A Customizable CAPTCHA Generation and Evaluation Platform Against Machine Learning Attacks

Shu-Yu Wu
National Taiwan University
Taiwan
b12902036@ntu.edu.tw

Long-Fong Chan
National Taiwan University
Taiwan
b10901119@ntu.edu.tw

Yun-Jhen Tu
National Taiwan University
Taiwan
41075047h@gapps.ntnu.edu.tw

ABSTRACT

CAPTCHAs are widely used to prevent automated access by bots. However, the rise of deep learning has rendered many traditional CAPTCHA schemes vulnerable. In this work, we present a machine-learning-resistant CAPTCHA defense platform with customizable parameters and a user-friendly interface. Our platform supports multiple noise types, real-time evaluation using popular OCR and CNN-based models, and provides experimental evidence demonstrating the robustness of the generated CAPTCHAs and our evaluation mechanisms.

KEYWORDS

CAPTCHA, Machine Learning Attack, GUI Platform, Adversarial Perturbation

1 INTRODUCTION

CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart) are among the most widely adopted security mechanisms on the internet. Their primary purpose is to prevent automated programs (bots) from abusing web services. From e-commerce to social media platforms, CAPTCHAs are commonly deployed in scenarios such as user registration, login verification, and spam prevention. However, with the rapid advancement of artificial intelligence (AI) and machine learning (ML), the defensive strength of CAPTCHAs is facing unprecedented challenges.

In particular, deep learning models—especially convolutional neural networks (CNNs)—have achieved superhuman performance across many computer vision tasks. As a result, text-based image CAPTCHAs have increasingly become vulnerable to these models. For instance, Yan and El Ahmad demonstrated that segmentation-resistant CAPTCHAs deployed by Microsoft could be broken with over 60% success rate by using simple image processing and segmentation heuristics [7]. Studies have shown that even simple CNN architectures can achieve decoding success rates exceeding 90%. Recent work further confirms this trend—Walia et al. proposed a CNN model named CapNet, which successfully recognized alphanumerical CAPTCHA images with over 96% test accuracy [6]. More advanced OCR (Optical Character Recognition) engines, such as Tesseract, can even decode conventional web CAPTCHAs end-to-end using pretrained weights. This poses a significant security risk to website operators, as CAPTCHAs are no longer effective at distinguishing between humans and machines.

To combat these threats, traditional CAPTCHA designers have resorted to increasing image distortion (e.g., ripples, rotation), introducing noise (e.g., dots, interference lines), and adding background complexity to raise the barrier for automated attacks. However, these methods often degrade the usability for legitimate users. Our observations show that excessive image processing leads to poor text legibility, reducing the success rate of human users and potentially causing frustration or abandonment.

Therefore, an ideal CAPTCHA system should possess the following characteristics: (1) user-friendly and easy to recognize by humans, (2) robust against machine-based attacks, (3) capable of generating diverse challenges through randomization to increase the attack difficulty, and (4) testable and quantifiable in terms of security effectiveness. In this work, we propose a CAPTCHA attack-and-defense platform that satisfies these requirements. The platform integrates an image generation module, an extensible perturbation engine, deep learning-based attack models, and a graphical user interface. It enables researchers and developers to rapidly set up experiments, observe attack behavior, and design more secure CAPTCHA variants.

Unlike prior work, which often focused solely on attacking CAPTCHAs using machine learning models or evaluating usability through large-scale user studies, our work integrates both attack and defense perspectives into a unified, interactive platform. We not only design CAPTCHAs, but also evaluate their robustness against adaptive attacks in a realistic and configurable environment. Our platform supports various perturbation types—including Gaussian noise, affine transformations, and brightness/contrast adjustments—and integrates mainstream recognition models such as Char-CNN, VGG16, and Tesseract. It provides real-time feedback, including model predictions, confidence scores, and key metrics such as Accuracy, CER (Character Error Rate), and SSIM (Structural Similarity Index), giving users a comprehensive understanding of the defense's effectiveness.

Ultimately, our goal is to deliver a reproducible, parameter-tunable, and visualization-capable framework for CAPTCHA generation and evaluation. We envision future CAPTCHA development to move beyond trial-and-error design and toward systematic, scientifically grounded approaches.

2 PROBLEM DEFINITION

2.1 Definition

The objective of this work is to design and implement a CAPTCHA system that simultaneously achieves two critical goals: (1) maintaining a high level of usability for human users, and (2) significantly reducing the success rate of machine learning-based attacks. In particular, the system should generate CAPTCHA images that are easily interpreted by humans—requiring minimal cognitive effort—while incorporating perturbations that degrade the performance of automated recognition models.

To accomplish this, we focus on the systematic generation of adversarial CAPTCHA samples through a perturbation engine that

introduces visual complexity without compromising clarity. These perturbations include random noise, geometric deformation, brightness and contrast variations, and compression artifacts. The generated CAPTCHAs should present a meaningful challenge to automated models such as CNNs and OCR engines, while remaining solvable for human participants across different ages and backgrounds.

This balance of security and usability is critical: if the CAPTCHA is too simple, it will be ineffective against automated solvers; if it is too complex, it will frustrate and deter human users. Thus, our design adheres to the principle of adversarial robustness with human-centered design.

2.2 Threat Model

Our threat model assumes a moderately capable attacker with access to hardware acceleration (e.g., GPU) and modern machine learning frameworks such as PyTorch or TensorFlow. The attacker can collect a limited dataset (e.g., through repeated access to a CAPTCHA-protected service), and may manually label it to train recognition models. However, the attacker is not assumed to have knowledge of the CAPTCHA generation parameters, perturbation configurations, or fonts used in our system.

We consider the attacker may use various types of recognition models: (1) Lightweight CNNs trained on single-character images (2) Deep models such as VGG16, fine-tuned for CAPTCHA-like datasets (3) Pretrained OCR engines such as Tesseract, which are capable of recognizing entire strings

The attacker's goal is to maximize prediction accuracy or bypass the CAPTCHA challenge with minimal effort. They may attempt brute-force decoding, adversarial training, or transfer learning. Our defense aims to limit these models' performance by generating high-entropy, dynamic CAPTCHA challenges that are difficult to generalize.

3 RELATED WORK

3.1 ML-Based Attacks

The emergence of deep learning has significantly undermined the security guarantees of traditional CAPTCHA mechanisms. Convolutional Neural Networks (CNNs), in particular, have demonstrated remarkable success in solving character recognition tasks, even under noisy or distorted conditions. Studies have shown that lightweight CNNs, when trained on modestly sized labeled datasets, can easily decode text-based CAPTCHA schemes with high accuracy [6, 9]. These models exploit the regularity of fonts, fixed character layouts, and limited visual variation in traditional CAPTCHAs.

More advanced recognition systems, such as Tesseract OCR, leverage pretraining on large-scale text corpora and can decode full-string CAPTCHAs without the need for per-character segmentation [5]. The growing accessibility of pretrained models and deep learning toolkits further lowers the barrier for adversaries to deploy effective CAPTCHA solvers.

A particularly dangerous direction involves the use of Generative Adversarial Networks (GANs). Ye et al. [8] demonstrated that GAN-based pipelines can synthesize CAPTCHA-like data and train solvers with minimal supervision, achieving high success rates against real-world deployments. This not only reduces the dependency on manual labeling but also introduces a realistic threat model wherein attackers can adapt rapidly to new CAPTCHA variants.

From a broader perspective, Dionysiou and Athanasopoulos [3] provide a comprehensive systematization of ML-based CAPTCHA breaking techniques. Their study categorizes solvers by architecture, attack surface, and generalization capabilities, and highlights how CNN-based solvers can achieve success rates up to 100% against many real-world CAPTCHAs. This reinforces the urgency of developing more robust and adaptive CAPTCHA defenses.

Despite these advances, some works argue that machine-only solvers may still struggle under certain conditions. Arp et al. [2] caution against overestimating ML robustness in security contexts, emphasizing that many adversarial systems fail under distribution shifts or noise-heavy environments. However, the trend remains clear: CAPTCHAs without adaptive, randomized, and semantically aware defenses are increasingly susceptible to ML-based attacks.

3.2 Defense Techniques

To counteract the growing capabilities of machine learning-based solvers, numerous defense strategies have been proposed in recent years. The most common approaches rely on visual obfuscation—introducing geometric distortions, noise overlays, background clutter, and font warping—to degrade the effectiveness of automated recognition models [7]. However, such techniques often come at the cost of human usability. Excessive perturbations may reduce readability for legitimate users, leading to abandonment and usability complaints [1].

Recent works have attempted to strike a better balance between robustness and usability. For instance, some researchers propose CAPTCHA generation techniques that incorporate adversarial perturbations specifically tailored to exploit weaknesses in CNNs and OCR engines [4]. These perturbations, when carefully calibrated, can significantly reduce model accuracy without compromising human interpretation.

Moreover, certain studies have explored the use of semantic- and context-aware CAPTCHAs, such as object matching, visual reasoning, or hybrid tasks [8]. Although effective, these methods are often computationally expensive and hard to scale.

From a systematization perspective, Dionysiou and Athanasopoulos [3] highlight several effective design features to enhance defense: introducing scale variance, maximizing character-level ambiguity without compromising segmentation resistance, and enforcing diverse font and layout generation. Our platform adopts many of these principles by offering modular control over noise type, intensity, and generation diversity.

In addition to visual defenses, behavioral-based approaches have been proposed to detect human-in-the-loop attacks. For example, Alreshoodi et al. [1] suggest the use of keystroke dynamics to distinguish legitimate users from hired solvers. While promising, such defenses require additional data collection and may raise privacy concerns.

Ultimately, CAPTCHA defense must evolve into a multi-faceted process—combining visual obfuscation, semantic reasoning, behavioral signals, and adversarial training. Our work contributes to this evolution by providing a flexible and testable defense framework

that allows experimentation across diverse threat models and perturbation strategies.

4 DESIGN AND IMPLEMENTATION

Our platform is implemented with a graphical user interface (GUI) using Streamlit, providing an interactive and user-friendly environment for CAPTCHA generation and evaluation. The system is composed of modular components, each responsible for a specific function in the CAPTCHA attack-defense workflow.

- **CAPTCHA Generation:** Users can generate CAPTCHA images using a wide variety of fonts, with support for multiple characters per image. The system supports different types of perturbations, including:
  - **Pixel-level noise:** Gaussian, Laplace, Salt-and-Pepper, Speckle
  - **Geometric distortions:** Rotation, affine transformation, cutout

Algorithm 1 Affine Shift Perturbation

Get image width w and height h
Sample $dx \sim U(-\delta \cdot w, \delta \cdot w)$
Sample $dy \sim U(-\delta \cdot h, \delta \cdot h)$
Define affine matrix $M = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \end{bmatrix}$
Apply affine transform to I using M and return I'

  - **Color compression:** Brightness/contrast adjustment, JPEG compression
- **Model Evaluation:** Users can evaluate the robustness of CAPTCHAs using:
  - Character-level CNNs trained on single characters (e.g., Char-CNN)
  - Deep CNNs (e.g., VGG16)
  - Tesseract OCR for full-string recognition
- **Metrics and Visualization:** The platform displays real-time metrics including Accuracy, Character Error Rate (CER), Structural Similarity Index (SSIM), Peak Signal-to-Noise Ratio (PSNR), and Tesseract confidence scores. Visual results and prediction outputs are shown side by side for comparison.

This interactive design allows users to iteratively generate, test, and refine CAPTCHA strategies against different types of machine learning models.

4.1 Noise Types

To increase the difficulty of automated attacks while preserving human readability, we apply a range of visual perturbations to CAPTCHA images. These perturbations are modular, parameterizable, and designed to target the weaknesses of deep learning models in image recognition. The supported noise types in our system include:

Algorithm 2 Gaussian Noise Perturbation

Convert I to float array A with shape (H, W, C)
Generate $N \sim \mathcal{N}(\mu, \sigma^2)$ of same shape as A
Set $A' = \text{clip}(A + N, 0, 255)$
Convert A' to uint8 and return image I' from A'

Gaussian Noise: Adds normally distributed random values to each pixel. This simulates low-light image noise and can distort model perception without severely affecting human recognition.

Algorithm 3 Salt-and-Pepper Noise Perturbation

Let $N = H \times W \times C$
Compute number of salt pixels $n_s = \lceil \alpha \cdot N \cdot r \rceil$
Sample n_s random coordinates and set pixels to 255 (salt)
Compute number of pepper pixels $n_p = \lceil \alpha \cdot N \cdot (1 - r) \rceil$
Sample n_p random coordinates and set pixels to 0 (pepper)
Return perturbed image I'

Salt-and-Pepper Noise: Randomly turns some pixels white ("salt") or black ("pepper"), simulating impulse noise often seen in corrupted image transmission.

Cutout (Masking): Removes square or rectangular regions in the image by masking them, forcing the model to infer characters with incomplete visual context.

Brightness and Contrast Adjustment: Simulates lighting variations by scaling pixel values. This directly impacts edge visibility and model confidence.

JPEG Compression Artifacts: Applies lossy compression to the image to introduce pixel-level distortion. This is particularly effective against models trained on clean images.

By allowing users to configure each type of noise via parameters (e.g., noise intensity, region size), the system can generate diverse and realistic CAPTCHA challenges tailored to specific threat levels.

4.2 Evaluation Models

To evaluate the robustness of generated CAPTCHA samples, we integrate three types of recognition models representing different levels of attacker sophistication:

Char-CNN: A lightweight convolutional neural network trained on single-character images. It is optimized for fast inference and demonstrates the baseline capacity of ML models with minimal resources.

VGG16: A deeper CNN architecture capable of learning richer feature representations. Though slower and more resource-intensive, it provides higher accuracy and serves as a stronger attack model.

Tesseract OCR: An open-source, pretrained OCR engine that recognizes full strings rather than individual characters. It returns confidence scores for each prediction, enabling a more granular evaluation.

These models are used to simulate realistic attackers with varying skill levels and tools. By comparing results across models, we gain insights into which perturbation strategies are effective across different attack types.

5 SYSTEM INTEGRATION AND USAGE

The platform consists of the following main modules:

- `app.py`: An interactive CAPTCHA generation and evaluation interface built with Streamlit, supporting multi-character CAPTCHAs, perturbation parameter settings, and model inference.

- `Perturber.py`: A library of various image perturbation functions that add multiple types of noise and geometric transformations to CAPTCHA images.
- `torch_char_cnn.py`: A PyTorch-based single-character CNN training script, supporting both CPU and GPU execution.
- `evaluate.py`: Script for model inference and performance evaluation, including comparison with Tesseract OCR.
- `metrics.py`: Functions to compute evaluation metrics such as Accuracy and Character Error Rate (CER).
- `data.py`: Generates CAPTCHA image datasets for training and testing.

Usage instructions:

(1) Run **torch_char_cnn.py** to train the model and save the weights as **char_cnn.pt**.
(2) Run `app.py` by executing python -m streamlit run app.py to launch the local web interface.
(3) In the interface, configure the CAPTCHA generation parameters and perturbation intensity, and select the recognition model to test (Char-CNN, VGG16, or Tesseract OCR).
(4) Observe model prediction results, confidence scores, and various evaluation metrics, and perform data download and analysis.

6 EXPERIMENT AND RESULTS

6.1 Dataset

To train and evaluate our models, we generate a synthetic dataset of CAPTCHA images using randomized parameters and fonts. Each CAPTCHA contains either a single character or a sequence of alphanumeric characters (0–9, a–z). The generation process ensures diversity in font type, character position, noise level, and background structure.

Training Set: 5,000 images per model, generated under clean conditions to establish baseline performance.

Evaluation Set: 500 images per model, generated under five perturbation levels—Clean, Weak, Moderate, Strong, and Extreme.

These synthetic datasets provide full control over the experimental settings and allow us to systematically evaluate each perturbation's impact on different models.

6.2 Metrics

To comprehensively evaluate the performance of our CAPTCHA recognition models and the robustness of the platform, we adopted multiple metrics:

Accuracy (ACC): This measures the proportion of correctly recognized CAPTCHA samples out of the total tested. It provides a straightforward measure of model correctness in classification tasks.

Character Error Rate (CER): Calculated as the edit distance (Levenshtein distance) between the predicted and ground truth text divided by the length of the ground truth string. CER is particularly important in sequence recognition tasks, as it quantifies character-level mistakes, including substitutions, insertions, and deletions.

6.3 Results

Our experimental results indicate that under conditions of strong and extreme noise, all evaluated machine learning models show a degradation in accuracy. However, they still maintain about 60 to 70% ACC, which makes them trustworthy evaluation tools.

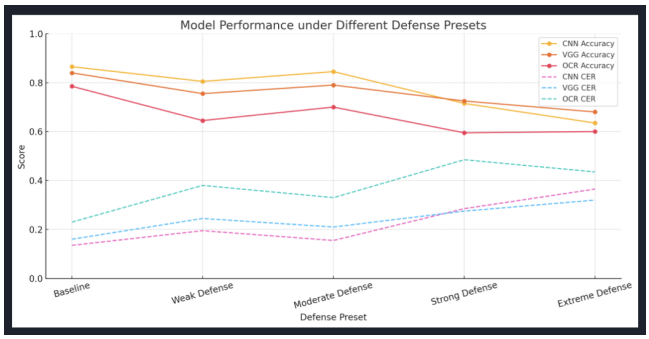

Figure 1: Model Evaluation

After we integrate them into our platform, the result shows that all three models do not perform well with our self-made CAPTCHAs. In contrast, we did a small survey about human usability, which gives a 93.2 score by the participants.

7 CONCLUSION AND FUTURE WORK

In this work, we presented an extensible CAPTCHA generation and evaluation platform that supports various noise-based and geometric perturbations, enabling robust testing of recognition models against adversarial conditions. Our platform demonstrates effective defenses against current state-of-the-art machine learning attacks, underscoring the importance of adaptive CAPTCHA design.

For future improvements, we plan to:

- Incorporate sequence-based deep learning models such as Long Short-Term Memory (LSTM) networks and Convolutional Recurrent Neural Networks (CRNN), which are known for their superior performance in sequence recognition tasks.
- Integrate real-world CAPTCHA datasets collected from widely used services like Google and Baidu to enhance the practical relevance and robustness of the platform.
- Extend support to additional CAPTCHA types beyond text-based ones, such as image-based, audio, or puzzle CAPTCHAs, to cover a broader spectrum of security challenges.
- Explore perturbation techniques that exploit the discrepancy between human and machine perception. In particular, we aim to move beyond pixel-level noise by introducing frequency-domain perturbations (e.g., modifying high-frequency components via Discrete Fourier Transform) that affect deep learning models but remain imperceptible to human users. This direction emphasizes not just increasing image complexity, but also introducing adversarial semantics—images that machines recognize incorrectly or confidently, yet remain ambiguous or unreadable to humans.
- Investigate the use of machine-deceptive patterns that are semantically misleading, such as adversarial textures, spectral filtering, or feature-space manipulation. These methods may complement conventional noise injection by targeting

model-specific weaknesses while preserving human usability.

ACKNOWLEDGMENT

We gratefully acknowledge the support and resources provided by the Department of Computer Science and Information Engineering at National Taiwan University (NTU), which were instrumental in the completion of this project.

REFERENCES

[1] Latifah A. Alreshoodi and Suliman A. Alsuhibany. 2020. A Proposed Methodology for Detecting Human Attacks on Text-based CAPTCHAs. *International Journal of Engineering Research and Technology (IJERT)* 13, 4 (2020), 625–630.

[2] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and Don'ts of Machine Learning in Computer Security. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, 3881–3898. https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[3] Antreas Dionysiou and Elias Athanasopoulos. 2020. SoK: Machine vs. Machine – A Systematic Classification of Automated Machine Learning-based CAPTCHA Solvers. *Computers & Security* 97 (2020), 101947. https://doi.org/10.1016/j.cose.2020.101947

[4] Yu-Kai Huang, Tsung-Han Wu, and Wu-Jun Pei. 2021. Defense against Machine Learning based CAPTCHAs Attack. Undergraduate Project Report, National Taiwan University. Unpublished.

[5] Andrew Searles, Yoshimichi Nakatsuka, Ercan Ozturk, Andrew Paverd, Gene Tsudik, and Ai Enkoji. 2023. An Empirical Study & Evaluation of Modern CAPTCHAs. *arXiv preprint arXiv:2307.12108* (2023). https://arxiv.org/abs/2307.12108

[6] Jaskaran Singh Walia and Aryan Odugoudar. 2023. Vulnerability Analysis for CAPTCHAs Using Deep Learning. In *2023 International Conference on ICT in Business Industry Government (ICTBIG)*. IEEE, 1–8. https://doi.org/10.1109/ICTBIG59752.2023.10456218

[7] Jeff Yan and Ahmad Salah El Ahmad. 2008. A Low-cost Attack on a Microsoft CAPTCHA. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS)*. ACM, Alexandria, Virginia, USA, 543–554. https://doi.org/10.1145/1455770.1455837

[8] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen, and Zheng Wang. 2018. 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)*. ACM, Toronto, Canada, 332–348. https://doi.org/10.1145/3243734.3243754

[9] Nathan Zhao, Yi Liu, and Yijun Jiang. 2017. CAPTCHA Breaking with Deep Learning. CS229 Final Project Report, Stanford University. https://cs229.stanford.edu/proj2017/final-reports/5239112.pdf Unpublished student report.