TEXT BASED CAPTCHA THE GUIDELINES TO ENHANCE THE ROBUSTNESS OF CAPTCHA USING CHOSEN PLAIN TEXT ATTACK METHOD

R. Menaka[1], *R. Vijayabhanu*[2]

ABSTRACT

In general, websites face a variety of security issues. To address these challenges, web pages are protected by specific security measures. One such measure is CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart), a technique used to identify human users and allow them to proceed within the selected webpage.

There is ongoing research aimed at distinguishing between human users and machines. To deceive machine learning models used in CAPTCHA systems, a common observation is: "The higher the cracking rate, the poorer the quality of the CAPTCHA." These results contribute to refining the design guidelines of CAPTCHA systems. Cracking CAPTCHAs using chosen plaintext attacks and analyzing the experimental results can help in designing more robust CAPTCHA mechanisms.

Keywords : CAPTCHA, Security, Chosen Plaintext Attack

I. INTRODUCTION

CAPTCHA is a Turing test tool that distinguishes the human user and machine user from their response. In the 2003, Von Ahn et al and colleagues introduced the term CAPTCHA and its usage in website security metrics. CAPTCHA can be categorized into four major aspects, i. Text based, ii. Image based, iii. Audio based, iv. Video based. Though there are many literature reviews in the above-mentioned types of CAPTCHAs, Text and Image based CAPTCHAs are widely used in prominent applications. A Chosen Plaintext Attack (CPA) is a type of cryptographic attack where the attacker can choose arbitrary plaintexts and obtain the corresponding cipher texts. This model is important in both cryptanalysis and security research, especially for evaluating the strength of encryption schemes and systems like CAPTCHA.

Department of Computer Science[1]
Karpagam Academy of Higher Education,
Coimbatore- 641021, Tamil Nadu[1]
menaka.rasan@kahedu.edu.in[1]

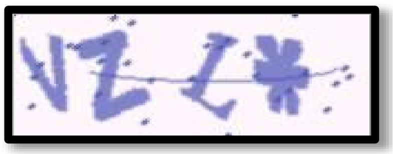
Department of Computer Science[2]
Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore - 641043, Tamil Nadu[2]
vijayabhanu_cs@avinuty.ac.in[2]

* Corresponding Author

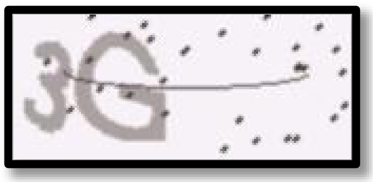
Table 1: Types of Text-Based CAPTCHA

Category	Example	Source	Features
SOLID CAPTCHA		Discuz!	Character independent, texture background, some interference
		Slashdot	A large number of interference lines and noise points
		Google	Unfixed length, distortion, adhesion
		Microsoft	Double-string, unfixed length, uneven thickness, tilting, adhesion
HOLLOW CAPCTHA		QQ	Hollow, shadows, interference shapes
		Sina	Hollow, adhesion, interference lines
		Yandex	Hollow, Virtual contours, distortion, adhesion, interference lines
3D CAPTCHA		Scihub	Hollow, shadows, interference lines, noise points
		Parc	Colorful, protrusion, distortion, background and character blending
ANIMATI-ON CAPTCHA		Program. Generate	Multiple characters jumping
		HCaptch a	Multilayer character images blinking transformation

The type of text-based CAPTCHA used in this study is Hollow CAPTCHA with distortions, like overlapping, skewing and interference line. CAPTCHA's are running into different evolutionary progress with the help of Cryptography and Digital Watermarking. One of the main feature of hollow CAPTCHA is the use of inference lines which looks like connecting the next-to-next characters, which may confuse the malicious learning machine in training phase. In the study [1], mentioned that cracking rate of Hollow CAPTCHA is lesser compared to the CAPTCHA scheme used in the popular websites like Yahoo, Tencent, CmPay, Sina and Baidu.



a) Alphanumeric CAPTCHA with special characters,



b) Alphanumeric CAPTCHA

Fig. 1. Hollow CAPTCHA generated using the open-source python CAPTCHA library.A chosen-plaintext scenario turns text CAPTCHAs into a supervised learning problem. If attackers can generate our challenges with known answers, modern models will learn them. Our study focus on how to break that assumption— hide/rotate the generator distribution, add secret-seeded randomness, rate-limit challenge harvesting, and layer additional signals—or choose alternatives that don't present a learnable, label-rich oracle.

II. LITERATURE REVIEW

A. Related Works: CAPTCHA its Early Stage to till now:

In 2005, Chellapilla and Simard [1] investigated the vulnerability of visual CAPTCHAs to machine learning attacks. Their study focused on the recognition component of CAPTCHA-solving and highlighted that machine learning models, at the time, performed significantly worse than human users. Based on this observation, they proposed that machine learning could serve as a suitable framework for evaluating recognition tasks, which subsequently became a foundation for many CAPTCHA designs. As a result, they developed effective CAPTCHA models, such as those adopted in MSN Passport, which relied on complex character-segmentation tasks to hinder automated solvers. Building on alternative approaches, Kluever [2] in 2008 introduced a video-tagging CAPTCHA that required users to provide descriptive labels for short video clips drawn from public platforms like YouTube. In experiments involving over 180 participants, the system demonstrated strong usability, with 70–90% of human users successfully completing the challenge by generating three descriptive words, compared to only 13% success among bots. Despite its promise, the approach faced practical challenges, including reliance on tag frequency data and susceptibility to network-based misrepresentation.

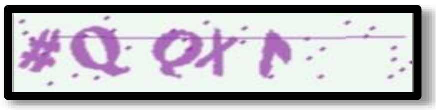
In 2014, Jaderberg et al. [9] proposed a pioneering model for text spotting in natural images, which directly influenced CAPTCHA recognition research. Their system divided the task into detecting word boundaries and recognizing text within those boundaries, employing a novel CNN architecture for both classification and detection. The architecture supported case-sensitive and case-insensitive recognition at the character and bigram level, while incorporating automated data mining from Flickr to provide large-scale annotations. By combining these elements, they developed an end-to-end text spotting framework that achieved state-of-the-art results on standard benchmarks, including the ICDAR Robust Reading dataset and the Street View Text dataset. Building on the effectiveness of CNNs, Kopp, Nikl, and Holena [8] in 2017 introduced a CAPTCHA recognition approach that was able to break 10 out of 11 CAPTCHA schemes using two CNNs or a hybrid of a localization perceptron with a recognition CNN, even without scheme-specific training data. When trained with scheme-generated instance images, their approach successfully cracked all 11 CAPTCHA schemes, achieving over 50% accuracy. Their experiments comparing Single hidden-Layer Perceptrons (SLP), Multi hidden-Layer Perceptrons (MLP), and CNNs demonstrated that CNNs consistently outperformed other architectures in both localization and recognition. Extending this trend, Bostik and Klecka [7] in 2018 evaluated multiple machine learning methods on a database of 4,950 synthetic CAPTCHA characters generated by a PHP-based system. Their results indicated that feed-forward neural networks outperformed K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and other OCR models, reinforcing the consensus that neural networks are among the most effective methods for CAPTCHA recognition. More recently, researchers have explored advanced techniques to counter increasingly powerful automated solvers. For instance, Diff-CAPTCHA [3] employed denoising diffusion models to fuse text and background features, making segmentation significantly harder for recognition systems while preserving usability. Similarly, neuroimaging-based studies [4] investigated human cognitive processing of distorted CAPTCHAs using functional near-infrared spectroscopy (fNIRS), revealing that distortions such as rotation and occlusion activate Gestalt-based perception mechanisms that aid human recognition. In parallel, fine-grained distortion strategies combining overlap, scaling, rotation, and noisy backgrounds reduced machine recognition accuracy to below 5% [5]. At the same time, benchmarking frameworks such as MCA-Bench [6] provided systematic evaluations of CAPTCHAs against vision–language model attacks, highlighting vulnerabilities and establishing principles for robust multimodal challenge design. Building on this trajectory, Aura-CAPTCHA [7] combined GAN-based image generation, reinforcement learning, and large language models to deliver adaptive, multimodal challenges. This system achieved human success rates above 90% while reducing automated solver success to only 10%, marking a significant advancement over traditional text-based CAPTCHA mechanisms.

III. DATASET

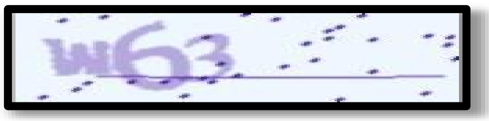
Dataset Description: The dataset used in this research is categorized into two main types: Internal Dataset and External Dataset.

i. The Internal Dataset consists of unlimited ciphertexts (CAPTCHAs) generated by the CAPTCHA generator under a chosen-plaintext attack framework. This dataset is designed to include a total of 42 characters, comprising 24 alphabets (excluding the visually ambiguous characters O and I), 10 numeric digits (0–9), and 8 special characters (*! @ # $ % ^ &

ii. The ciphertext samples are generated iteratively, with each iteration producing a new CAPTCHA instance. For this study, between 8,000 and 10,000 iterations were employed to construct the internal dataset used for model training and evaluation.



a) Alphanumeric CAPTCHA with special characters



b) Alphanumeric CAPTCHA

Fig. 2 : Internal Dataset: CAPTCHAs generated using a chosen- plaintext attack.

ii. The External Dataset is employed to evaluate the model and compare its performance against the internally generated dataset. Based on related literature, two publicly available sources were selected: the Hashkiller dataset and the Delta Airlines dataset. The Hashkiller dataset contains 13 CAPTCHA image files of length 6, with a mixture of alphanumeric and special characters. In contrast, the Delta Airlines dataset comprises 40 CAPTCHA image files of length 5, consisting exclusively of numeric characters. This diversity allows the external dataset to serve as a robust input source for the evaluation process, complementing the internal dataset and ensuring a comprehensive assessment of the proposed model.

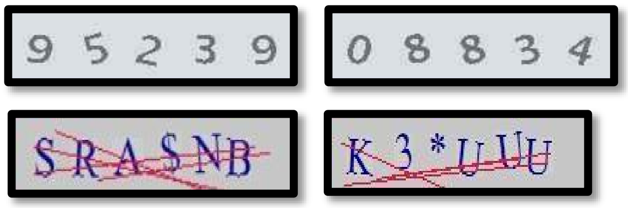


Fig. 3: External Dataset using Delta Airlines – Numeric CAPTCHA with Special characters

IV. METHODOLOGY

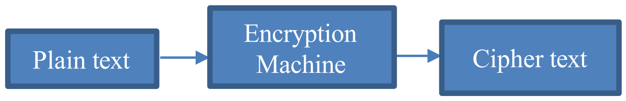
A. Distorted CAPTCHA

In this phase, the input CAPTCHA is extracted using a chosen- plaintext attack, wherein an encryption machine converts.

In the study of CAPTCHA schemes, two major categories can be identified: proprietary CAPTCHA schemes and open- source CAPTCHA libraries. A well-known example of the first category is *Google reCAPTCHA*, whereas the second category includes libraries such as the *Python Claptcha library*. This paper primarily focuses on the latter, as open-source CAPTCHA libraries have gained popularity in both commercial and non-profit organizations, including platforms like *Delta Airlines* and *MITBBS*, where they serve as general-purpose Internet service authentication mechanisms.

Proprietary CAPTCHA schemes typically enhance security by adopting advanced segmentation techniques, such as space reduction and character overlap in the Google reCAPTCHA algorithm. However, in recent years, even these schemes have been compromised by machine learning–based bots. Open- source CAPTCHA libraries, in particular, face greater security risks because their transparency allows malicious actors to exploit them for training learning models. This provides an avenue for launching a chosen-plaintext attack, where an adversary can leverage the unlimited generation of well-labeled cipher texts from the CAPTCHA generator (encryption machine) to train malicious learning models. Such vulnerabilities highlight the need for designing robust text- based CAPTCHAs that can withstand attacks from modern machine learning systems.Cipher text

Encryption Machine



Flowchart.1. Chosen plain text attack – Encrypted Machine (CAPTCHA Generator)

B. Preprocessing Image Binarization:

In this stage, the colored CAPTCHA image is converted into a binary image consisting of only two pixel values: black and white. This process enhances the contrast between the foreground (text characters) and the background, thereby simplifying subsequent preprocessing steps such as noise removal, contour line elimination, and character segmentation.

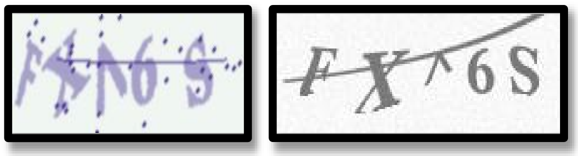
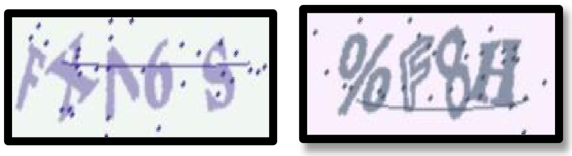
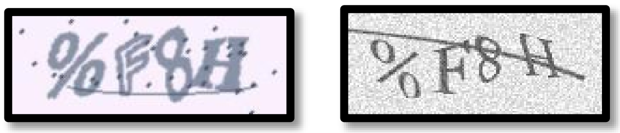


Fig. 4. CAPTCHA in Image binarization



(a) Alphanumeric CAPTCHA (b) CAPTCHA with special char

Fig. 5. Distorted CAPTCHAs generated using the open-source Python CAPTCHA library.



(a) Alphanumeric CAPTCHA (b) CAPTCHA with special char

Fig. 6. Distorted CAPTCHA after Preprocessing

This encryption machine functions as a CAPTCHA generator and is capable of producing unlimited text-based CAPTCHA images with the support of the open-source Python CAPTCHA library. The generated CAPTCHA characters are displayed with distortion features such as overlapping, skewing, interference lines, and background dots, which increase the complexity of recognition and enhance robustness against automated attacks

C. Character Extractor

After preprocessing the CAPTCHA image, the character profiles are extracted from the source code of the Python CAPTCHA library. The distortion character profile records essential parameters such as image position, height, width, axis, and location, which are mapped using distortion features like distance, direction, shape, and angle.

Table 2: Parameter used in CAPTCHA Character profile tracker

S.No	Variable	Data type
1.	Index	Integer
2.	Label	Character Array
3.	Height	Integer Array
4.	Width	Integer Array
5.	X-Axis	Integer Array
6.	Y-Axis	Integer

The Index represents the order of extracted characters, while the Label stores the actual character value. Height and Width record the dimensions of each character's bounding box. The X-Axis and Y-Axis parameters specify the spatial location of the character within the CAPTCHA image, ensuring accurate mapping for subsequent CNN training.

In this phase, a feed-forward batch system is employed with the restriction that it only accepts uppercase alphanumeric characters (excluding the characters *O* and *I*, since they closely resemble *0* and *1*). Each character is carefully tracked to ensure accurate extraction, and the corresponding text along with the updated parameters is recorded. The resulting dataset forms the suitable input for the Convolutional Neural Network (CNN) used in the training phase.

D. Segmentation

In this research, TensorFlow Object Detection (TOD) is employed to identify the positions of characters within CAPTCHA images. Based on insights from the related literature on CAPTCHA cracking methods, TOD is considered one of the primary approaches for the segmentation process.

Initially, each CAPTCHA image is converted into a binary format, which enhances the contrast and preserves the positions of the characters. The image is then analyzed across two dimensions: the X-axis (representing horizontal position) and the Y-axis (representing vertical position). For each position along the X-axis, the corresponding sum of vertical pixels is calculated. These pixel intensity distributions are used to detect character boundaries.

Finally, the characters are segmented according to their positional indices, producing a structured list of character regions. These segmented character profiles are then used as inputs for the subsequent recognition stage.

Input: Binary CAPTCHA image
Output: List of segmented character positions

1. Initialize parameters: Create an empty list Character_ Index to store segmented positions. Set Start_Position = 0, Set_Read = False, and Char_Count = 0.
2. Compute average projection: Generate the vertical pixel projection array Y_axis from the binary image. Replace each value in Y_axis as follows: if the value is greater than or equal to the average of Y_axis, keep it unchanged; otherwise, set it to 0.
3. Scan through Y-axis projection: For each position char_index in Y_axis, check the following conditions. If the value is not equal to 0, reset Char_Count = 0. If Set_Read = False, mark the beginning of a new character by setting Start_Position = char_index and update Set_Read = True. Otherwise, if the value is 0, increment Char_Count. If Set_Read = True, append (Start_Position, char_index) to Character_Index, then reset Set_Read = False and Char_Count = 0.
4. Return segmented positions: Output the list Character_Index, containing the start and end indices of each character.

TensorFlow Object Detection (TOD):

TensorFlow Object Detection (TOD) is a computer vision technique used to identify objects in terms of their location and to track them from still images or video sequences. It is classified under machine learning tools that are designed to develop and train object detection models. With the help of TOD, a better understanding of model performance can be achieved using the Data Flow Graph, which explains the importance and flow of each data element in the network. TOD primarily deals with four parameters: (i) Tensor, (ii) Shape, (iii) Type, and (iv) Session and Operators. Furthermore, it integrates effectively with Convolutional Neural Networks (CNNs), enabling the construction of flexible architectures for object detection tasks.

TOD in the Research Work

In this research, TOD is employed as a character detection model by using the Google Object Detection module within the TensorFlow source library. The model was trained over 10,000 iterations, and the analysis of the loss function revealed six distinct types of losses:

1. Grouping Loss – The loss between the set of detected characters and their expected grouping.
2. Positional Loss – The loss between each individual character and the bounding box covering it.
3. GP Loss – A combined loss that merges Grouping Loss and Positional Loss.
4. Bounding Object Loss – The loss incurred during the grouping process within bounding boxes.
5. Net Loss - The aggregated loss combining Grouping, Positional, GP, and Bounding Object losses.
6. Replicate Loss - A replicated form of the Net Loss, ensuring consistency across training iterations.

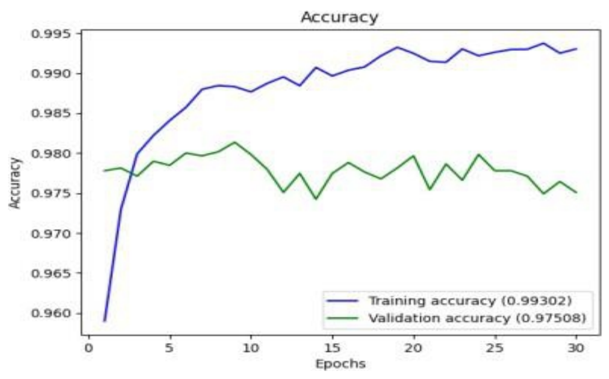
E. Recognition

The cracking rate of a CAPTCHA system can be significantly increased if the breaking framework is well normalized. The breaking flow typically follows three main stages- preprocessing, segmentation, and recognition-which are applied based on the character portfolio of the CAPTCHA dataset. Since the character paradigm may vary across schemes, different combinations of breaking flows are often required. For instance, text-based CAPTCHAs frequently employ Crowded Characters Together (CCT) as a security feature, which complicates the recognition task. Depending on the complexity, the breaking process can adopt one of the following approaches:

1. Preprocessing and Recognition only.
2. Preprocessing, Recognition, and Post-processing.
3. Preprocessing, Segmentation, Combination, and Recognition.
4. Preprocessing, Segmentation, Combination, Recognition, and Post-processing.

To strengthen resistance against machine recognition, several design features are commonly incorporated in text-based CAPTCHA systems:

- The length of the CAPTCHA character set is directly proportional to its resilience against brute-force attacks.
- The use of overlapping, distortion, and adhesion increases the difficulty of segmentation.
- Introducing variations in size, angle, width, location, and font style reduces recognition accuracy.
- Employing strings of variable length increases the complexity of CAPTCHA breaking.
- Manipulating the color and shape of the background makes noise removal more challenging.



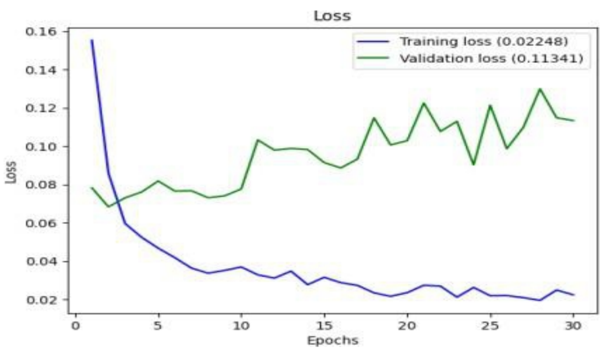


Fig. 8: Sample training and validation graph that shows the accuracy and loss of CNN model5. 40

V. RESULTS AND DISCUSSION

The integration of TensorFlow Object Detection (TOD) with Convolutional Neural Networks (CNN) proved highly effective in cluttering and tightening the bounding boxes around characters extracted from both internal and external datasets. This combination significantly improved segmentation accuracy, thereby enhancing the character cracking rate, as illustrated in Fig. 8 Furthermore, the evaluation of recognition performance was supported through confusion rate analysis. The confusion rate was computed using the formula:

i= TOTAL FREQUENCY OF CHARACTER / 2

If the frequency of incorrect recognition exceeds *i*, then the Confusion Rate = TRUE (1); otherwise, Confusion Rate = FALSE (0).

The overall confusion rate charts, presented in Figs. 9 and 10, provide a summary of the recognition errors and highlight the most frequently misclassified characters. This analysis confirms that while TOD+CNN achieved a higher cracking accuracy compared to standalone approaches, certain characters with similar visual features (e.g., *5 vs. S*, *O vs. 0*) still posed challenges in recognition.

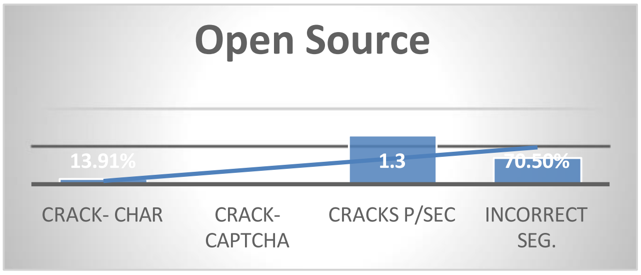


Fig. 9 : Evaluation metrics illustrating the cracking rate performance on the Internal Dataset

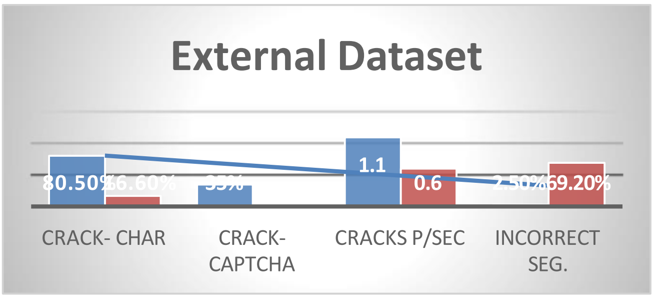


Fig. 10 : Evaluation metrics illustrating the cracking rate performance on the External Dataset40

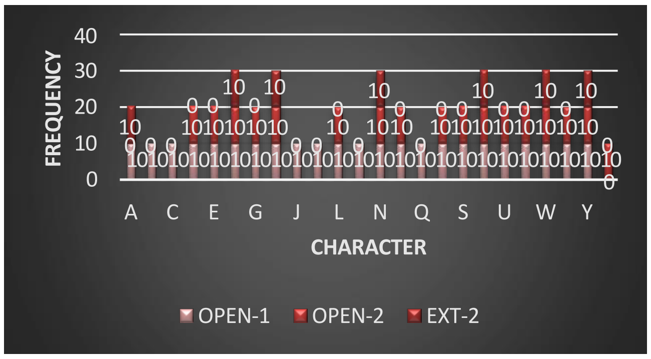


Fig. 11: Confusion Rate chart summary for 24 Alphabet character

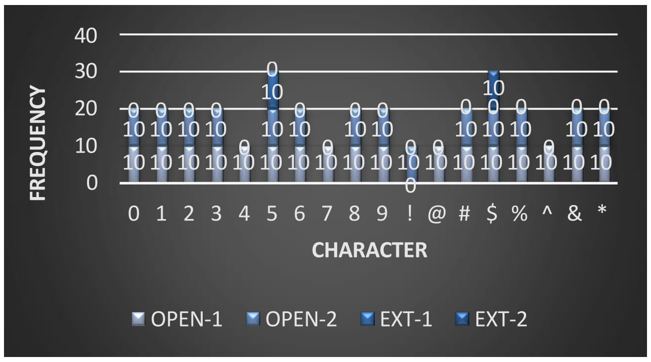


Fig. 12 : Confusion Rate chart summary for 18 Numeric character and Special Character

Guideline

The analysis of confusion rates (Figs. 11 and 12) indicates that characters such as ‘F’, ‘H’, ‘N’, ‘T’, ‘W’, ‘Y’, ‘5’, and ‘$’ are consistently misclassified due to their structural similarities with other characters or symbols. To improve the robustness of text-based CAPTCHA systems, it is recommended to either exclude these ambiguous characters from the CAPTCHA character pool or modify their visual representation by applying distinctive font styles, spacing adjustments, or distortion patterns that reduce confusion. Furthermore, introducing adaptive CAPTCHA schemes that dynamically avoid problematic character combinations could further minimize misclassification. These revisions will strengthen the resistance of CAPTCHA systems against automated machine- learning attacks while maintaining usability for human users. Conclusion-and-Limitations: In this paper, the work concentrates on testing the open-source CAPTCHA library that generates unlimited text-based CAPTCHAs through chosen plain text attacks. The existing cracking technique using the TensorFlow Object Detection method demonstrates strong adaptability in handling text cluttering during the segmentation phase. By incorporating Deep Learning techniques, the image patterns are effectively trained and recognized, enabling the exploration of semantic details within the dataset and extending the ability to generalize across new patterns. This approach contributes to optimizing the overall model performance.

VI. CONCLUSION

Limitations: Despite the progress, challenges remain in defending against advanced cracking strategies. Future AI-based CAPTCHA systems must adopt essential guidelines to counter emerging threats. In particular, the development of unpredictable and dynamic CAPTCHA techniques will be crucial to fool machine learning bots and ensure robustness against automated attacks.

Future Scope: Building upon this research, several directions can be explored to design more resilient CAPTCHA systems. Incorporating Generative Adversarial Networks (GANs) could introduce real-time variability in CAPTCHA generation, making it harder for bots to learn predictable patterns. Multimodal CAPTCHA approaches that combine text, image, and audio elements may enhance security while maintaining usability. Additionally, integrating reinforcement learning models to dynamically adjust CAPTCHA complexity based on user interactions could provide a balance between security and accessibility. Finally, collaboration with large-scale datasets and benchmarking platforms will further validate the robustness of proposed CAPTCHA mechanisms in real-world applications.

REFERENCES

[1] K. Chellapilla and P. Simard, "Using machine learning to break visual human interaction proofs (HIPs)," *Advances in Neural Information Processing Systems*, vol. 18, 2005.

[2] K. A. Kluever, "Evaluating the usability and security of a video CAPTCHA," in *Proc. 26th Annu. SIGCHI Conf. Human Factors in Computing Systems*, 2008.

[3] X. Zhou, H. Liu, Y. Zhang, and J. Wang, "Diff-CAPTCHA: A denoising diffusion model based text CAPTCHA generation method," *arXiv preprint arXiv:2308.08367*, 2023.

[4] Y. Zhang, J. Li, and F. Chen, "Neuroimaging analysis of human cognitive processing of distorted CAPTCHAs using fNIRS," *arXiv preprint arXiv:2311.18436*, 2023.

[5] Q. Li, Z. Sun, and M. Wu, "Experimental research on text CAPTCHA of fine-grained security features," *Journal of Information Security Research*, vol. 14, no. 2, pp. 85–97, 2024.

[6] H. Wang, Y. Zhao, and T. Xu, "MCA-Bench: Benchmarking multimodal CAPTCHAs against vision–language model attacks," *arXiv preprint arXiv:2506.05982*, 2025.

[7] L. Chen, P. Gao, and R. Huang, "Aura-CAPTCHA: A multimodal CAPTCHA system with GANs, reinforcement learning, and large language models," *arXiv preprint arXiv:2508.14976*, 2025.

[8] M. Kopp, M. Nikl, and M. Holena, "Convolutional neural networks for solving CAPTCHA," in *Proc. Int. Conf. Artificial Neural Networks (ICANN)*, 2017, pp. 211–220.

[9] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman, "Reading text in the wild with convolutional neural networks," *Int. J. Comput. Vis.*, vol. 116, no. 1, pp. 1–20, 2016 (original work 2014, extended journal version).

[10] P. Bostik and J. Klecka, "Evaluation of machine learning methods for CAPTCHA recognition," in *Proc. Int. Conf. Computer Science and Information Systems (FedCSIS)*, 2018, pp. 347–355.

[11] L. von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Commun. ACM*, vol. 47, no. 2, pp. 56–60, Feb. 2004.

[12] J. Chen, X. Luo, Y. Guo, Y. Zhang, and D. Gong, "A survey on breaking techniques of text-based CAPTCHA," *Security and Communication Networks*, vol. 2017, Art. ID 6898617, Nov. 2017.

[13] Y. W. Chow, W. Susilo, and P. Thorncharoensri, "CAPTCHA design and security issues," in *Advances in Cyber Security: Principles, Techniques, and Applications*. Berlin, Germany: Springer, 2019, pp. 69–92.

[14] N. Yu and K. Darling, "A low-cost approach to crack Python CAPTCHAs using AI-based chosen plain text attack," in *Applied Sciences*, vol. II, S. Tang and M. King, Eds. Xi'an, China: Jiaoda Press, 1998, pp. 158–176.

[15] K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (HIPs)," in *Advances in Neural Information Processing Systems*, vol. 17, pp. 265–272, 2005.

[16] K. A. Kluever, "Video CAPTCHAs: usability vs. security," in *Proc. IEEE Workshop on Image Processing*,

2008, pp. 1–4.

[17] O. Bostik and J. Klecka, "Recognition of CAPTCHA characters by supervised machine learning algorithms," *IFAC-PapersOnLine*, vol. 51, no. 6, pp. 208–213, 2018.

[18] M. Kopp, M. Nikl, and M. Holeňa, "Breaking CAPTCHAs with convolutional neural networks," *CEUR Workshop Proc.*, vol. 1885, pp. 93– 99, 2017.

[19] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Zurich, Switzerland, Sept. 6–12, 2014, pp. 512–528.

[20] 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," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, Toronto, Canada, Oct. 15–19, 2018, pp. 332–348.

[21] Y. Zhang, H. Gao, G. Pei, S. Luo, G. Chang, and N. Cheng, "A survey of research on CAPTCHA designing and breaking techniques," in *Proc. 2019 IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Big Data Sci. Eng. (BigDataSE), 2019, pp. 1553–1560.

[22] M. Moradi and M. Keyvanpour, "CAPTCHA and its alternatives: A review," *Security and Communication Networks*, vol. 8, no. 13, pp. 2135– 2156, 2015.

[23] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am robot: (Deep) learning to break semantic image CAPTCHAs," in *Proc. IEEE Eur. Symp. Security and Privacy (EuroS&P)*, Saarbrücken, Germany, Mar. 21–24, 2016, pp. 388–403.

[24] V. D. Nguyen, Y. W. Chow, and W. Susilo, "Attacking animated CAPTCHAs via character extraction," in *Proc. Int. Conf. Cryptology and Network Security (CANS)*, Paraty, Brazil, Nov. 20–22, 2012, pp. 98–113.

[25] Jyotsna, S. Chauhan, E. Sharma, and A. Doegar, "Binarization techniques for degraded document images—A review," in *Proc. Int. Conf. Reliability, Infocom Technologies and Optimization (ICRITO)*, Amity Univ., Noida, India, Sept. 7–9, 2016, pp. 374–379.

[26] S. Y. Huang, Y. K. Lee, G. Bell, and Z. Ou, "An efficient segmentation algorithm for CAPTCHAs with line cluttering and character warping," *Pattern Recognit. Lett.*, vol. 31, no. 11, pp. 1274–1283, Aug. 2009.

[27] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan, "The robustness of hollow CAPTCHAs," in *Proc. ACM Conf. Comput. Commun. Secur.*, Berlin, Germany, Nov. 4–8, 2013, pp. 107–118.

[28] Cloudways, "10 essential eCommerce security tips," [Online]. Available: https://www.cloudways.com/blog/ecommerce-security-tips/. [Accessed: Sept. 1, 2025].

[29] OWASP, "Automated threats to web applications project," [Online]. Available: https://owasp.org/www-project-automated-threats-to-web- applications/. [Accessed: Sept. 1, 2025].

[30] C. Thomas, P. Fraga-Lamas, and T. M. Fernandez-Carames, *Computer Security Threats*. London, U.K.: IntechOpen, 2020. ISBN 978-1-83880- 240-0.

[31] Barracuda Networks, "Bot attacks report vol. 1," [Online]. Available: https://assets.barracuda.com/assets/docs/dms/Bot_Attacks_report_vol1_EN. pdf. [Accessed: Sept. 1, 2025].

[32] J. Chen, X. Luo, Y. Liu, J. Wang, and Y. Ma, "Selective learning confusion class for text-based CAPTCHA recognition," *IEEE Access*, vol. 7, pp. 22246–22259, 2019.

[33] Z. Wang and P. Shi, "CAPTCHA recognition method based on CNN with focal loss," *Security and Communication Networks*, vol. 2021, Art. ID 6641329, Jan. 2021.

[34] C. Li, X. Chen, H. Wang, Y. Zhang, and P. Wang, "An end-to-end attack on text-based CAPTCHAs based on cycle-consistent generative adversarial network," *arXiv preprint arXiv:2008.09102*, Aug. 2020.

[35] S.Senthil Kumar, "Deep Learning-Driven Gesture and Speech Recognition for Human-Machine Interaction: Enhancing Virtual Reality through Ai Integration, "*Karpagam Journal of Computer Science"*, Sep. 2025.