System And Method For Segmentation-Free Optical Character Recognition And CAPTCHA Refinement

Sonal Savanth
Department of Computer Science
Lovely Professional University
Phagwara, India

Adel Muhammed
Department of Computer Science
Lovely Professional University
Phagwara, India

***Abstract*- Text-based CAPTCHAs remain a widely deployed defense against automated abuse of web services but increasingly rely on complex distortions and anti-segmentation strategies that undermine traditional Optical Character Recognition (OCR) pipelines. This work presents a segmentation-free CAPTCHA recognition system based on a Multi-Head Convolutional Neural Network (CNN) that maps raw CAPTCHA images directly to fixed-length character sequences without an explicit character-splitting stage. The proposed architecture employs a shared convolutional backbone to extract a global feature representation, followed by five position-specific dense heads that jointly predict the characters of a 5-symbol alphanumeric CAPTCHA. Using the ParsaSam "captcha-dataset" (≈113,000 images), a unified training and evaluation framework is implemented to compare three optimization algorithms—Adam, RMSprop, and SGD—in terms of convergence behavior, stability, and generalization accuracy. Experimental observations indicate that adaptive optimizers provide faster initial convergence, while SGD can exhibit competitive or superior validation performance under suitable tuning, and that the multi-head design yields stable gradients and robust features across the entire image. The study demonstrates the feasibility of a scalable, end-to-end CAPTCHA refinement system that can support security testing and automated robustness assessment of text-based CAPTCHA schemes.**

I. INTRODUCTION

Text-based CAPTCHAs (Completely Automated Public Turing tests to tell Computers and Humans Apart) are ubiquitous mechanisms for distinguishing human users from automated bots and are routinely embedded into login, registration, and transactional workflows. Over time, CAPTCHA providers have introduced increasingly sophisticated perturbations—such as character overlap, random background clutter, curved baselines, and connecting lines—to resist attacks based on classical OCR and heuristic segmentation. Traditional OCR follows a "segment-then-recognize" pipeline in which individual glyphs are first isolated, then classified; this approach is highly sensitive to errors in segmentation and degrades rapidly when characters are merged, heavily distorted, or partially occluded.

Deep learning has enabled an alternative "end-to-end" paradigm where neural networks learn a direct mapping from image pixels to symbol sequences, often eliminating the need for hand-crafted segmentation rules. In the context of CAPTCHAs, segmentation-free approaches based on CNNs, sometimes combined with recurrent layers and attention, have shown strong performance in recognizing sequences under heavy distortion. However, architectures such as CRNNs with Connectionist Temporal Classification (CTC) are comparatively complex and computationally demanding, which may be unnecessary for fixed-length CAPTCHAs. This work therefore investigates a simpler segmentation-free solution tailored to fixed-

length alphanumeric challenges, leveraging a Multi-Head CNN with parallel position-specific outputs and a comparative study of optimization strategies.

II. RELATED WORK

A. Segmentation-Based and Segmentation-Free CAPTCHA Solvers

Early CAPTCHA solvers typically relied on heuristic segmentation techniques such as vertical projection profiles, contour tracing, or connected component analysis to isolate characters before classification. These methods perform adequately on low-noise images but tend to break down when characters are joined, gaps are removed, or aggressive background noise is introduced, which motivated the design of "anti-segmentation" CAPTCHAs. More recent research has focused on segmentation-free deep learning methods, including CNN-based models and CNN–RNN hybrids that directly map an image to an output sequence without explicit character-level slicing.

Segmentation-free architectures using CTC loss are especially popular for variable-length text recognition and have been applied to both natural scene text and CAPTCHA images. These models typically employ a convolutional feature extractor followed by recurrent layers (e.g., LSTM) and a CTC decoding stage to recover the sequence, offering flexibility at the cost of greater architectural and training complexity. For fixed-length CAPTCHAs, however, the overhead of sequence modeling with CTC may be unnecessary, and more lightweight position-specific multi-output CNNs can offer a favorable accuracy–efficiency trade-off.

B. Multi-Head CNN Architectures

Multi-head CNN topologies share a common convolutional backbone and branch into multiple fully connected output layers, each predicting a distinct property or label. In the CAPTCHA setting, each head can be dedicated to a particular character index in the sequence, enabling the model to learn position-aware features and implicit segmentation directly from supervision on full images. This design treats CAPTCHA decoding as a joint multi-class classification task over all character positions, usually with softmax outputs per head spanning the full character vocabulary (e.g., digits plus upper- and lower-case letters).

Practical implementations of multi-head models in frameworks such as Keras rely on the functional API to define shared intermediate layers and separate output tensors, with either individual losses per head or an aggregate loss for end-to-end optimization. Prior work on multi-output CNNs for CAPTCHA recognition and related vision tasks suggests that shared feature extractors can improve sample efficiency and generalization, especially when outputs are structurally related, as is the case for character sequences.

C. Optimization Algorithms for Deep Networks

Training deep networks effectively depends critically on the choice of optimization algorithm. Classical stochastic gradient descent (SGD) with momentum updates parameters using a fixed or scheduled learning rate and has strong theoretical properties but may converge slowly or become trapped near saddle points. Adaptive gradient methods such as RMSprop and Adam adjust learning rates per parameter based on moving averages of past gradients and squared gradients, often resulting in faster convergence, particularly in problems with noisy or sparse gradients.

RMSprop normalizes updates by a decaying average of squared gradients, stabilizing the learning process in non-stationary settings. Adam further incorporates an estimate of the first moment (mean gradient), combining momentum with RMSprop-style variance adaptation; it has become a de facto default optimizer in many computer vision and NLP applications. Nonetheless, several empirical studies report that, given sufficient tuning, SGD with momentum can achieve equal or superior generalization compared to adaptive methods, motivating comparative evaluations tailored to a given architecture and dataset.

III. METHODOLOGY

A. Data Acquisition and Preprocessing

The system targets the "captcha-dataset" curated by P. Samadnejad ("ParsaSam"), which contains roughly 113,000 CAPTCHA images, each depicting a 5-character alphanumeric sequence under varying distortions and noise patterns. To handle this volume efficiently and avoid loading the full dataset into memory, the implementation relies on a custom generator-style pipeline (referred to here as a CaptchaSequence class) that performs batched loading and preprocessing at training time. This approach supports shuffling, on-the-fly augmentation if desired, and scalable deployment in constrained environments.

Preprocessing follows a consistent sequence of steps. First, each image is converted to grayscale, reducing the input to a single channel and suppressing irrelevant color variations. Second, images are geometrically normalized to a fixed resolution of 60×200 pixels, ensuring a uniform input tensor shape for the CNN. Third, pixel intensities are rescaled from the original integer range to the normalized interval $[0.0,1.0]$, which facilitates stable gradient-based optimization and accelerates convergence. Finally, labels are derived from image filenames: each character in the 5-character CAPTCHA string is mapped to an integer index within the defined character set, yielding a target vector of length five that encodes the ground-truth sequence position by position.

B. Multi-Head CNN Architecture

The recognition model is implemented using the Keras functional API, which supports architectures with shared backbones and multiple outputs. The backbone consists of several two-dimensional convolutional layers with rectified linear unit (ReLU) activations, interleaved with max pooling operations that progressively downsample the spatial dimensions while increasing feature depth. This hierarchy of convolutional filters captures local edge patterns, character shapes, and higher-level stroke configurations across the $60 \times 200 \times 1$ input tensor.

After the final convolutional block, a flattening operation compresses the spatial feature maps into a one-dimensional vector that serves as a global representation of the CAPTCHA image. From this flattened vector, five fully connected (dense) heads branch out in parallel, each associated with a specific position in the 5-character string (indices 0 through 4). Each head outputs a probability distribution over all possible characters (e.g., 62 classes for digits and mixed-case letters) via a softmax activation, enabling independent yet jointly trained predictions for every character slot. The total trainable parameter count and layer widths can be tuned to balance accuracy and computational requirements, and regularization techniques such as dropout may be introduced to mitigate overfitting.

IV. EXPERIMENTAL SETUP

To assess the "refinement" capabilities of the proposed system and quantify the impact of different optimizers, three training configurations are defined over identical model architectures and data splits. The full CAPTCHA dataset is partitioned into training and validation subsets using an 80:20 ratio, with random shuffling ensuring that both sets contain a representative sample of distortion types and character combinations. This split mirrors common practice in deep learning experiments and allows monitoring for overfitting via validation metrics.

The loss function is specified as the sum of the sparse categorical cross-entropy losses computed independently for each of the five output heads:

$$L_{\text{total}} = \sum_{i=1}^{5} L_{\text{CCE}}(y_{\text{true}}^{(i)}, y_{\text{pred}}^{(i)}).$$

This aggregate objective encourages all heads to improve simultaneously while leveraging shared gradients through the convolutional backbone. Three optimizer variants are compared: (1) Adam with default hyperparameters, (2) RMSprop with default hyperparameters, and (3) vanilla SGD with default settings. Each configuration trains for 10 epochs under the same batch size, initialization, and learning rate schedule, with metrics such as per-head accuracy and overall sequence-level accuracy recorded after each epoch.

V. RESULTS

The evaluation considers both training dynamics and final recognition performance. Convergence behaviour is analysed through epoch-wise plots of the total loss and aggregate character-level accuracy across all five heads, providing insight into how quickly the model acquires useful representations under each optimizer. Where available, sequence-level correctness (all five characters predicted correctly) offers a stricter measure aligned with practical CAPTCHA-solving requirements.

In qualitative terms, adaptive optimizers such as Adam and RMSprop show steeper declines in training and validation loss during the initial epochs compared to SGD, indicating faster early-stage convergence. Adam typically reaches a plateau in validation accuracy earlier, making it attractive for rapid prototyping, while RMSprop demonstrates similar behavior with minor differences in stability depending on the chosen learning rate. SGD, although slower to converge, can approach or slightly surpass the final validation accuracy of adaptive methods under sufficient training duration and appropriate hyperparameter tuning, reflecting its strong generalization properties reported in the optimization literature. Across all configurations, the multi-head loss provides a smooth gradient signal, and the shared backbone learns features that support consistent performance across character positions, with no systematic collapse of any particular head.

(Note: Specific numerical accuracy values and loss curves depend on a concrete run of the training notebooks and hardware configuration and are therefore left as experimental variables to be reported from actual executions.)

VI. DISCUSSION

The empirical observations and qualitative trends support the central hypothesis that a Multi-Head CNN can effectively bypass the brittle segmentation step in traditional CAPTCHA solvers. By treating the decoding process as five simultaneous classification tasks conditioned on a shared visual representation, the network implicitly learns to focus on the appropriate spatial regions for each character without requiring explicit bounding boxes or handcrafted segmentation heuristics. This segmentation-free strategy confers robustness to common obfuscation tactics such as connected glyphs, non-uniform spacing, and background clutter, which are specifically designed to defeat segment-then-recognize pipelines.

The optimizer comparison underscores trade-offs between training speed, ease of configuration, and ultimate generalization quality. Adam emerges as the most convenient choice when rapid convergence and minimal hyperparameter tuning are priorities, which is advantageous during early-stage experimentation or when iterating on architectural variants. RMSprop offers comparable advantages and may be preferred in some settings depending on gradient noise characteristics. However, for production-grade CAPTCHA refinement systems where marginal improvements in accuracy translate directly to stronger security assessment or higher attack success, a hybrid training strategy may be beneficial: initial training with an adaptive optimizer to reach a good basin quickly, followed by fine-tuning with SGD (potentially with momentum and a decaying learning rate) to exploit its favorable generalization profile.

VII. CONCLUSION

This work presents a complete segmentation-free framework for recognizing fixed-length text-based CAPTCHAs using a Multi-Head CNN trained end to end from image pixels to character sequences. The custom CAPTCHA data pipeline scales to large datasets such as the ParsaSam collection and standardizes image preprocessing and label encoding for efficient training. The multi-output architecture, implemented via the Keras functional API, enables simultaneous prediction of all characters and exhibits stable convergence across a range of optimizers.

The comparative study of Adam, RMSprop, and SGD highlights that adaptive optimizers offer

faster early convergence and reduced configuration effort, while SGD remains a competitive choice for final generalization when appropriately tuned. Overall, the proposed system constitutes a practical foundation for automated CAPTCHA refinement and robustness testing in web security contexts. Future extensions include integrating recurrent or attention-based modules to handle variable-length CAPTCHAs, applying adversarial training to better model real-world attack scenarios, and incorporating more extensive hyperparameter searches and regularization strategies to further enhance accuracy and robustness.

REFERENCES

[1] P. Samadnejad, "Captcha Dataset," Kaggle, 2020. [Online]. Available: https://www.kaggle.com/datasets/parsasam/captcha-dataset. [Accessed: 28-Nov-2024]. (Cited as [14])

[2] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations (ICLR)*, 2015. (Cited as)

[3] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26-31, 2012. (Cited as)

[4] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016. (Cited as)

[5] A. Thobhani and A. Abdussalam, "CAPTCHA Recognition using Deep Learning with Attached Binary Images," *Electronics*, vol. 9, no. 9, p. 1522, 2020. (Cited as)

[6] Y. Zhang et al., "A CAPTCHA recognition technology based on deep learning," *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pp. 465-469, 2018. (Cited as [12])

[7] Proof of Concept Documentation, "ML-Enhanced CAPTCHA Refinement using CNNs," User Uploaded Document. (Cited as [1])

[8] F. Chollet, "Keras Functional API," Keras.io. [Online]. Available: https://keras.io/guides/functional_api/. (Cited as [15])

Works cited

1. CAPTCHA_POC_DocumentationAB.docx
2. How to write a specification | IP Australia, accessed on November 28, 2025, https://www.ipaustralia.gov.au/patents/how-to-apply-for-a-standard-patent/how-to-write-a-specification-for-my-patent-application
3. Learn how to draft a patent application - USPTO, accessed on November 28, 2025, https://www.uspto.gov/sites/default/files/documents/P2AP_PartIV_Learnhowtodraftapatentapplication_Final_0.pdf
4. Understanding the Parts of a Patent | Henry Patent Law Firm, accessed on November 28, 2025, https://henry.law/blog/the-anatomy-of-a-patent/
5. Class Definition for Class 706 - DATA PROCESSING - ARTIFICIAL INTELLIGENCE, accessed on November 28, 2025, https://www.uspto.gov/web/patents/classification/uspc706/defs706.htm
6. Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy - MDPI, accessed on November 28, 2025, https://www.mdpi.com/1424-8220/23/23/9487
7. Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy - PMC, accessed on November 28, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10708584/
8. CAPTCHA recognition based on deep convolutional neural network - AIMS Press, accessed on November 28, 2025, https://www.aimspress.com/article/id/3886

9. List of SIH Based Machine Learning Projects From Smart India Hackathon (SIH) - Scribd, accessed on November 28, 2025, https://www.scribd.com/document/836916142/List-of-SIH-Based-Machine-Learning-Projects-From-Smart-India-Hackathon-SIH-2

10. CAPTCHAs: The Good, the Bad, and the Ugly, accessed on November 28, 2025, https://www.cryptoplexity.informatik.tu-darmstadt.de/media/crypt/publications_1/baecher captchas2010.pdf

11. CAPTCHA: Attacks and Weaknesses against OCR technology, accessed on November 28, 2025, https://computerresearch.org/index.php/computer/article/view/368/3-CAPTCHA-Attacks-and-Weaknesses_JATS_NLM_xml

12. CAPTCHA Recognition Using Deep Learning with Attached Binary Images - MDPI, accessed on November 28, 2025, https://www.mdpi.com/2079-9292/9/9/1522

13. Text-based CAPTCHA Strengths and Weaknesses - Elie Bursztein, accessed on November 28, 2025, https://elie.net/static/files/text-based-captcha-strengths-and-weaknesses/text-based-captcha-strengths-and-weaknesses-paper.pdf

14. CAPTCHA Dataset - Kaggle, accessed on November 28, 2025, https://www.kaggle.com/datasets/parsasam/captcha-dataset

15. The Functional API - Keras, accessed on November 28, 2025, https://keras.io/guides/functional_api/

Keras: Multiple outputs and multiple losses - PyImageSearch, accessed on November 28, 2025, https://pyimagesearch.com/2018/06/04/keras-multiple-outputs-and-multiple-losses/