be used: Pillow, TensorFlow, Numpy, pytesseract, etc. If the following modules are not sufficient, new modules will be added. For our interface, we will use Django Framework to build a web application for our project. All of our work will be deployed in our GitHub repository.

DELIVERABLES

- Captcha Solver: CNN model to solve a text-based CAPTCHA (if time permits we can include Image-based CAPTCHA too) will be used in the Django server that will be presented in a neat web interface.
- Adversarial CAPTCHA Generator: Python program that generates the adversarial CAPTCHA images that are expected to be solved with pretty low accuracy.
- Project Demo: To demonstrate the features of our CAPTCHA Solver and Adversarial CAPTCHA generator.
- Final Project Report: This will include our implementation of the CAPTCHA solver, its accuracy, our novel implementation of Adversarial CAPTCHA, and its performance on the CAPTCHA Solver.

TIMELINE

Task	Dec 26- Jan 1	Jan 2- 8	Jan 9-15	Jan 16-26
Captcha solver implementation	Alireza, Esad			
Adversarial examples on the implemented captcha solver		Alireza, Arman		
Coding a web application to demonstrate functionality of the proposed attack			Arman, Esad	
Writing final report				Esad, Arman, Alireza

BIBLIOGRAPHY

[1] Ahn, L., Blum, M., Hopper, N., & Langford, J. (2007). Using hard ai problems for security. In Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques (pp. 294-311).

[2] Bursztein, E., Martin, M., & Mitchell, J. (2011, October). Text-based CAPTCHA strengths and weaknesses. In Proceedings of the 18th ACM conference on Computer and communications security (pp. 125-138). ACM.

[3] Gao, H., Yan, J., Cao, F., Zhang, Z., Lei, L., Tang, M., ... & Li, J. (2016). A Simple Generic Attack on Text Captchas. In NDSS.

[4] Gao, H., Wang, X., Cao, F., Zhang, Z., Lei, L., Qi, J., & Liu, X. (2016). Robustness of text-based completely automated public turing test to tell computers and humans apart. IET Information Security, 10(1), 45-52.

[5] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks. ICLR, abs/1312.6199.

[6] Shao, R., Shi, Z., Yi, J., Chen, P., Hsiesh, C. (2021) Robust Text CAPTCHAs Using Adversarial Examples

[7] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. Neural processing letters, 9(3):293–300, 1999.

[8] Naomi S Altman. An introduction to kernel and nearestneighbor nonparametric regression. The American Statistician, 46(3):175–185, 1992.

[9] Andy Liaw, Matthew Wiener, et al. Classification and regression by randomforest. R news, 2(3):18–22, 2002.

[10] Yann LeCun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6):84–90, 2017.

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[13] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[15] Gao, H., Tang, M., Liu, Y., Zhang, P., & Liu, X. (2017). Research on the Security of Microsoft's Two-Layer Captcha. IEEE Transactions on Information Forensics and Security, 12(7), 1671-1685.

[16] Chenghui Shi, Xiaogang Xu, Shouling Ji, Kai Bu, Jianhai Chen, Raheem Beyah, and Ting Wang (2019) Adversarial CAPTCHAs, pages 1-3.

[17] Bibek Poudel (2020). Explaining the Carlini & Wagner Attack Algorithm to Generate Adversarial Examples.