Breaking and Building CAPTCHAs: A Dual Approach to CAPTCHA Security Using Machine Learning and Adversarial Techniques

İdil Görgülü Batu Orhun Gündüz Aylin Melek Eren Can

1. Summary

This project focuses on addressing the vulnerabilities of CAPTCHA systems caused by the development of advanced machine learning models. It aims to design a Convolutional Neural Network (CNN)-based solver model which can decode CAPTCHA images and, concurrently, develop unsolvable CAPTCHA designs to perform evasion attacks on the trained model. Combining two datasets found through Kaggle, the project involves preprocessing data, training a robust machine learning model, and introducing adversarial noise to evaluate weaknesses in the system through black-box and white-box evasion attacks. The dual-focus approach of this project combines offensive and defensive security research.

2. Motivation and Problem Statement

CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart) are computer tests that serve as significant defence mechanisms for formulating the distinction between automated bots and humans. These tests contribute to the prevention of possible cyber security threats, attacks, credential thefts, and breaches of anonymity of web services [1]. CAPTCHAs have been integral systems for protecting sensitive information inside online services such as user account registrations and financial transactions, yet the development of advanced machine learning algorithms and AI-based methods made the solution of these tests by automated systems possible and thus decreased the functionality of CAPTCHAs to a certain degree. This project comes at a crucial time when the usage of sensitive information online is at an all-time high level and it is necessary to implement robust mechanisms to safeguard digital resources.

CAPTCHAs are still widely used despite the availability of various methods to bypass them. The existing methods for solving them have high time consumption, require intensive prior labour and are mostly scheme specific [2]. Therefore, it is possible to generate CAPTCHA models which are unsolvable by ML models through numerous techniques, including introducing background noise.

This project presents a multifaceted challenge: designing an automated system capable of solving existing CAPTCHAs while simultaneously generating unsolvable ones to use on evasion attacks. As it will have the aim of implementing an ML-based method to solve these problems while concurrently producing adversarial unsolvable CAPTCHAs, this project will focus on both offensive and defensive security research. The project demands a deep understanding of pattern recognition as well as adversarial techniques. The novelty lies in combining two seemingly opposing goals: developing a CAPTCHA solver and defeating the model.

3. Technical Approach

The project will be divided into two main components, one which aims to develop a Convolutional Neural Network (CNN)-based machine learning model capable of decoding CAPTCHAs and one which utilizes adversarial techniques to generate CAPTCHA images which are resistant to the solver and performs evasion attacks.

3.1. Dataset

We will utilize a custom CAPTCHA dataset generated by combining two datasets found through Kaggle. Both datasets include images which contain 5 character-long alphanumeric sequences and in both datasets, the labels of images are encoded within their filenames. The first dataset [3] consists of

approximately 113,000 colorful jpg images. This dataset was preferred because the source code used for generating the images is accessible through a Github repository [4]. We aim to employ the same source code for generating unsolvable images during the implementation of the second part of the project. The second dataset [5] contains 83,300 colored png files. This dataset was chosen to increase the diversity of training data while maintaining the overall characteristics such as number of characters and included character range.

3.2. Resources

- **Programming Languages:** Python is chosen as the main programming language to accomplish the development of the CNN model. Using Python alongside Python-based tools will help to achieve an effective workflow. We may also utilize PHP in order to generate custom CAPTCHAs for the second part of the project.
- **Programming Environments**: The primary programming environment for this project will be Jupyter Notebook (.ipynb). For computationally intensive tasks where we need GPU support, we will be using Google Colab to increase processing speed.
- **Libraries and Tools:** The library which will be used the most during the project is PyTorch, because of its large variety of features for implementing learning tasks. In one of the possible training approaches, we may use OpenCV to split multi-character CAPTCHAs into individual character images for training. We will also use Pandas and NumPy for data preprocessing and matplotlib for visualization of results.

3.3. Data Preprocessing

The initialization process of the dataset will include deciding on the optimal split ratios for training, validation and testing using Python's Pandas and NumPy. After the organization of the dataset into splits, the images will be normalized to a uniform and fixed size and will be converted to grayscale. To increase diversity, it is possible to apply augmentation through rotations, scaling, noise additions and distortions.

3.4. Model Development and Evaluation

A CNN will be designed and implemented using PyTorch and the customized CAPTCHA dataset for training. The model architecture will include multiple convolutional layers for feature extraction, followed by pooling layers to dimensionality reduction and overfit prevention. The model will also include fully connected layers for mapping the extracted features to alphanumeric classifications. Hyperparameters such as learning rate, batch size, and number of epochs will be fine-tuned through further experimentation. The trained model will be tested on unseen CAPTCHA samples from the test set and the model performance will be evaluated using concepts of accuracy, precision and recall as well as confusion matrices to map and analyze classification errors.

3.5. CAPTCHA Manipulation and Evasion Attacks

To generate adversarial CAPTCHA images not detectable by the solver, various noise-adding methodologies including the targeted and untargeted Fast Gradient Sign Method (FGSM) to add background noise to images. We will also try to introduce patches and overlapping characters to various images, which will not prevent human-detectability of images but affect ML model performance, which will be evaluated later on through the utilization of the same techniques we plan to use in model development to make comparisons achievable. Using the generated images, black-box and white-box evasion attacks will be employed to assess and exploit vulnerabilities in the solver system. White-box attacks will be performed by the group members who participated in model training whereas for black-box attacks, a group member who does not have access to the model will be chosen.

4. Deliverables

By the conclusion of the project, the following deliverables will be provided:

- **Jupyter Notebook:** A .ipynb file containing the CAPTCHA solver, including data preprocessing, model training, evaluation, and visualization of results.
- **CAPTCHA Manipulation Code and Example CAPTCHAs:** Scripts for generating unsolvable CAPTCHA images using adversarial techniques and image distortions and example images.
- **Adversarial Attack Implementation:** Code scripts demonstrating the application of white-box and black-box evasion attacks on the solver model.
- **Documentation and Presentation:** The final project report and a slide deck summarizing the project, the key findings and their implications for data privacy and security.

5. Timeline

Task	Dec 16-22	Dec 23-29	Dec 30-Jan 5	Jan 6-12	Jan 13-19	Jan 20-24
Research & Planning	Explore CAPTCHA types, existing ML models for solving and evasion attacks. (Everyone)	Draft project goals and deliverables. Finalize project plan and proposal (Everyone)				
Data Collection & Preprocessing		Collect CAPTCHA datasets. (İdil, Eren)	Preprocess data. (İdil, Batu, Eren)			
CAPTCHA Solver Implementation		Design model architecture. (Batu, İdil)	Train the model and fine-tune hyperparameters (Batu, Eren)	Test solver on diverse samples and measure accuracy. (Aylin)		
Generate Unsolvable CAPTCHAs		Research techniques for unsolvable CAPTCHA generation. (Aylin)	Implement noise, distortions, and adversarial perturbations. (Everyone)	Test the CAPTCHA solver against the unsolvable designs. (İdil)	Finalize unsolvable CAPTCHA designs. (Everyone)	
Evasion Attacks Implementation				Implement evasion attack algorithms. (Batu, Eren)	Test and evaluate attacks. (Aylin, İdil, Eren)	
Documentation & Reporting			Prepare the presentation. (Batu, Aylin)	Document accuracy results. (Everyone)	Write the final report. (Everyone)	Submit project & deliverables. (Everyone)

Table 1: Project Timeline and Task Allocation

6. Bibliography

[1] Z. Noury and M. Rezaei, “Deep-CAPTCHA: A Deep Learning based CAPTCHA SOLVER for vulnerability assessment,” arXiv.org, [Online]. Available: https://arxiv.org/abs/2006.08296 (accessed Dec. 25, 2024).

[2] 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," *Proc. 2018 ACM SIGSAC Conf. Computer and Communications Security (CCS '18)*, New York, NY, USA, pp. 332–348, 2018. [Online]. Available: https://doi.org/10.1145/3243734.3243754 (accessed Dec. 26, 2024).

[3] P. Samadnejad, “Captcha Dataset,” Kaggle, https://www.kaggle.com/datasets/parsasam/captcha-dataset/data (accessed Dec. 26, 2024).

[4] G. Passault, “Gregwar/CAPTCHA: PHP Captcha Library,” GitHub Repository, https://github.com/Gregwar/Captcha?tab=MIT-1-ov-file (accessed Dec. 26, 2024).

[5] A. Guna, “Large Captcha Dataset,” Kaggle, https://www.kaggle.com/datasets/akashguna/large-captcha-dataset (accessed Dec. 27, 2024).