RESEARCH ARTICLE

Denoising Telerik RadCaptcha: A Comparative Evaluation of the Effectiveness of Pre-Processing Techniques and Deep Learning Method Using a Novel Dataset

Talha Bin Omar[1,†], **Tahir Sher**[2,†], **Abdul Rehman**[3,*] and **M. Haroon Khan**[1]

[1] Department of Creative Technologies, Air University, Islamabad 44000, Pakistan
[2] Department of Artificial Intelligence, Korea University, Seoul 02842, Republic of Korea
[3] Human Data Convergence Institute, Jeonju University, Jeonju 55069, Republic of Korea

Abstract

Text-based CAPTCHAs remain a widely deployed mechanism to distinguish humans from automated bots. The Telerik RadCaptcha, a component of the ASP.NET AJAX suite, generates distorted alphanumeric images with character overlap, intersecting lines, and dynamic background noise. This study introduces a novel, real-world dataset of 3,000 labeled Telerik RadCaptcha images and proposes a specialized multi-stage preprocessing pipeline featuring adaptive binarization and contour-based segmentation to robustly isolate overlapping and noisy characters—challenges where conventional methods frequently fail. The segmented characters are then classified using a lightweight Convolutional Neural Network (CNN). Experimental results demonstrate 99.26% training accuracy, 97.60% character-level test accuracy, and 92.08% full-sequence accuracy on unseen 5-character CAPTCHAs, with stable learning curves indicating effective generalization and minimal overfitting. These findings reveal critical vulnerabilities in traditional text-based CAPTCHA designs and provide empirical insights to guide the development of more resilient verification mechanisms.

Keywords: Convolutional neural network, deep learning, Telerik RadCaptcha.

1 Introduction

CAPTCHA (Completely Automated Public Turing test to tell Humans and Computers Apart), a term coined by Von *et al.* [1] is a challenge-response-based authentication mechanism test that most humans can pass, but current computer programs cannot pass [2]. The concept of CAPTCHA was inspired by the pioneering work of Alan Turing, who in 1950

Submitted: 18 May 2025
Accepted: 10 September 2025
Published: 10 February 2026

Vol. 2, **No.** 2, 2026.
10.62762/TACS.2025.469136

***Corresponding author:**
✉ Abdul Rehman
a.rehman.jj@jj.ac.kr

† These authors contributed equally to this work

Citation
Omar, T. B., Sher, T., Rehman, A., & Khan, M. H. (2026). Denoising Telerik RadCaptcha: A Comparative Evaluation of the Effectiveness of Pre-Processing Techniques and Deep Learning Method Using a Novel Dataset. *ICCK Transactions on Advanced Computing and Systems*, 2(2), 85–106.

proposed the idea of the "Imitation Game" as a means to assess a machine's ability to exhibit intelligent behavior indistinguishable from that of a human [3]. This foundational idea has since been built upon and refined through various iterations, resulting in the modern CAPTCHA system with diverse types and forms.

CAPTCHAs are typically employed in web applications to prevent automated scripts and bots from accessing or exploiting sensitive resources [4]. There exist several types of CAPTCHAs, including but not limited to: text-based CAPTCHAs that require users to type a distorted sequence of characters [5], as shown in Figure 1, image-based CAPTCHAs that ask users to identify specific objects within an image [6], as shown in Figure 2, and audio-based CAPTCHAs that demand users to listen to an audio clip and respond accordingly Figure 3 [7]. Furthermore, some CAPTCHA systems incorporate machine learning techniques to adapt to evolving threats and improve their security posture [8].

Figure 1. Text-based CAPTCHA systems employ distorted characters or alphanumeric sequences designed to challenge and verify human users, ensuring robust defense against automated bots. These systems often include noise, overlapping letters, or variable font styles to enhance security [9, 10].

To enhance robustness against automated attacks, CAPTCHA systems frequently incorporate techniques such as character distortion, rotation, addition of background noise, or complex image overlays to challenge machine-based recognition systems. These measures aim to prevent malicious actors from using machine learning algorithms to bypass authentication mechanisms. However, with the advent of advanced deep learning frameworks, malicious actors have been able to bypass these authentication mechanisms [15, 16]. Research has shown that deep learning techniques-based solvers can effectively break even the most sophisticated CAPTCHAs created by leading providers [17–19]. In particular, convolutional neural networks (CNN) along with recurrent neural networks (RNN) show high efficiency in recognizing distorted

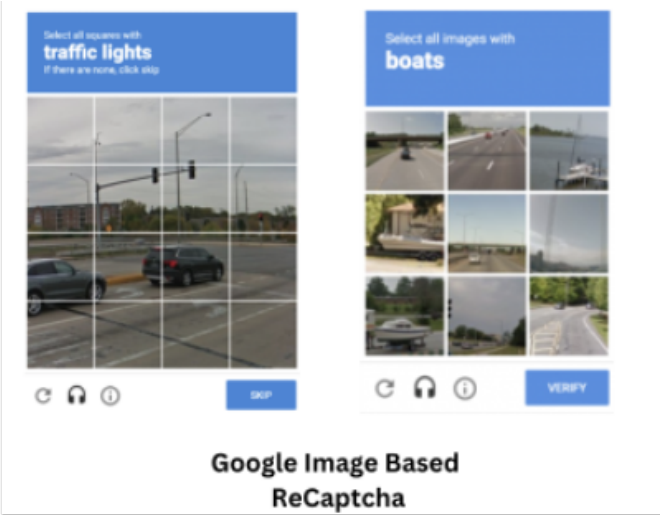

Figure 2. Google Image-Based reCAPTCHA: A security system that verifies users by asking them to identify objects in images, leveraging human cognition and machine learning to block bots [11, 12].

Figure 3. Google Audio and Telerik AJAX Audio CAPTCHA: Audio CAPTCHA systems that verify users by requiring them to interpret and input spoken characters or numbers, ensuring accessibility and security against bots [13, 14].

characters and decoding the intended content [20–22].

Figure 4. Examples of Telerik RadCaptcha from dataset curated in this study.

Given this context, our research seeks to investigate the security posture of Telerik RadCaptcha against such deep learning-based attacks. As a prominent CAPTCHA system, Telerik RadCaptcha employs a graphical interface, as shown in Figure 4 that presents users with a noisy image containing 5 alphanumeric characters, which they must correctly identify and type into the corresponding field [23]. Our study aims to

examine the effectiveness of this system in preventing automated attacks and analyze its design, functionality, and vulnerabilities.

Despite the numerous studies on CAPTCHA systems, Telerik RadCaptcha remains understudied. Characterized by intricacies that demand specialized attention, it presents a unique set of challenges that necessitate dedicated research attention. The dataset exhibits distinct characteristics, including intertwined patterns overlaid with lines that obscure and overlap characters; non-linear character alignment; and dynamic noise distributions, comprising speckles and irregular shading. These features confound standard segmentation algorithms. Traditional preprocessing techniques [24–27], such as median filtering, employed in previous research, are ineffective in separating characters from the intertwined patterns and noise lines present in this CAPTCHA. Consequently, existing methodologies fall short in addressing the intricacies of Telerik RadCaptcha. Furthermore, the absence of pre-existing datasets for this CAPTCHA type demands the creation of a bespoke dataset and advanced models specifically designed to handle unique characteristics. In response, this study introduces two primary contributions: a novel, labeled dataset curated from live Telerik RadCaptcha instances, and a specialized preprocessing pipeline engineered to overcome the specific challenges where conventional methods fail. This research aims to systematically analyze these challenges and aims to evaluate the effectiveness of current pre-processing and deep learning methods against this complex format. By evaluating these techniques under different conditions, we aim to understand the factors that contribute to their resilience and susceptibility, ultimately guiding the development and generation of more robust countermeasures [28–30].

The subsequent sections of this paper will establish a solid foundation for our research; specifically, Section 2 provides an overview of the state-of-the-art methods used to evade CAPTCHAs using deep learning techniques. We highlight the limitations of existing CAPTCHAs and the potential vulnerabilities that attackers can exploit. In Section 3, we present our methodology for evaluating the security posture of Telerik RadCaptcha, including a detailed description of our experimental configuration and the evaluation criteria used. Our findings are detailed in the following Section 4, where we demonstrate the effectiveness of deep learning-based attacks on Telerik RadCaptcha and highlight its vulnerabilities. We conclude this paper by analyzing the broader implications of our findings. Finally, Section 7 provides a summary of our contributions to the field of computer security and identifies potential areas for future research.

2 Related work

This section provides a review of key studies that have helped shape our understanding of CAPTCHA security, highlighting important findings and insights in the field. Summary of related work is shown in Table 1.

Literature [31] proposed Deep-CAPTCHA, a deep learning-based model that leveraged a CNN, to address the challenges of recognizing distorted alphanumeric characters amidst complex noise and overlapping patterns, overcoming limitations of traditional preprocessing methods. The network structure consisted of multiple layers, including Convolutional-MaxPooling pairs with increasing numbers of neurons (32, 48, and 64), followed by a dense layer with Rectified Linear Unit (ReLU) activation and dropout rate, and finally, L separate Softmax layers for character classification. Notably, the authors achieved impressive performance on CAPTCHA samples, with efficacy rates of 98.94% for numerical and 98.31% for alphanumeric CAPTCHAs.

More recently, Kumar *et al.* [32] addressed the challenge of breaking two-color and multi-color Hindi CAPTCHAs, examining how color variability, noise, and distortion affect the performance of various classification models. The author conscientiously evaluated the performance of the k-nearest neighbors (k-NN) classifier on both color schemes, utilizing a range of features including raw pixel, horizontal projection (HP), scale-invariant feature transform (SIFT), Speeded up Robust Feature (SURF), and Oriented FAST and rotated BRIEF (ORB). Their results indicated segmentation rates ranging from 82% to 98% for two-color CAPTCHAs and 86% to 100% for multi-color CAPTCHAs, with the k-NN classifier exhibiting optimal performance when employing raw pixel features across various parameter settings.

Literature [31] proposed significant advancements in the field, which were further extended by [32]. Building on these efforts, [33] tackled the challenge of optimizing CAPTCHA recognition, specifically focusing on alphanumeric combinations with noise, rotation, and adhesion, by developing a refined SVM (Support Vector Machine)-based approach. The authors' approach involved image preprocessing

through binarization and denoising, followed by segmentation using projection methods and Color Filling Segmentation (CFS).

This is then complemented by classification using an SVM model with a radial basis function (RBF) kernel, achieving an accuracy of 98.5% in recognizing pre-processed CAPTCHAs. In recent research, [34] proposed a CAPTCHA solver framework based on a skip-connection CNN to tackle modern text-based CAPTCHAs. Using two public datasets with 4- and 5-character images, the study employed 5-fold validation, achieving accuracy rates of 98.82% and 85.52%, respectively. The methodology included preprocessing steps like binarization and segmentation to normalize input data, followed by a deep learning model enhanced with skip connections to improve feature extraction and reduce bias in prediction. The novel architecture demonstrated resilience in recognizing circuitous CAPTCHA features, outperforming traditional CNN approaches. Similarly, Derea *et al.* [35] proposed a dual-layer attention-based CAPTCHA recognition approach with guided visual attention that integrates CNN feature extraction and recurrent LSTM networks with adaptive attention mechanisms to more effectively handle noise, interference, and character correlation in complex CAPTCHA images, demonstrating robust performance on varied real-world datasets. These works highlight the vulnerability of current CAPTCHA designs to advanced machine-learning attacks, emphasizing the need for more secure CAPTCHA mechanisms.

In their study, Zhang *et al.* [36] proposed a framework leveraging Generative Adversarial Networks (GANs) for breaking text-based CAPTCHAs, particularly those on dark web platforms with complex background noise and variable character lengths. The framework combines GAN-based background removal, an advanced character segmentation algorithm, and CNN for character recognition. Their approach achieved a success rate of over 92.08% on benchmark and dark web CAPTCHA datasets, significantly surpassing existing methods. This research contributes to automated CAPTCHA breaking and supports large-scale monitoring of dark web activities for Cyber Threat Intelligence. Kumar *et al.* [37] employed MSER descriptors with deep learning for CAPTCHA recognition, highlighting effectiveness in noisy environments.

Derea *et al.* [38] introduced a novel CAPTCHA recognition method called CNN-based Recognition with Grouping Strategy (CRNGS), which bypasses the computationally expensive segmentation step. This method generates multiple binary images, each associated with a specific group of characters within the CAPTCHA. A dedicated softmax layer is then employed for character classification within each group. The CRNGS model was evaluated on four different CAPTCHA schemes: Bank of China, Weibo, Captha 0.3, and Gregwar. The reported accuracies reached up to 99.87% for Bank of China, 99.37% for Weibo, 98.76% for Captha 0.3, and 99.78% for Gregwar. Compared to existing methods utilizing Recurrent Neural Networks (RNNs) and Multilabel classification, CRNGS demonstrated superior performance in terms of accuracy, parameter size, and storage requirements, making it a more efficient and effective solution for CAPTCHA recognition. The authors highlight the flexibility of the CRNGS architecture, allowing for adjustments to the number of softmax layers based on the characteristics of the specific CAPTCHA scheme being targeted.

Wan *et al.* [39] proposed Adaptive CAPTCHA, a novel CRNN (Convolutional Recurrent Neural Network)-based text CAPTCHA solver addressing the challenges of noise and computational efficiency. Their key contribution is the Adaptive Fusion Filter Network (AFFN), which dynamically adjusts filter weights based on the estimated noise level of the CAPTCHA image, enabling more effective feature extraction. Unlike traditional denoising techniques, the AFFN learns an optimal filtering strategy through a fusion factor, allowing it to generalize across datasets with varying noise characteristics. Integrated within a CRNN architecture incorporating LSTM layers and residual connections, the AFFN achieved over 99% accuracy on both the M-CAPTCHA and P-CAPTCHA datasets. Furthermore, the Adaptive CAPTCHA significantly reduces the parameter count (by 39% and 70% compared to a baseline CRNN on the respective datasets), suggesting a more efficient architecture. However, further investigation is needed to assess the model's performance on more complex CAPTCHA styles and explore potential limitations of the adaptive filtering approach.

Dankwa *et al.* [40] developed a novel Depth-Wise Separable Convolutional Neural Network (DWSCNN) aimed at assessing vulnerabilities in text-based CAPTCHAs. Their work addressed the challenges of CAPTCHA-breaking by leveraging a single-character extraction (SCE) algorithm, which splits CAPTCHA

Table 1. Summary of related works.

Ref	Year	Methodology	Language/Scheme	Accuracy
[32]	2023	Character segmentation using vertical and horizontal projections; morphological operations.	Hindi (Two-colour & Multi-colour, various designs)	k-NN (up to 90.5% two-colour, up to 89.6% multi-colour), SVM (up to 91.4% both), Random Forest (up to 94.5% both)
[33]	2019	SVM (with RBF kernel), Preprocessing (Binarization, Denoising, Segmentation-Projection and Color Filling).	Alphanumeric, Distorted/Rotated/Adhered, Chinese Characters, Custom Dataset (10,000 images per CAPTCHA type)	Up to 98.81% (SVM), Up to 98.43% (VGG16)
[42]	2019	Reconstructs CAPTCHA generator, generates training data, then uses peak segmentation, object detection (TOD), and CNN for character recognition.	Two self-generated datasets based on open-source libraries	More than 75%
[31]	2020	Deep CNN with 5 convolutional layers, 3 max-pooling layers; optimization using Adam/SGD.	Persian Numerical, Synthetic Dataset	98.90%
[36]	2020	DW-GAN (Generative Adversarial Network + Character segmentation + CNN recognition), pre-processing (grayscale conversion, Gaussian smoothing, pixel normalization).	Text-based, Dark Web (Rescator types 1 & 2, Yellow Brick)	92.08% (Rescator 1), 97.50% (Rescator 2), 95.98% (Yellow Brick)
[44]	2020	Unsupervised learning reconstructs the CAPTCHA generation process. Representation learning trains a recognizer on synthesized CAPTCHAs with simulated distortions.	Synthetic CAPTCHA data generated using reconstructed CAPTCHA algorithms; Real-world CAPTCHAs from Bing, eBay, Microsoft, Wikipedia, and Weibo.	Up to 94.5%
[48]	2020	CNN and LSTM for text-based CAPTCHA breaking and solving. Preprocesses CAPTCHAs based on type (Rotated, Noisy Arc, Complicated Background) using techniques like erosion, dilation, and binarization.	Rotated (9955 images), Noisy Arc (1070 images), Complicated Background (1000 images).	85.97% (Rotated), 84.52% (Noisy Arc), 82.91% (Complicated background)
[40]	2021	Depth-wise Separable Convolutional Neural Network (CNN) architecture for CAPTCHA breaking. The model uses separable convolutions, batch normalization, and dropout.	CAPTCHA Python Library Dataset	100%
[43]	2021	Semi-supervised learning approach by combining basic segmentation with transfer learning from a small, labeled synthetic dataset to a larger unlabeled real-world CAPTCHA dataset.	Synthetic CAPTCHA dataset (500 images); Unlabeled real-world CAPTCHA dataset (50,000 images).	Up to 95.8% overall; Numbers (0-9): 94.78% (5 char), Lowercase letters (a-z): 32.89% (5 char)
[47]	2021	k-Nearest Neighbors (k-NN) with bit-based similarity. Framework includes preprocessing (binarization, denoising, segmentation), building a standard library, and image recognition.	MNIST; Self-created CAPTCHA datasets.	96.67% (MNIST), 95.65% (CAPTCHAS), 96.25% (CNKI), 99.61% (CMS)
[34]	2022	Deep CNN with skip connections, 5-fold cross-validation, preprocessing (binarization, erosion, etc.).	4-character (5200 images, 32 classes) and 5-character (5200 images, 19 classes) publicly available	98.92% (4-char), 98.86% (5-char)
[38]	2023	CRNN Grouped Softmax (CNN with grouped Softmax layers for character-group recognition), ABIs Algorithm.	Bank of China (4-char CAPTCHAs with distortion), Weibo (4-char with selective exclusions), Captcha 0.3 (4-char with noise), Gregwar (4-char with noise lines and rotation).	96.39% (BoC), 92.68% (Weibo), 95.33% (Captcha 0.3), 51.23% (Gregwar)
[41]	2023	Three tailored deep learning architectures using CNNs and LSTMs with attention for three types of CAPTCHAs: Type I (Text Based), Type II (instructions + image), and Type III (image instructions + image). Dynamic LSTM decoder for Type I; Multi-encoder-decoder for Types II & III.	Real-world CAPTCHAs from seven Indian government websites; synthetic datasets for data augmentation	Up to 98.50% (Type I)
[45]	2023	CNNs and ensemble methods with different voting schemes, for text-based CAPTCHA recognition. Performs hyperparameter optimization using grid search cross-validation.	Synthetic text-based CAPTCHA dataset (1070 images).	92.28% (Plurality Voting Function), 92.42% (Fuzzy Average Voting Function)
[39]	2024	AFFN to preprocess CAPTCHA images, reducing noise and interference; Convolutional Recurrent Neural Network (CRNN) architecture consisting of CNN, Bi-LSTM, and a connectionist temporal classification (CTC) layer.	M-CAPTCHA, P-CAPTCHA	99% (M-CAPTCHA), 99% (P-CAPTCHA)

images into individual characters for efficient training. The authors generated a dataset of 10,000 CAPTCHA images with overlapping and rotated characters designed to simulate real-world CAPTCHA schemes. Using a SCE algorithm, this dataset was expanded to 40,000 single-character images and divided into training, validation, and testing subsets. Ambiguous characters, such as "O" and "I" were excluded to improve reliability in model training. The proposed DWSCNN architecture incorporates six separable depth-wise convolutional units, seven stages of batch normalization to stabilize the learning process, four dropout layers for robust regularization, three max-pooling operations to capture local spatial hierarchies, and a single fully connected layer for predictive inference, resulting in a model with 191,607 trainable parameters and a compact size of 2.36 MB. The network achieved a testing accuracy of 99.8%, demonstrating superior performance in breaking CAPTCHAs with simple backgrounds.

Bhowmick *et al.* [41] investigated the automated breaking of CAPTCHAs using deep learning, focusing on three common types: image-based text, expression-based text, and combined image-and-text CAPTCHAs. To address the challenges posed by diverse CAPTCHA formats, they developed specialized encoder-decoder architectures for each type, leveraging lightweight convolutional and recurrent neural networks with LSTM and attention mechanisms. Their experimental evaluation, using a dataset of real CAPTCHAs scraped from seven Indian government websites, achieved accuracies between 88% and 98% across the different CAPTCHA types. Notably, these results were achieved with relatively small training datasets, in the order of a few thousand samples, requiring only several hours of training. This demonstrates the effectiveness of their approach even with limited data. The paper's focus on lightweight architectures and minimal training data requirements is particularly relevant to research in deep learning for CAPTCHA breaking, though further investigation into failure cases and comparisons with other state-of-the-art techniques would strengthen the analysis.

Yu *et al.* [42] proposed a low-cost method for breaking Python CAPTCHAs using a plain-text-based attack. By capitalizing on the open-source nature of CAPTCHA libraries, they developed a synthetic CAPTCHA generator that replicated the target system's functionality, thereby overcoming the challenge of limited training data without relying on manual annotation. Their method integrated a peak segmentation algorithm for character localization with a CNN for character recognition. The CNN architecture consisted of two convolutional layers with filter sizes of 3×3, each followed by ReLU activation and max-pooling layers. These were followed by a dropout layer to mitigate overfitting, two fully connected layers with 128 and 64 hidden units, respectively, and an output layer with 42 classes activated by a softmax function. Evaluated on two open-source Python CAPTCHA modules (Claptcha and Captcha), their approach achieved character-level accuracies exceeding 90%. Additionally, they tested their framework on the HashKiller CAPTCHA dataset, composed of numeric CAPTCHAs, and achieved a 73% accuracy. While this accuracy is moderate compared to modern deep learning models trained on larger and more diverse datasets, their low-cost approach, relying on synthetic data generation and minimal computational resources, is particularly relevant for resource-constrained settings.

Bostik *et al.* [43] introduced a semi-supervised method for circumventing text-based CAPTCHA systems. In addressing the challenges of text-based CAPTCHAs, including limited labeled data, design variations, and computational cost, they leveraged transfer learning to employ a convolutional neural network initially trained on a limited set of labeled CAPTCHA images. The model's performance is then bolstered by fine-tuning it on a significantly larger, unlabeled dataset. Experiments demonstrate the efficacy of this method, achieving a classification accuracy of up to 98.9% on 5-character CAPTCHAs. This result represents a significant advancement, as their semi-supervised strategy requires substantially fewer labeled samples compared to previous fully supervised approaches, which often necessitated thousands of manually annotated CAPTCHAs to achieve comparable performance levels. Further investigation revealed variations in the reported accuracies depending on CAPTCHA length and character type (numeric vs. alphanumeric), with lower performance observed for longer CAPTCHAs.

Tian *et al.* [44] introduced a novel framework to address the challenges of achieving high accuracy with limited labeled training data and overcoming the complexities introduced by diverse CAPTCHA styles and distortions. Their approach combines unsupervised representation learning with a self-supervised recognition model which is centered on a GAN-based CAPTCHA decomposer trained to

separate foreground characters from background noise and distortion without manual annotation. This decomposer generates a character layer image and a corresponding weight mask, effectively isolating individual characters. A self-supervised recognizer, trained on synthetically augmented data derived from a small set of labeled real CAPTCHAs, then processes these segmented characters. Evaluated on a range of CAPTCHA schemes, their method demonstrated high effectiveness. With only 500 labeled samples, they achieved accuracies of 93.07% and 91.25% on Bing and Weibo CAPTCHAs, respectively. The unsupervised decomposer itself demonstrated notable denoising capabilities, removing up to 75% of background interference on certain CAPTCHA schemes. However, the authors acknowledge limitations with handling severely distorted or overlapping characters. Specifically, performance on Google's reCAPTCHA v2 was limited, highlighting challenges posed by more advanced CAPTCHA designs.

Kovács *et al.* [45] introduce a novel approach by combining a Convolutional Neural Network (CNN) architecture with ensemble learning techniques. They aimed to overcome the challenges posed by CAPTCHA distortions and variations through systematic hyperparameter tuning and a rigorous evaluation framework. Their methodology centers on a CNN featuring three convolutional layers coupled with max-pooling for hierarchical feature extraction. Which is then followed by a dense classification layer with branched outputs for individual CAPTCHA character prediction. A key contribution lies in their systematic hyperparameter optimization via grid search cross-validation, encompassing batch size, dropout rate, penultimate dense layer neuron count, and activation function selection. Furthermore, the authors explore the efficacy of ensemble methods, comparing plurality voting and fuzzy average voting schemes for aggregating predictions from multiple CNN instances. Experimental evaluations demonstrate the superiority of the fuzzy average voting ensemble, yielding accuracies exceeding 92% on the test set. This work highlights the synergistic potential of combining optimized CNN architectures with ensemble strategies for achieving robust and accurate CAPTCHA decryption. Relatedly, Derea *et al.* [46] demonstrated a novel CAPTCHA recognition system based on refined visual attention, which substantially improves classification robustness in noisy environments with overlapping and distorted characters, aligning well with CAPTCHA decryption under uncertain conditions.

Lin *et al.* [47] developed a configurable image recognition framework targeting text-based CAPTCHAs and handwritten digits, employing k-NN classification and a novel bit-based similarity metric. Their research sought to overcome the limitations of rigid, CAPTCHA-specific solutions by developing a configurable framework based on k-NN and bit-based similarity measures, thereby allowing for adjustable parameters. The framework involved preprocessing steps such as binarization, denoising, and segmentation, followed by the construction of a standard library of known characters. The core of their approach is a bit-based similarity model, which computes the ratio of bitwise AND and OR operations between the binary representations of the input image and the standard library images. Experimental evaluation on MNIST, synthetic CAPTCHAs, CNKI website CAPTCHAs, and PHP DedeCMS CAPTCHAs yielded an average accuracy of 97.05%, showcasing the framework's effectiveness and adaptability across diverse datasets. The authors highlight the framework's potential for broader application through customizable preprocessing and the integration of additional recognition algorithms.

UmaMaheswari *et al.* [48] investigated the breakability of text-based CAPTCHAs employing various defense mechanisms, such as rotations, noise arcs, and complex backgrounds by proposing a deep learning-based approach utilizing a blend of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. Their approach preprocesses CAPTCHAs based on their type (Rotated, Noisy Arc, Complicated Background) using techniques like erosion, dilation, and binarization. The CNN extracts features which are then fed to the LSTM to predict the character sequence. They evaluated their model on datasets of 9,955 Rotated, 1,070 Noisy Arc, and 1,000 Complicated Background CAPTCHAs, achieving accuracies of 85.97%, 84.52%, and 82.91% respectively. Our work complements recent advancements in intelligent system design and user-centered evaluation frameworks, such as expertise-based recommendation systems in global software development contexts [49] and UX enhancement models in open-source environments [50], by addressing robustness and accuracy in human-computer interaction through CAPTCHA denoising.

Existing CAPTCHA-breaking methodologies, often

relying on CNNs, SVMs, or k-NN classifiers coupled with binarization and segmentation and utilizing synthetic captcha datasets which frequently falter when applied to real-world CAPTCHAs. This research is motivated by the significant limitations of existing approaches when applied to real-world CAPTCHAs, which, as evidenced by our novel dataset, exhibit substantial noise, character distortions, overlapping characters and which often prove insurmountable for existing techniques. Furthermore, the scarcity of labeled real-world data necessitates the exploration of alternative approaches. This work addresses these limitations by focusing on a novel, real-world dataset and developing a more sophisticated CAPTCHA-breaking methodology. Our objective is to enhance the practical feasibility of CAPTCHA decryption in real-world deployments, explicitly addressing the diversity and complexity encountered in practical applications.

3 Methodology

3.1 Dataset Collection and Overview

This study utilizes a novel dataset containing 3000 images for training and an additional 580 images for testing which were obtained by scraping data from the official website of the [51]. The source for scraping CAPTCHA images is available at: https://ugadmissions.nust.edu.pk/result/meritsearch.aspx. The scraping process involved using an automated script written in Python utilizing libraries such as BeautifulSoup and requests to download the images. This dataset comprises CAPTCHA images generated by Telerik RadCaptcha, featuring a consistent width (w) of 180 pixels and height (h) of 50 pixels. Each image contains 5 alphanumeric characters, accompanied by visual noise elements such as lines and background distortions. Following the collection phase, each Telerik RadCaptcha image was manually annotated by the authors. This annotation process involved overcoming several challenges inherent to the RadCaptcha's design. A primary difficulty was handling visual noise, which included random lines and background distortions that often cut through or obscured parts of the characters. Careful inspection was required to distinguish character strokes from this noise. Another significant challenge was the presence of overlapping and touching characters. The characters were often warped and positioned in close proximity, making it difficult to delineate individual glyphs and determine their correct order. Finally, ambiguity between similar-looking glyphs (e.g., 'O' and '0', '1' and 'I') required establishing a consistent labeling protocol. The annotator carefully examined these ambiguous cases, sometimes referencing clearer instances from other images, to ensure high-quality and reliable ground-truth labels for the dataset. Given the novelty of this dataset, it represents a significant contribution to Text-Based CAPTCHA research. Based on existing available information, there is no prior publicly available dataset of Telerik RadCaptcha. To promote further research and innovation in the field, the dataset has been made publicly available on Kaggle [52] to enable researchers to replicate experiments and build upon the foundation established in this work.

3.2 Preprocessing

Our proposed methodology is founded on the efficacy of character segmentation in enabling accurate image-to-text analysis. Specifically, we will leverage this technique to facilitate the accurate analysis, segmentation, and prediction of text from Telerik RadCaptcha images.

- The dataset images are initially converted to grayscale format using OpenCV's read function, thereby eliminating color channel information and retaining only the luminance values of each pixel, which significantly diminishes computational complexity and facilitates subsequent processing. According to the ITU-R BT.601 standard [53], the transformation from RGB to grayscale is computed as:

$$Y = 0.299R + 0.587G + 0.114B \tag{1}$$

- The adaptive thresholding method is then used to effectively mitigate the impact of local variations in illumination on image contrast as defined in Equation 2. By dynamically adjusting threshold values for each region of interest, the algorithm compensates for varying lighting conditions and subtle textural nuances, ensuring accurate binary representation. The resultant binary images exhibit enhanced text contrast, with characters prominently highlighted against a dark background.

$$B(x,y) = \begin{cases} 0 & \text{if } I(x,y) > T(x,y) \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

- A subsequent inversion operation is executed on the thresholded images, as defined in Equation

3, which yields a binary representation with white characters prominently displayed against a black background, thereby facilitating enhanced character recognition and image quality.

$$I_{\text{inv}}(x, y) = 255 - B(x, y) \tag{3}$$

- Following the inversion of the images, a thorough examination of a representative sample of CAPTCHA images revealed that the upper and lower regions of each image were predominantly composed of noise and irrelevant information, whereas the central section contained the actual characters of interest (see Figure 5). Consequently,

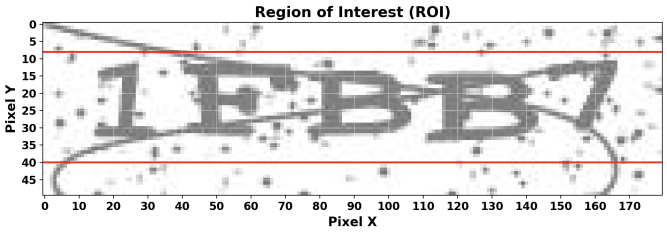

Figure 5. Region of Interest Identification: CAPTCHA image overlaid with a pixel-based grid. Red horizontal lines mark the top and bottom boundaries of the region containing relevant character data, effectively excluding noisy or irrelevant upper and lower areas.

a preprocessing step was implemented to isolate the region of interest (ROI) by converting the top and bottom portions of the image to a black background, thereby eliminating borders and extraneous features. The resulting image displayed pronounced noise manifestations in various forms, including isolated pixel clusters, disconnected fragments, and curved line artifacts that often intersected or overlapped character strokes, introducing fragmentation or spurious connections between characters. These anomalous elements, distinct from the actual character strokes, presented a significant challenge for accurate character segmentation. Consequently, the image quality at this stage remained suboptimal for reliable character recognition, necessitating further refinement through morphological operations.

- To address the image noise artifacts described previously, a sequence of morphological operations was implemented. The first step involved the application of a morphological closing operation using a 1×1 kernel to the thresholded image. This operation, performed before any further transformations, helped reconnect fragmented character strokes and is mathematically defined as:

$$A \bullet B = (A \oplus B) \ominus B \tag{4}$$

where A represents the binary image, B is the structuring element, $\oplus$ denotes dilation, $\ominus$ denotes erosion, and $\bullet$ represents the closing operation and bridges small gaps caused by noise, without significantly altering the overall character shapes. The selection of a small kernel ensured that only immediate neighboring pixels were considered, minimizing the unintended merging of adjacent characters. Following this, a 2×2 kernel was employed for two iterations of erosion, followed by two iterations of dilation. This step effectively removed isolated noise pixels and thinned the character strokes, while the dilation step restored the original stroke thickness without reintroducing the previously removed noise. This sequence can be mathematically expressed as:

$$A' = ((A \ominus B) \ominus B) \oplus B) \oplus B \tag{5}$$

where A is the input binary image, B is the 2×2 structuring element, $\ominus$ denotes erosion, $\oplus$ denotes dilation, and A' represents the resulting image after refinement.

- Next, a second morphological closing operation was performed using a 3×3 kernel, designed to fill in any remaining small holes or gaps in the character shapes, reinforcing the continuity of the character strokes. In the final step, a slimming operation was applied using a 1×1 kernel for erosion. This erosion step further thinned out any remaining unwanted pixels which ensured that only the most prominent parts of the characters were zealously preserved. All preprocessing steps are visualized in Figure 6.

3.3 Character Segmentation

The pre-processing stage, comprising binarization, noise reduction, and morphological transformations, were instrumental in rendering the CAPTCHA images into a format that facilitated accurate segmentation. This comprehensive treatment significantly enhanced character-background separation and mitigated the adverse effects of artifacts such as spurious noise and distortions. The resulting processed image (morph_image) served as input to the segmentation pipeline.

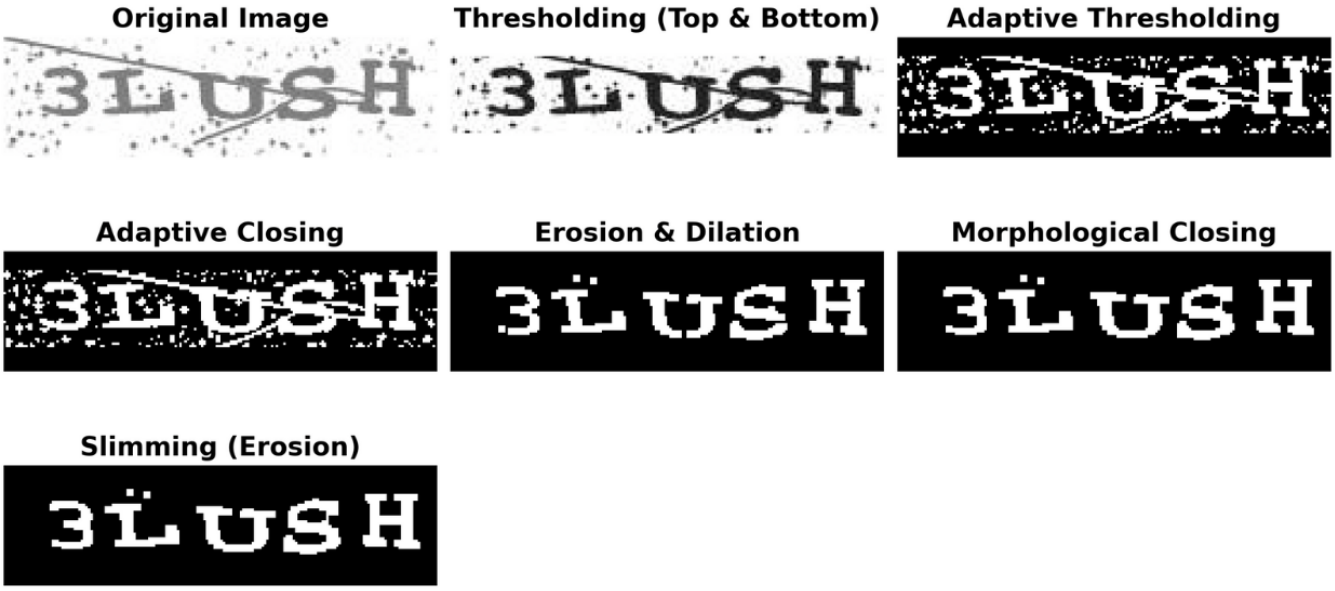

Figure 6. Preprocessing Steps Visualized: A graphical representation of the key preprocessing stages, showcasing techniques used to prepare data for analysis or model input.

The segmentation pipeline demonstrated resilience across a diverse set of images, effectively handling variations in character spacing, inherent image generation noise, and character irregularities, including overlapping or distorted glyphs. Connected component analysis provided a foundational framework for precise character region extraction, isolating potential character regions while excluding irrelevant artifacts and spurious background connections. Subsequent refinement steps, notably bounding box analysis and splitting, further enhanced segmentation accuracy by addressing challenging edge cases such as merged or excessively wide bounding boxes, thereby ensuring the accurate delineation of individual characters.

In this stage of character segmentation, connected components were identified in the binarized and morphologically processed image (referred to as slimmed_image) using the cv2.connectedComponentsWithStats function to segment individual characters. To isolate components representing valid characters, a filtering criterion based on the area was applied. Components were retained if their area fell within a predefined range (min_area=100 and max_area=800). This range was empirically chosen to exclude components that were either too small (representing background noise) or too large (noisy lines or merged character regions).

The bounding boxes of the valid components were then adjusted by adding padding. Specifically, padding was introduced around the bounding boxes to address issues observed during resizing, where characters like 'O' and '0' became stretched, leading to misclassification. This additional space ensured that the aspect ratio of such characters was preserved, preventing distortion and misclassification of characters. A constant padding size of 4 was added to the left, right, top, and bottom of the bounding boxes. The adjusted bounding boxes were then used to extract the character regions from the image.

Once the padding was applied, the bounding boxes of the valid components were sorted by their x-coordinate to maintain a left-to-right order, consistent with natural reading sequences.

To address instances of merged characters (2 characters joined by noise), an additional step was implemented: wide bounding boxes where the width exceeded twice the height ($w > h * 2$) were split into two separate components to handle cases where two adjacent characters were connected, leading to a single merged bounding box with twice the width. After splitting, the number of valid components was checked again. If fewer than five components were detected, indicating an incorrect segmentation, the image was discarded. This safeguard prevented erroneous segmentation from proceeding to subsequent stages of character recognition and prediction. Finally, the extracted character regions, adjusted for padding and split, were

adjusted to a consistent size dimension of 40×50 pixels using cv2.resize function to standardize the input size for subsequent stages. However, the pipeline also exhibited certain limitations. For instance, it was highly dependent on predefined thresholds for area filtering (min_area and max_area), which, while effective in most cases, could fail in multiple scenarios that could involve characters with exceptionally large or small areas due to extreme distortions. Similarly, in instances where even after bounding box splitting, fewer than five valid components could be identified—the image was discarded. This strict criterion ensured the quality of segmented characters but led to a higher rejection rate for CAPTCHA images that deviated significantly from the thresholds. Examples of Failed Images are shown in Figure 7.

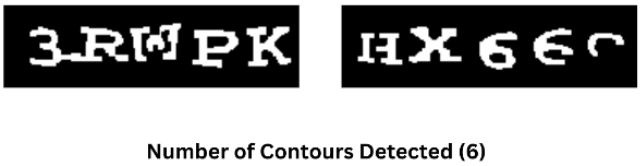

Figure 7. Contour Examples: Instances where contour detection algorithms fail, highlighting inaccuracies due to noise, incomplete edges, or complex shapes.

3.4 Image Augmentation

The initial dataset consisted of 3,000 CAPTCHA images, which after preprocessing was followed by character segmentation and low-quality sample elimination to produce a refined dataset of 14,480 individual character images. In particular, this size of the data set had inherent limitations in capturing diverse patterns, thereby leading to overfitting and poor generalization on out-of-sample data. In contrast, larger datasets such as MNIST [54] (60,000 samples) have highlighted the critical role of extensive and diverse training data in enabling models to generalize effectively to new and varied scenarios.

To mitigate this limitation, an image augmentation pipeline was designed and implemented to enrich the dataset with diverse variations, thereby preserving its fundamental characteristics. This process entailed generating additional variations of each character through controlled transformations, including random tilts ($\pm 6°$ rotation) and translations (up to ± 3 pixels). The resulting augmentation lead to a significant increase in dataset size to 101,345 images. By systematically augmenting our dataset with greater diversity, we sought to mitigate the risk of overfitting and improve the model's generalization capacity. The Character frequency distribution after augmentation is shown in Figure 8

3.5 Balancing Dataset Disparities

Following augmentation, the character class distribution was evaluated as presented in Figure 7, revealing a significant class imbalance. Characters such as 'I' and 'O' demonstrated a marked over-representation, and notably the digit '0', character 'F', 'A', 'R', and 'P', exhibited a substantially lower prevalence revealing a non-uniform distribution across character classes. With fewer examples of the digit '0' and other under-represented characters, the model risked becoming biased towards the more prevalent classes. To counter this potential bias, a targeted augmentation strategy was employed. Under-represented characters were augmented by minor geometric transformations, comprising rotations within a $\pm 6°$ range. Conversely, for over-represented classes, a subset of augmented samples was removed from the training set with the goal of achieving a more balanced class distribution. By implementing this approach, we have increased the dataset size to 102,669. Figure 9 shows the frequency of the character distribution before and after balancing.

3.6 CNN Model Architecture

The design of our model architecture was inspired by the striking similarities between CAPTCHA images and handwritten digits in MNIST. Nevertheless, the very large character set involved presented significant challenges associated with traditional classification methods. Therefore, careful consideration was given to developing a deep learning approach that could accurately classify these complex images. As a result, we selected a deep learning approach using a Convolutional Neural Network (CNN) as the most appropriate model for this study. CNNs excel at spatial feature extraction, making them well-suited for this task [55]. The architecture was attentively designed to balance accuracy with computational efficiency.

The core of the CNN consists of three convolutional blocks, each composed of a convolutional layer and a max-pooling layer. The consistent use of 3x3 kernels in all convolutional layers was motivated by their effectiveness in capturing local spatial features while minimizing computational overhead. ReLU activation functions were chosen for their non-linearity and ability to mitigate the vanishing gradient problem,

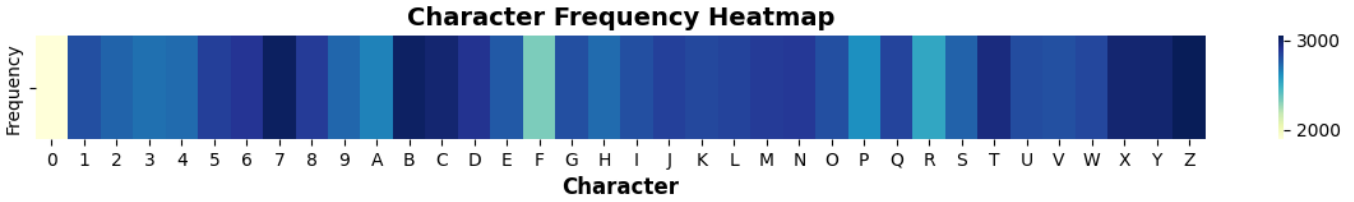

Figure 8. Character Frequency Heatmap: This heatmap illustrates the frequency distribution of characters within the dataset, enabling visual analysis of data imbalance and underlying textual trends.

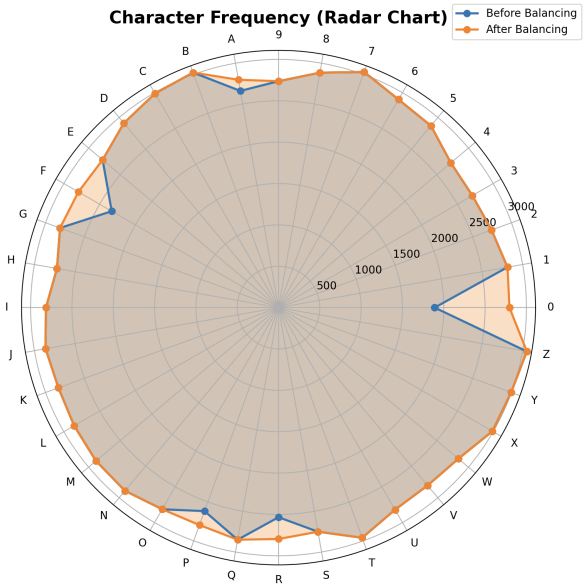

Figure 9. Radar chart of character frequency before and after balancing: A comparative visualization showcasing the distribution of character occurrences in the dataset before and after balancing. The radial layout highlights relative changes in frequency across all character classes.

which can hinder training of deeper networks. The increasing number of filters in each convolutional layer (32, 64, and 128, respectively) allows the network to learn a hierarchical representation of features, from simple edges and textures in the initial layers to more complex character-specific patterns in deeper layers. This hierarchical approach is inspired by the success of similar designs in image recognition tasks. Max-pooling, with a 2x2 pool size, is employed after each convolutional layer to downsample the feature maps. This reduces the dimensionality of the data, decreasing computational cost and increasing robustness to minor spatial variations. The specific choice of 2x2 pooling was determined empirically through cross-validation on a held-out portion of the training data, balancing information retention with dimensionality reduction.

Formally, the operations within each convolutional block are defined as:

- **Convolutional Layer:** The 2D convolution operation on an input feature map F with a kernel K produces an output feature map G. The value at position (i, j) is:

$$\begin{aligned} G(i,j) &= (F * K)(i,j) \\ &= \sum_m \sum_n F(i-m, j-n)\, K(m,n) + b \end{aligned} \tag{6}$$

  where b is a learnable bias term.

- **ReLU Activation:** The Rectified Linear Unit (ReLU) activation function is applied element-wise to the output of the convolutional layer:

$$\text{ReLU}(x) = \max(0, x) \tag{7}$$

- **Max-Pooling Layer:** The max-pooling operation reduces the spatial dimensions of the feature map. For a pooling window $\mathcal{W}$ of size 2×2, the output is:

$$P(i,j) = \max_{(m,n)\in\mathcal{W}_{ij}} F(m,n) \tag{8}$$

Following the output of convolutional blocks in the proposed model yield multiple two-dimensional feature maps, which encode specific spatial hierarchies extracted from the character images. These feature maps are rich in localized information, and have captured essential patterns such as edges, textures, and shapes at varying levels of abstraction. However, for this spatial information to inform classification effectively, it has to be transformed into a format compatible with the fully connected layers which require one-dimensional vector inputs. For which a flattening operation to reshape the multi-dimensional feature maps into a single concatenated vector was performed. This flattening operation maintains the spatial context learned by the convolutional layers while transforming the data into a format suitable for input to the fully connected layers. Mathematically, a feature map tensor $T \in \mathbb{R}^{H' \times W' \times C}$ is reshaped into

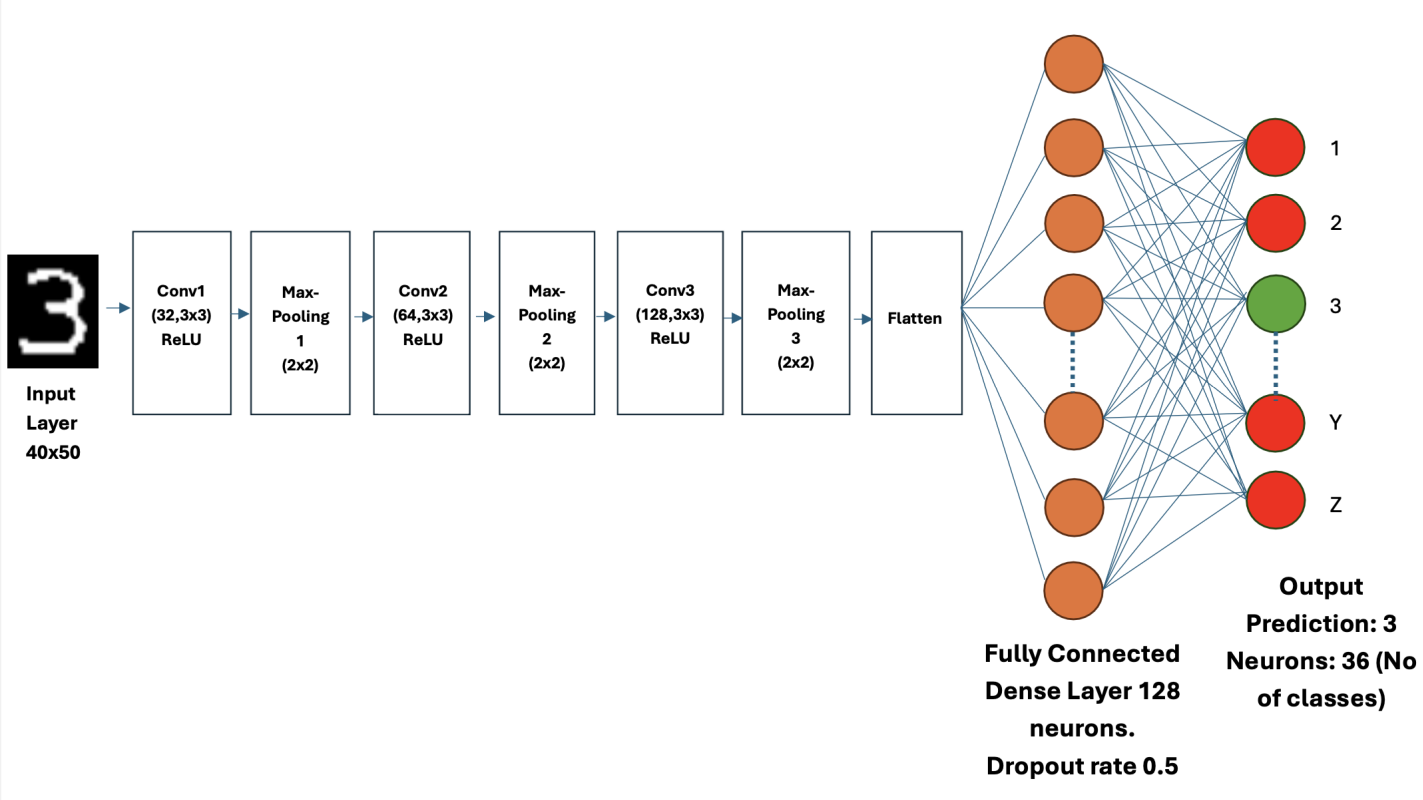

Figure 10. CNN Architecture: A deep learning model designed to automatically and adaptively learn spatial hierarchies of features from input data, primarily used for image and video recognition tasks.

a vector $V \in \mathbb{R}^{H' \cdot W' \cdot C}$, where H', W', and C are the height, width, and number of channels of the final feature map.

A fully connected layer of 128 units then refines this representation. This layer size was determined through extensive cross-validation that balances model capacity and generalization performance, preventing overfitting while ensuring that the network can capture the complex spatial relationships inherent in character images. This fully connected layer serves two key purposes. Integrates the localized features extracted by the convolutional layers into a global representation, allowing the model to analyze relationships between features throughout the image, regardless of their original spatial location. Furthermore, it captures non-linear dependencies within this integrated feature space, enabling discrimination between visually similar characters, which is a critical part of our research. The transformation is described by:

$$\mathbf{a} = \sigma(\mathbf{W}\mathbf{v} + \mathbf{b}) \tag{9}$$

where $\mathbf{v}$ is the input vector from the flattening layer, $\mathbf{W}$ is the weight matrix, $\mathbf{b}$ is the bias vector, σ is the activation function (ReLU in this case), and $\mathbf{a}$ is the output vector of activations.

ReLU activation was chosen for this layer over the more traditional sigmoid and tanh functions. Although sigmoid and tanh have been historically prevalent in neural networks, their susceptibility to vanishing gradients in deeper architectures hinder's training. ReLU's ability to mitigate this issue and facilitate faster convergence made it a more suitable choice for the fully connected layer.

To further amplify generalization and mitigate the risk of overfitting, a dropout layer is introduced after the fully connected layer to randomly deactivate a portion of the neurons during each training epoch. This stochastic deactivation forces the network to learn more robust features, preventing over-reliance on any single neuron. A dropout rate of 0.5 was selected based on empirical evaluation across different rates. This rate, determined through cross-validation on a held-out set, provided the optimal balance between regularization strength (preventing overfitting) and retention of learned information. During training, a binary mask $\mathbf{r}$ is generated from a Bernoulli distribution, where each element r_i has a probability p of being 0 (the dropout rate). The output $\mathbf{a}'$ is:

$$\mathbf{a}' = \mathbf{a} \odot \mathbf{r} \tag{10}$$

where $\odot$ denotes element-wise multiplication. During inference, the activations are scaled by $(1 - p)$ to compensate for the dropped units.

Finally, the network culminates in a dense output layer with 36 units, directly corresponding to the 36 character classes (0-9 and A-Z) within the CAPTCHA images. This layer provides the final classification scores for each character. To convert these scores into probabilities, a softmax activation function is applied. Softmax transforms the output into a probability distribution over all character classes. This enables the model to express varying degrees of confidence in its predictions. For a vector of raw output scores (logits) $\mathbf{z} = (z_1, z_2, ..., z_K)$ for $K = 36$ classes, the softmax function computes the probability $P(y = i|\mathbf{z})$ for each class i as:

$$P(y = i|\mathbf{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{11}$$

For model compilation, the Adam optimizer was selected due to its adaptive learning rate capabilities, demonstrably effective in training CNNs. To enhance the model convergence and to avoid premature plateauing, an exponential learning rate decay was implemented. The learning rate was initially set to 0.001, with a decay rate of 0.96, using a staircase schedule.

$$\alpha_t = 0.001 \times 0.96^{\left\lfloor \frac{\text{global\_step}}{10000} \right\rfloor} \tag{12}$$

This approach allows for larger initial weight changes in the early stages of training resulting in faster convergence, with progressively smaller adjustments as the model approaches optimal performance. Categorical cross-entropy serves as the loss function, as it is appropriate for multi-class classification. Model performance was monitored during training using accuracy on a validation set, and early stopping was employed to prevent over-fitting. The final model selection was based on achieving the highest validation accuracy. The Model architecture is visualized in Figure 10 and parameters summary is given in Table 2.

3.7 Model Training

The model was trained using a dataset of split character images from Telerik RadCaptcha, containing 36 classes: 10 digits and 26 uppercase letters. This dataset closely resembles the structure of the MNIST dataset, which is commonly used for digit classification, but with the addition of uppercase letters. The training was conducted using the Keras library, with TensorFlow as the backend, and was executed on a MacBook Air M1 with 8gb ram. We employed a data partitioning strategy that allocated 70% of the dataset to training, 15% to validation testing and 15% to test dataset to ensure reliable evaluation and mitigate over-fitting. For training, the Adam optimizer was utilized, with categorical cross-entropy serving as the loss function and accuracy, F1 score and recall scores being used to assess model performance. Training process was conducted over 8 epochs with a batch size of 32.

Table 2. Model architecture and parameter summary.

Layer (Type)	Output Shape	Number of Params
Conv2D	(None, 38, 48, 32)	320
MaxPooling2D	(None, 19, 24, 32)	0
Conv2D_1	(None, 17, 22, 64)	18,496
MaxPooling2D_1	(None, 8, 11, 64)	0
Conv2D_2	(None, 6, 9, 128)	73,856
MaxPooling2D_2	(None, 3, 4, 128)	0
Flatten	(None, 1536)	0
Dense	(None, 128)	196,736
Dropout	(None, 128)	0
Dense_1	(None, 36)	4,644
Total Parameters		**882,158**
Trainable Parameters		**294,052**
Non-Trainable Parameters		**0**
Optimizer Parameters		**588,106**

4 Results and Discussion

This section presents the performance analysis of the proposed CNN architecture for Telerik RadCaptcha character recognition. We evaluate the model's ability to classify segmented character images across 36 classes, encompassing numerical digits (0-9) and uppercase letters (A-Z). After training for 8 epochs, the proposed model achieved a training accuracy of 99.26% and a validation accuracy of 99.37%. This remarkably high and consistent performance across both datasets demonstrates the model's exceptional ability to learn discriminative features from the character images and generalize effectively to unseen data. The negligible difference between training and validation accuracy indicates that the model is not over-fitting and that the dropout regularization technique employed, was successful in preventing over-reliance on the training data.

Figure 11 depicts the training and validation performance of the model over 8 epochs, showcasing accuracy metrics. The accuracy curves show a corresponding rapid increase in both the training and validation accuracy. Notably, both training and validation accuracies reach high levels early in the training process and plateau near their peak values.

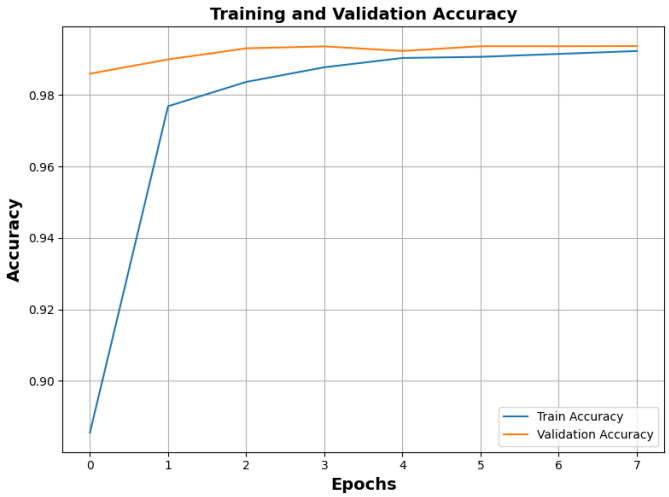

Figure 11. Training and Validation Accuracy.

The close tracking of the validation accuracy with the training accuracy throughout the training process reveals the absence of significant over-fitting and indicates that the model is learning original patterns from the data rather than memorizing the training examples.

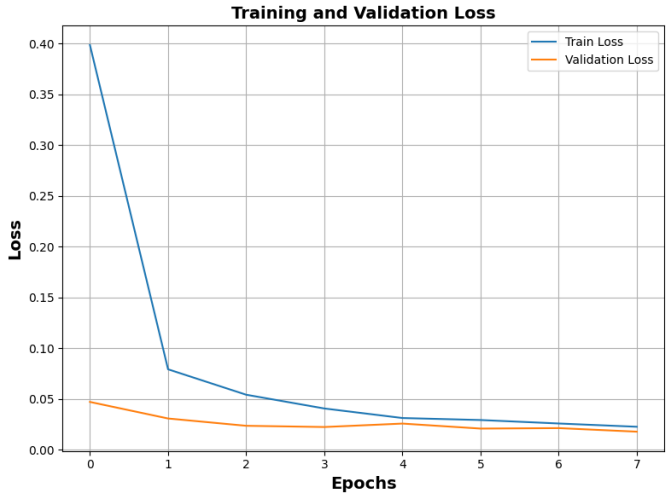

Figure 12. Training and Validation Loss

The loss curves in Figure 12 demonstrate a rapid decrease in both training and validation loss during the initial epochs, indicating effective learning. By the 4th epoch, the loss stabilizes and plateaus, showing negligible further reduction. The close alignment between training and validation loss throughout the training process reflects the strong generalization of the model to unseen data, without observable signs of over-fitting in the presented results.

The character prediction model was evaluated on a test dataset of 580 unseen Telerik RadCaptcha-labeled images. The initial stage of our methodology is a preprocessing step where individual characters are segmented from each CAPTCHA image. To quantify the performance of this stage, we define preprocessing accuracy as the percentage of CAPTCHA images that are correctly segmented. A CAPTCHA is considered correctly segmented if the number of characters isolated by our algorithm exactly matches the known number of characters in its ground truth label. This accuracy is calculated using the following formula:

$$\text{Preprocessing Accuracy} = \frac{\text{Number of Correctly Segmented Images}}{\text{Total Number of Images}} \times 100 \tag{13}$$

Following this protocol, our algorithm processed all 580 images and successfully segmented 577 of them, resulting in a preprocessing accuracy of 99.48%. These 577 correctly processed images yielded the 2,881 individual character instances that formed the final test set. This test set was then used to evaluate the performance of the character prediction model. The explicit definitions for the performance metrics used are provided below.

For evaluating segmented characters, we used the following standard metrics. **Accuracy** is the ratio of correctly predicted characters to the total number of characters. **Precision** is the ratio of true positives to the sum of true and false positives, measuring prediction exactness. **Recall** is the ratio of true positives to the sum of true positives and false negatives, measuring the model's ability to find all relevant instances. The **F1-score** is the harmonic mean of precision and recall.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{14}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

Here, TP, FP, and FN represent the counts of True Positives, False Positives, and False Negatives, respectively, aggregated across all character classes. For full CAPTCHA sequences, the **overall accuracy** was computed as the percentage of entire CAPTCHA strings where every character was correctly predicted.

Table 3 summarizes the model's performance on the held-out test set. We achieved 97.60% test accuracy,

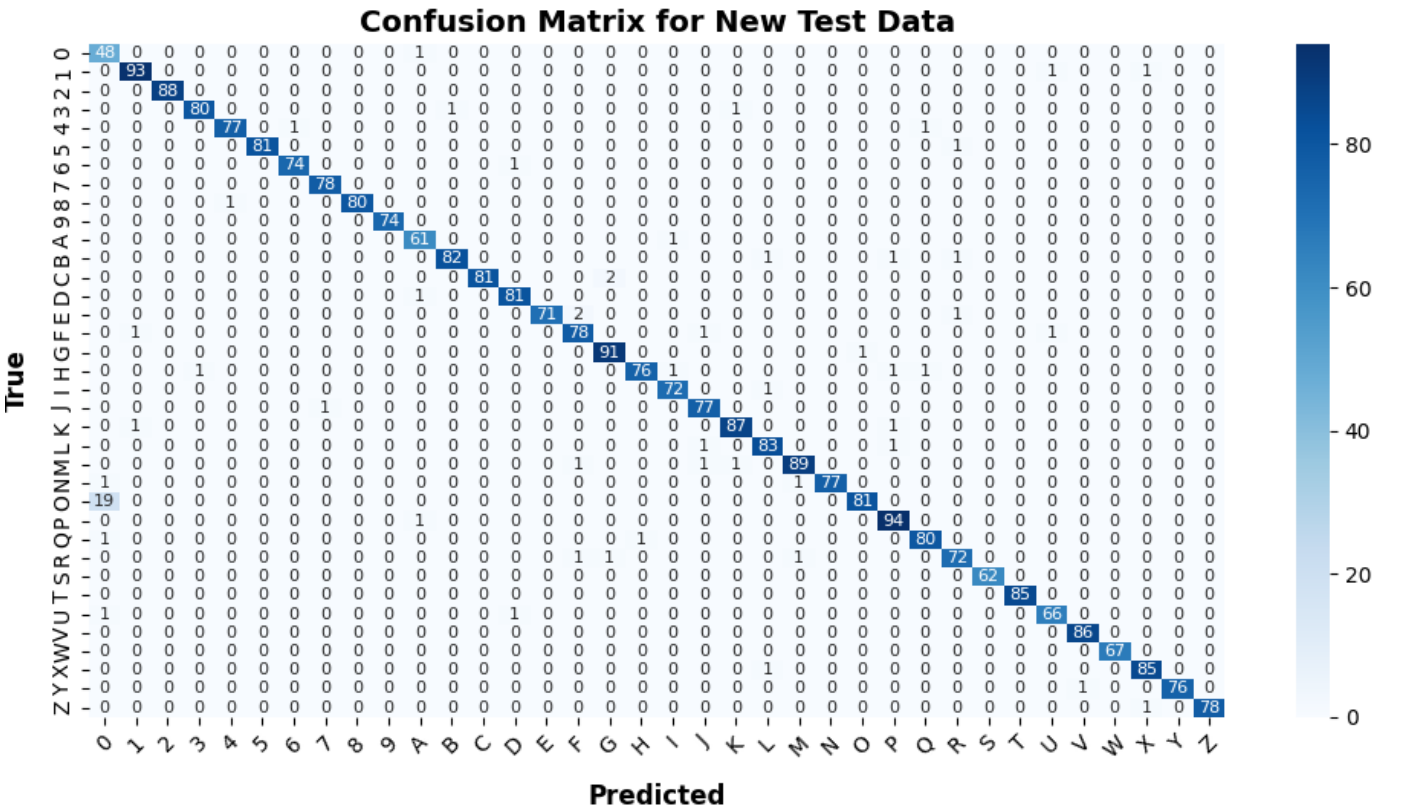

Figure 13. Confusion matrix on test data: A tool to evaluate model accuracy by comparing predicted and true labels in the test dataset.

Table 3. Evaluation metrics for the model.

Metric	Value
Test Accuracy	0.9760
Precision	0.9784
Recall	0.9760
F1 Score	0.9764

slightly below the training (99.26%) and validation (99.37%) accuracies and indicates minimal overfitting. The 97.84% precision and 97.60% recall contribute to a balanced F1-score of 97.64% which highlights the model's ability to generalize to new, unseen data. We consider the minor difference between training, validation, and test accuracy to be a small but acceptable generalization gap.

While the preceding analysis and Table 3 detailed the model's efficacy at the character level, practical CAPTCHA breaking required accurate decryption of the entire sequence of characters for which we conducted further evaluation on test set of 580 complete Telerik RadCaptcha images in which each image comprised of five characters, demanding successful prediction of the entire sequence for a positive outcome. This holistic evaluation methodology differs from assessing individual character accuracy, providing a direct measure of the model's real-world CAPTCHA breaking capability. The results showed that our model achieved 92.08% accuracy in correctly predicting all five characters within each RadCaptcha, albeit lower than the individual character accuracy. Notably, this accuracy is based on raw, unprocessed RadCaptcha images that undergo several stages of processing - preprocessing, character segmentation, and final character recognition by the model. Errors introduced at any stage of this pipeline can compound and contribute to the lower overall CAPTCHA-solving accuracy. For instance, imperfect character segmentation could lead to misclassification even if the character recognition model performs well in isolation. This evaluation method offers a more informed assessment of the model's performance, taking into account the complexities of real-world CAPTCHA images.

Theoretical vs. Empirical Performance

A theoretical baseline for sequence-level accuracy can be established by modeling the character predictions as independent Bernoulli trials. Given a single-character recognition accuracy of $p_c = 0.9760$, the probability of correctly identifying a sequence of five independent

characters is:

$$P(\text{Sequence Success}) = p_c^5 = (0.9760)^5 \approx 0.8858 \tag{17}$$

This model predicts a theoretical accuracy of 88.58%. Our empirically observed accuracy of 92.08% is substantially higher, which suggests that character recognition errors may not be perfectly independent and necessitates a more granular, two-stage analysis of the processing pipeline.

Quantitative Error Attribution

The accuracy of the final system is conditional on the success of two sequential stages: character segmentation and character recognition. The overall probability of solving a CAPTCHA can be formally expressed as follows.

$$\begin{aligned} P(\text{Solve}) = {} & P(\text{Recog Success} \mid \text{Seg Success}) \\ & \times P(\text{Seg Success}) \end{aligned} \tag{18}$$

We decomposed the 46 unsolved CAPTCHAs to attribute failures to each stage.

1. Segmentation Failures This constitutes the first potential source of error. Of the 580 total images, 3 were incorrectly segmented by our preprocessing algorithm. The probability of successful segmentation is therefore:

$$P(\text{Seg Success}) = \frac{580 - 3}{580} = \frac{577}{580} \approx 0.9948 \tag{19}$$

These segmentation failures represent an upper bound on system performance and account for 6.5% (3/46) of total CAPTCHA-level errors.

2. Character Recognition Failures This second error category applies to the 577 correctly segmented images, of which 43 contained at least one misclassified character. The conditional probability of recognition success, given a successful segmentation, is:

$$\begin{aligned} P(\text{Recog Success} \mid \text{Seg Success}) &= \frac{577 - 43}{577} \\ &= \frac{534}{577} \\ &\approx 0.9255 \end{aligned} \tag{20}$$

Consequently, recognition failures constitute the predominant error source, accounting for 93.5% (43/46) of all unsolved CAPTCHAs.

Synthesis and Conclusion

The product of these observed probabilities confirms our empirical measurements.

$$P(\text{Solve}) \approx 0.9255 \times 0.9948 \approx 0.9209 \tag{21}$$

This result aligns perfectly with our observed system accuracy of 92.08%, thereby validating our two-stage error attribution model.

In conclusion, our analysis yields two primary findings. First, a simple probabilistic model assuming error independence provides a conservative baseline but fails to capture the system's full dynamics. Second, our quantitative pipeline analysis demonstrates that while the segmentation stage is highly reliable (99.48%), the principal contributor to performance degradation is character recognition error within correctly segmented images. A qualitative review identified the misclassification of visually ambiguous characters (e.g., 'O' vs. '0', 'l' vs. 'I') as the primary cause. Therefore, future research should prioritize enhancing the model's discriminative capabilities for these challenging cases to mitigate the discrepancy between character- and sequence-level accuracy.

This performance, while achieving a complete CAPTCHA solution accuracy of 92.08%, compares with related works. [31] report higher accuracies, up to 98.94%, using a CNN architecture; however, their model benefited from training on a substantially larger dataset of synthetic CAPTCHAs, potentially lacking the diversity and noise characteristics inherent in real-world examples like Telerik RadCaptchas. Similarly, [40] achieve an impressive 99.8% accuracy with a DWSCNN, but their focus is on simpler CAPTCHAs with no noise and distortion than those presented by Telerik's implementation. The work of [34], using a skip-connection CNN on 5-character CAPTCHAs, is a closer comparison. However, they achieve a lower overall accuracy of 85.52%. This reduced performance may be attributed to the nature of their dataset, which exhibits a horizontal line artifact bisecting characters which thereby significantly increases segmentation difficulty. This artifact, unlike the thinner lines present in our Telerik RadCaptcha dataset, introduces a systematic challenge that likely hinders their character segmentation and subsequent recognition performance, regardless of model architecture. Our approach, while employing a moderately sized dataset, still achieves moderate performance. In essence, the inherent complexity of Telerik RadCaptchas presents a more demanding test

scenario, making direct accuracy comparisons less meaningful without acknowledging the differences in CAPTCHA complexity across these studies.

Figure 13 represents the confusion matrix of our model predictions on test set and the detailed classification performance on all the 36 character classes identified in the dataset. The strong diagonal dominance of the matrix visually confirms the high overall accuracy, with the majority of predictions correctly aligning with the true labels. While most of the off-diagonal entries that represent misclassifications are negligible, the most prominent errors occur between the characters 'O', with 19 instances of 'O' being misclassified as '0'. This confusion arises from the significant visual similarity between the characters '0' and 'O', which is challenging even for human annotators and users to distinguish in certain cases. All other misclassifications are negligible which indicates that our model can effectively discriminates between character classes on unseen data.

5 Practical Deployment and Security Implications

Beyond theoretical performance metrics, the true measure of our proposed CAPTCHA-breaking framework lies in its practical applicability against live systems. To demonstrate its applied relevance, we outline a deployment scenario targeting the Telerik RadCaptcha implementation on the NUST Merit Portal. This portal protects sensitive student admission data, making it a prime example of a system where automated attacks can be executed.

The practical deployment of our framework would follow the following workflow:

1. **Automated Navigation and Image Acquisition:** A script, using a web automation tool like Selenium or Puppeteer, navigates to the portal's page containing the Telerik RadCaptcha. The script then isolates the CAPTCHA image element and downloads the image.

2. **Preprocessing and Segmentation:** The captured image is passed directly to our specialized preprocessing pipeline. This crucial stage applies the tailored algorithms used in this study to isolate the individual characters, overcoming the specific noise and character merging defenses used by Telerik RadCaptcha.

3. **Character Recognition and Submission:** Each segmented character image is fed into our trained CNN model for prediction. The resulting character predictions are concatenated to form the complete CAPTCHA string. The automation script then submits this string into the web form.

The high success rates achieved in our experiments, specifically the accuracy of 92. 08% on full unseen CAPTCHA sequences, directly translates into a high probability of successfully bypassing the portal's security on any given attempt. An attacker could deploy this framework at scale, making thousands of attempts in a short period to scrape sensitive merit lists, test stolen user credentials, or access other private data not intended for public or automated access.

This practical scenario underscores a critical security vulnerability. The effectiveness of our framework demonstrates that conventional static image-based CAPTCHAs like Telerik RadCaptcha are no longer a reliable defense mechanism against determined, automated threats. This highlights the urgent need for websites to migrate towards more dynamic and robust security measures, such as interactive challenges, risk-based analysis, or behavioral biometrics.

6 Limitations

Despite our model's high accuracy, there are several inherent limitations in scenarios where CAPTCHA images contains intermingled lines of similar thickness to the characters appearing close together, the preprocessing step may fail, leading to incomplete predictions. Additionally, this model is designed specifically for Telerik RadCaptcha and does not generalize to other CAPTCHA systems. Another limitation is the confusion between visually similar characters, such as the letter 'O' and the digit '0', which can be challenging even for human observers under certain conditions. Thus, the model's applicability is limited to Telerik RadCaptcha and may not perform effectively on other CAPTCHA formats.

7 Conclusion

This study introduces a novel dataset of 3,000 CAPTCHA images, laying the groundwork for advancing character recognition research in challenging real-world contexts. Our CNN achieved a commendable accuracy of 97.67% on segmented characters but faced a notable drop to 92.08% when tested on full CAPTCHA images, emphasizing the critical dependency on effective preprocessing.

This outcome highlights a fundamental challenge: the entire system hinges on accurate segmentation.

Even minor segmentation errors or character ambiguities—such as distinguishing 'O' from '0'—can cascade into significant performance degradation. These limitations expose the fragility of current approaches when applied to real-world scenarios with noisy and variable data.

Despite these challenges, our findings reveal promising directions for improvement. Future research should prioritize the development of a versatile, one-size-fits-all CAPTCHA-solving approach capable of handling a wide variety of CAPTCHA types, whether noisy, clean, segmented, or unsegmented. By moving beyond reliance on precise preprocessing and segmentation, researchers can aim to design systems that adapt to real-world variability and perform reliably in diverse scenarios. Incorporating error correction mechanisms and rigorous testing on broader, more diverse datasets will be key to advancing the practicality and effectiveness of CAPTCHA solvers.

Moreover, our work raises questions about the broader resilience of CAPTCHA-solving systems. Addressing vulnerabilities to adversarial attacks and expanding the scope to handle varied CAPTCHA designs will be key steps toward creating adaptable and practical solutions.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported without any funding.

Conflicts of Interest

The authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Von Ahn, L., Blum, M., & Langford, J. (2004). Telling humans and computers apart automatically. *Communications of the ACM, 47*(2), 56-60. [CrossRef]

[2] Reddy, A., & Cheng, Y. (2024). User Perception of CAPTCHAs: A Comparative Study between University and Internet Users. *arXiv preprint arXiv:2405.18547*.

[3] Turing, A. M. (2007). Computing machinery and intelligence. In *Parsing the Turing test: Philosophical and methodological issues in the quest for the thinking computer* (pp. 23-65). Dordrecht: Springer Netherlands. [CrossRef]

[4] Singh, T., Kumar, A., & Goel, P. (2024, September). Analysis of Text-CAPTCHA Using Machine Learning. In *2024 International Conference on Communication, Computing and Energy Efficient Technologies (I3CEET)* (pp. 238-243). IEEE. [CrossRef]

[5] Guerar, M., Verderame, L., Migliardi, M., Palmieri, F., & Merlo, A. (2021). Gotta CAPTCHA'Em all: a survey of 20 Years of the human-or-computer Dilemma. *ACM Computing Surveys (CSUR), 54*(9), 1-33. [CrossRef]

[6] Udoidiok, I., & Zhang, J. (2024, October). When XAI Meets CAPTCHA: A Case Study. In *2024 Cyber Awareness and Research Symposium (CARS)* (pp. 1-6). IEEE. [CrossRef]

[7] Kumar, M., Jindal, M. K., & Kumar, M. (2022). A systematic survey on CAPTCHA recognition: types, creation and breaking techniques. *Archives of Computational Methods in Engineering, 29*(2), 1107-1136. [CrossRef]

[8] Sharma, S., & Singh, D. (2024, March). Captcha in web security and deep-captcha configuration based on machine learning. In *2024 3rd International Conference for Innovation in Technology (INOCON)* (pp. 1-6). IEEE. [CrossRef]

[9] 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). [CrossRef]

[10] Chellapilla, K., & Simard, P. (2004). Using machine learning to break visual human interaction proofs (HIPs). *Advances in neural information processing systems, 17*.

[11] Sivakorn, S., Polakis, J., & Keromytis, A. D. (2016). I'm not a human: Breaking the Google reCAPTCHA. *Black Hat, 14*, 1-12.

[12] Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C., & Mitchell, J. (2011, May). The failure of noise-based non-continuous audio captchas. In *2011 IEEE symposium on security and privacy* (pp. 19-31). IEEE. [CrossRef]

[13] *ASP.NET AJAX Captcha - RadControls for web forms* | Telerik UI for ASP.NET AJAX. (n.d.). Telerik.com. Retrieved from https://www.telerik.com/products/aspnet-ajax/captcha.aspx

[14] Holman, J., Lazar, J., Feng, J. H., & D'Arcy, J. (2007, October). Developing usable CAPTCHAs for blind users. In *Proceedings of the 9th international ACM*

SIGACCESS conference on Computers and accessibility (pp. 245-246). [CrossRef]

[15] Xing, W., Mohd, M. R. S., Johari, J., & Ruslan, F. A. (2023, June). A Review on Text-based CAPTCHA Breaking Based on Deep Learning Methods. In *2023 International Conference on Computer Engineering and Distance Learning* (*CEDL*) (pp. 171-175). IEEE. [CrossRef]

[16] Deng, X., Zhao, R., Xue, Z., Liu, M., Chen, L., & Wang, Y. (2021, October). A Semi-supervised Deep Learning-Based Solver for Breaking Text-Based CAPTCHAs. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications* (*TrustCom*) (pp. 614-619). IEEE. [CrossRef]

[17] Mistry, R., Thatte, G., Waghela, A., Srinivasan, G., & Mali, S. (2021, October). DeCaptcha: Cracking captcha using Deep Learning Techniques. In *2021 5th International Conference on Information Systems and Computer Networks* (*ISCON*) (pp. 1-6). IEEE. [CrossRef]

[18] Aiken, W., & Kim, H. (2018, May). POSTER: DeepCRACk: Using deep learning to automatically crack audio CAPTCHAs. In *Proceedings of the 2018 on Asia conference on computer and communications security* (pp. 797-799). [CrossRef]

[19] Sivakorn, S., Polakis, I., & Keromytis, A. D. (2016, March). I am robot:(deep) learning to break semantic image captchas. In *2016 IEEE European Symposium on Security and Privacy* (*EuroS&P*) (pp. 388-403). IEEE. [CrossRef]

[20] Dou, Z. (2021, June). The text captcha solver: A convolutional recurrent neural network-based approach. In *2021 International Conference on Big Data Analysis and Computer Science* (*BDACS*) (pp. 273-283). IEEE. [CrossRef]

[21] Pattabiraman, V., & Maheswari, R. (2022). Image to Text Processing Using Convolution Neural Networks. In *Recurrent Neural Networks* (pp. 43-52). CRC Press.

[22] Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., & Shet, V. (2013). Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*.

[23] Telerik web forms Captcha overview - Telerik UI for ASP.NET AJAX. (n.d.). Telerik & Kendo UI - .NET Components Suites & JavaScript UI Libraries. Retrieved from https://www.telerik.com/products/aspnet-ajax/documentation/controls/captcha/overview

[24] Jähne, B. (2005). *Digital image processing*. Berlin, Heidelberg: Springer Berlin Heidelberg.

[25] Gao, H., Wang, W., Qi, J., Wang, X., Liu, X., & Yan, J. (2013, November). The robustness of hollow CAPTCHAs. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security* (pp. 1075-1086). [CrossRef]

[26] Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural computation*, *22*(12), 3207-3220. [CrossRef]

[27] Shirali-Shahreza, M., & Shirali-Shahreza, S. (2007, December). CAPTCHA for blind people. In 2007 IEEE international symposium on signal processing and information technology (pp. 995-998). IEEE. [CrossRef]

[28] Wang, T., Wu, D. J., Coates, A., & Ng, A. Y. (2012, November). End-to-end text recognition with convolutional neural networks. In *Proceedings of the 21st international conference on pattern recognition* (*ICPR2012*) (pp. 3304-3308). IEEE.

[29] Zhu, B. B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., ... & Cai, K. (2010, October). Attacks and design of image recognition CAPTCHAs. In *Proceedings of the 17th ACM conference on Computer and communications security* (pp. 187-200). [CrossRef]

[30] Aguilar, D., Riofrío, D., Benítez, D., Pérez, N., & Moyano, R. F. (2021, October). Text-based CAPTCHA vulnerability assessment using a deep learning-based solver. In *2021 IEEE Fifth Ecuador Technical Chapters Meeting* (*ETCM*) (pp. 1-6). IEEE. [CrossRef]

[31] Noury, Z., & Rezaei, M. (2020). Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment. *arXiv preprint arXiv:2006.08296*.

[32] Kumar, M., Jindal, M. K., & Kumar, M. (2023). An efficient technique for breaking of coloured Hindi CAPTCHA. *Soft Computing*, *27*(16), 11661-11686. [CrossRef]

[33] Wei, L., Li, X., Cao, T., Zhang, Q., Zhou, L., & Wang, W. (2019, February). Research on optimization of CAPTCHA recognition algorithm based on SVM. In *Proceedings of the 2019 11th International Conference on Machine Learning and Computing* (pp. 236-240). [CrossRef]

[34] Lu, S., Huang, K., Meraj, T., & Rauf, H. T. (2022). A novel CAPTCHA solver framework using deep skipping Convolutional Neural Networks. *PeerJ Computer Science*, *8*, e879. [CrossRef]

[35] Derea, Z., Zou, B., Kui, X., Thobhani, A., & Abdussalam, A. (2025). A Dual-Layer Attention Based CAPTCHA Recognition Approach with Guided Visual Attention. *Computer Modeling in Engineering & Sciences* (*CMES*), *142*(3). [CrossRef]

[36] Zhang, N., Ebrahimi, M., Li, W., & Chen, H. (2020, November). A generative adversarial learning framework for breaking text-based captcha in the dark web. In *2020 IEEE International conference on intelligence and security informatics* (*ISI*) (pp. 1-6). IEEE. [CrossRef]

[37] Kumar, D., Singh, R., & Bamber, S. S. (2022). Your CAPTCHA Recognition Method Based on DEEP Learning Using MSER Descriptor. *Computers, Materials & Continua*, *72*(2). [CrossRef]

[38] Derea, Z., Zou, B., Al-Shargabi, A. A., Thobhani, A., & Abdussalam, A. (2023). Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy. *Sensors, 23*(23), 9487. [CrossRef]

[39] Wan, X., Johari, J., & Ruslan, F. A. (2024). Adaptive captcha: a CRNN-based text captcha solver with adaptive fusion filter networks. *Applied Sciences, 14*(12), 5016. [CrossRef]

[40] Dankwa, S., & Yang, L. (2021). An efficient and accurate depth-wise separable convolutional neural network for cybersecurity vulnerability assessment based on CAPTCHA breaking. *Electronics, 10*(4), 480. [CrossRef]

[41] Bhowmick, R. S., Indra, R., Ganguli, I., Paul, J., & Sil, J. (2023). Breaking CAPTCHA system with minimal exertion through deep learning: Real-time risk assessment on Indian government websites. *Digital Threats: Research and Practice, 4*(2), 1-24. [CrossRef]

[42] Yu, N., & Darling, K. (2019). A low-cost approach to crack python CAPTCHAs using AI-based chosen-plaintext attack. *Applied sciences, 9*(10), 2010. [CrossRef]

[43] Bostik, O., Horak, K., Kratochvila, L., Zemcik, T., & Bilik, S. (2021). Semi-supervised deep learning approach to break common CAPTCHAs. *Neural Computing and Applications, 33*(20), 13333-13343. [CrossRef]

[44] Tian, S., & Xiong, T. (2020, April). A generic solver combining unsupervised learning and representation learning for breaking text-based captchas. In Proceedings of The Web Conference 2020 (pp. 860-871). [CrossRef]

[45] Kovács, Á., & Tajti, T. CAPTCHA recognition using machine learning algorithms with various techniques. In *Annales Mathematicae et Informaticae* (pp. 81-91). [CrossRef]

[46] Derea, Z., Zou, B., Kui, X., Abdullah, M., Thobhani, A., & Abdussalam, A. (2025). A Novel CAPTCHA Recognition System Based on Refined Visual Attention. *Computers, Materials & Continua, 83*(1). [CrossRef]

[47] Lin, G., Liang, Y., Chen, Y., & Pan, W. (2022, May). Configurable image recognition framework design based on KNN and bit-based similarity model. In *International Conference on Computer Application and Information Security (ICCAIS 2021)* (Vol. 12260, pp. 396-402). SPIE. [CrossRef]

[48] UmaMaheswari, P., Ezhilarasi, S., Harish, P., Gowrishankar, B., & Sanjiv, S. (2020, December). Designing a text-based CAPTCHA breaker and solver by using deep learning techniques. In *2020 IEEE international conference on advances and developments in electrical and electronics engineering (ICADEE)* (pp. 1-6). IEEE. [CrossRef]

[49] Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Berlin, Heidelberg: Springer Berlin Heidelberg. [CrossRef]

[50] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*(1), 1929-1958.

[51] National University of Sciences and Technology (NUST). (2024). Merit Search. Retrieved from https://ugadmissions.nust.edu.pk/result/meritsearch.aspx

[52] Tbogamer22. (2024). 5-Characters CAPTCHA Labeled Dataset. Retrieved from https://www.kaggle.com/datasets/tbogamer22/5characters-captcha-labeled-dataset

[53] Brweb. (n.d.). BT.601 : Studio encoding parameters of digital television for standard 4:3 and wide screen 16:9 aspect ratios. Retrieved from https://www.itu.int/rec/R-REC-BT.601

[54] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (2002). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324. [CrossRef]

[55] Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging, 9*(4), 611-629. [CrossRef]

Mr. Talha Bin Omar is a Data Science student at Air University Islamabad, Pakistan. His research interests include data analytics, deep learning, artificial intelligence, and applying data-driven methodologies to solve complex real-world problems.

Mr. Tahir Sher pursuing his PhD in Artificial Intelligence from Korea University, Seoul Campus, Korea. He completed MS in Data Science from Air University and bachelor's degree in mathematics from the International Islamic University, Islamabad, where he graduated with a gold medal and received both distinction and position certificates. With extensive teaching experience across various institutions, Mr. Sher specializes in delivering courses such as Calculus, Ordinary and Partial Differential Equations, Numerical Analysis, Linear Algebra, and Statistical & Mathematical Methods for Data Science. His students hail from Pakistan and allied countries. As a research scholar in the Explainable AI Research Group at Air University, his areas of interest include the Internet of Things (IoT), Social IoT, Machine Learning, Deep Learning, Natural Language Processing (NLP), Data Analysis, Time Series Analysis, Mathematical Modeling for Decision-Making, Social Media Analysis, Federated Learning, and Computer Vision. Committed to advancing interdisciplinary research, Mr. Tahir Sher strives to bridge the fields of computer science and human-centered disciplines. (Email: 2025010294@korea.ac.kr)

Dr. Abdul Rehman is currently a Research Professor at the Human Data Convergence Institute, Jeonju University, South Korea. He previously worked as a Postdoctoral Research Associate at the Hyper-Connectivity Convergence Technology Research Center, Kyungpook National University (KNU), Daegu, South Korea, and served as an Assistant Professor at the University of Central Punjab, Lahore, Pakistan. He holds a bachelor's degree in Mathematics from the International Islamic University, Islamabad, Pakistan, and an Integrated Ph.D. in Computer Science and Engineering from KNU. His research interests include Deep Learning, Machine Learning, Data Science, and their applications in Computer Vision, Smart IoT (S-IoT), and Data Analysis. He also explores Complex Network Navigation, Mathematical Modeling, and Big Data Analytics. Dr. Rehman received the KINGS Scholarship for his Ph.D. studies and was honored with the "Outstanding Researcher" Award from the School of Computer Science and Engineering at KNU. He serves as a guest editor and editorial board member for several international journals and has contributed to numerous international conferences as a session chair and publication chair. More information on Dr. A. Rehman is available at https://sites.google.com/view/drrehman/home.

Mr. M. Haroon Khan is a Data Science student at Air University Islamabad, Pakistan. His research interests focus on data analytics, deep learning, and Natural Language Processing.