

A REPORT ON
DEVELOP A ML BASED SOLUTION TO REFINE CAPTCHA

Submitted by,

SHAIK ASLAM - 20211CEI0042
GILAJIRLA SUJITHA REDDY - 20211CEI0133
SANIVARAPU VISWESWAR - 20211CEI0161

Under the guidance of,

Ms. AMIRTHA PREEYA V

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

At



PRESIDENCY UNIVERSITY
BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

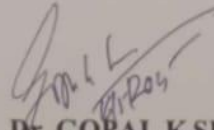
PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

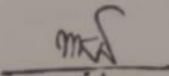
This is to certify that the Project report “**DEVELOP A ML BASED SOLUTION TO REFINE CAPTCHA**” being submitted by “Shaik Aslam, Sujitha Reddy, Visweswar Reddy” bearing roll numbers “20211CEI0042,20211CEI0133,20211CEI0161” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Engineering [Artificial Intelligence and Machine Learning] is a Bonafide work carried out under my supervision.



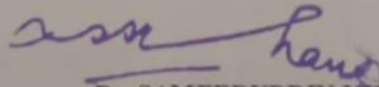
Ms. AMIRTHA PREEYA V
Assistant Professor
Presidency School of
CSE&IS
Presidency University



Dr. GOPAL K SHYAM
Professor & HoD
Presidency School of
CSE&IS
Presidency University



Dr. MYDHILI NAIR
Associate Dean
Presidency School of
CSE
Presidency University



Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean - Presidency School of
CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **DEVELOP A ML BASED SOLUTION TO REFINE CAPTCHA** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Engineering [Artificial Intelligence and Machine Learning]**, is a record of our own investigations carried under the guidance of, **Ms. Amirtha Preeya V, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Details of students:

S.no	Name(s)	Roll Number(s)	Signature(s)
1.	Shaik Aslam	20211CEI0042	S. Aslam
2.	Gilajirra Sujitha Reddy	20211CEI0133	G. Sujitha
3.	Sanivarapu Visweswar Reddy	20211CEI0161	G. Visweswar Reddy

A REPORT ON
DEVELOP A ML BASED SOLUTION TO REFINE CAPTCHA

Submitted by,
SHAIK ASLAM - 20211CEI0042
GILAJIRLA SUJITHA REDDY - 20211CEI0133
SANIVARAPU VISWESWAR - 20211CEI0161

Under the guidance of,
Ms. AMIRTHA PREEYA V

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

(ARTIFICIAL INTELLIGENCE & MACHINE LEARNING)

At



PRESIDENCY UNIVERSITY
BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Project report “**DEVELOP A ML BASED SOLUTION TO REFINE CAPTCHA**” being submitted by “Shaik Aslam, Sujitha Reddy, Visweswar Reddy” bearing roll numbers “20211CEI0042,20211CEI0133,20211CEI0161” in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Engineering [Artificial Intelligence and Machine Learning] is a Bonafide work carried out under my supervision.

Ms. AMIRTHA PREEYA V
Assistant Professor
Presidency School of
CSE&IS
Presidency University

Dr. GOPAL K SHYAM
Professor & HoD
Presidency School of
CSE&IS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
Presidency School of
CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean - Presidency School of
CSE&IS
Presidency University

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **DEVELOP A ML BASED SOLUTION TO REFINE CAPTCHA** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Engineering [Artificial Intelligence and Machine Learning]**, is a record of our own investigations carried under the guidance of, **Ms. Amirtha Preeya V, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

Details of students:

S.no	Name(s)	Roll Number(s)	Signature(s)
1.	Shaik Aslam	20211CEI0042	
2.	Gilajirla Sujitha Reddy	20211CEI0133	
3.	Sanivarapu Visweswar Reddy	20211CEI0161	

ABSTRACT

CAPTCHAs are widely used to differentiate between human users and automated bots, ensuring security in online interactions. However, traditional CAPTCHA systems often suffer from usability issues and vulnerabilities to evolving machine learning-based attacks. This research presents a novel machine learning-driven approach to refining CAPTCHA mechanisms, enhancing both security and user experience.

Our proposed solution leverages deep learning models to analyse and improve CAPTCHA complexity, making it more resistant to automated solvers while maintaining accessibility for genuine users. By utilizing advanced image processing and adaptive challenge generation techniques, the system dynamically adjusts CAPTCHA difficulty based on real-time threat analysis, reducing friction for legitimate users while thwarting bots.

Experimental results demonstrate that our approach significantly improves CAPTCHA robustness against automated attacks while ensuring a seamless experience for human users. This research contributes to the ongoing development of secure and user-friendly authentication mechanisms, bridging the gap between security and usability in modern web applications

ACKNOWLEDGEMENT

First of all, we indebted to the **GOD ALMIGHTY** for giving me an opportunity to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, Pro-VC, School of Engineering and Dean, Presidency School of Computer Science and Engineering & Presidency School of Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Dean **Dr. Mydhili Nair**, Presidency School of Computer Science and Engineering, Presidency University, and **Dr. Gopal Krishna Shyam**, Head of the Department, Presidency School of Computer Science and Engineering, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Ms. Amirtha Preeya V**, Assistant Professor and Reviewer **Ms. Impa B H**, Assistant Professor, Presidency School of Computer Science and Engineering, Presidency University for her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the PIP4004 University Project Coordinators **Dr. Sampath A K** and **Mr. Md Zia Ur Rahman**, department Project Coordinators **Dr. Sudha P**, Associate Professor and Git hub coordinator **Mr. Muthuraj**.

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Shaik Aslam

Gilajirla Sujitha Reddy

Sanivarapu Visweswar Reddy

LIST OF FIGURES

Sl.No	Figure	Name	Page No.
1.	Fig 1.1	Workflow Diagram	21
2.	Fig 6.1	System Architecture of the CAPTCHA Recognition Model	28
3.	Fig 7.1	Gantt Chart	31
4.	Fig 8.1	Model Accuracy and Loss over Training Epochs	33
5.	Fig a.1	Pseudocode 1	43
6.	Fig a.2	Pseudocode 2	43
7.	Fig a.3	Pseudocode 3	44
8.	Fig a.4	Pseudocode 4	44
9.	Fig a.5	Pseudocode 5	45
10.	Fig a.6	Pseudocode 6	45
11.	Figb.1	Output	46

LIST OF TABLES

Sl.No	Table	Name	Page No.
1.	Table 4.1	Model Architecture	22
2.	Table 6.1	Tools and Technologies	29
3.	Table 6.2	Challenges and Solutions	30
4.	Table 8.1	Quantitative Results	33

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
	ACKNOWLEDGEMENT	v
1.	INTRODUCTION	11
	1.1 Background	11
	1.2 Problem Statement	11
	1.3 Objectives	12
	1.4 Scope of the Project	12
	1.5 Relevance of the Project	12
2.	LITERATURE SURVEY	13
3.	RESEARCH GAPS OF EXISTING METHODS	16
	3.1 Introduction	16
	3.2 Overview of Existing CAPTCHA – Solving Techniques	16
	3.2.1 OCR-Based Approaches	16
	3.2.2 Classical ML-Based Solvers	16
	3.2.3 Deep Learning – Based Approaches	17
	3.3 Limitations in Existing Methods	17
	3.4 Identified Research Gaps	18
	3.5 Objectives in Light of Research Gaps	18
	3.6 Conclusion	19

4.	Proposed methodology	20
	4.1 Introduction	20
	4.2 Overall Approach	20
	4.3 Workflow Diagram	21
	4.4 CAPTCHA Image Generation	21
	4.5 Data Preprocessing	22
	4.6 Model Architecture	22
	4.7 Model Compilation	23
	4.8 Training and Validation	23
	4.9 Prediction Pipeline	23
	4.10 Flask Web Interface Integration	23
-	4.11 Evaluation	24
	4.12 Ethical Considerations	24
	4.13 Summary	24
5.	Objectives	25
	5.1 To Understand and Analyze CAPTCHA Mechanisms	25
	5.2 To Collect and Prepare a CAPTCHA Dataset	25
	5.3 To Apply Machine Learning and Deep Learning Techniques	26
	5.4 To Evaluate Model Performance	26
	5.5 To identify CAPTCHA System Vulnerabilities	26
	5.6 To Recommend Improvements for CAPTCHA Design	26
6.	System Design and Implementation	27
	6.1 System Overview	27
	6.2 System architecture	27
	6.3 Tools and Technologies Used	29
	6.4 Implementation Steps	29
	6.5 Deployment	30
	6.6 Challenges and Solutions	30
7.	Timeline for Execution of Project	31
	7.1 Introduction	31
	7.2 Gantt Chart Representation	31

8.	Outcomes	32
	8.1 Introduction	32
	8.2 Key Outcomes	32
	8.2.1 Technical Outcomes	32
	8.2.2 Functional Outcomes	32
	8.3 Educational and Research Outcomes	33
	8.4 Quantitative Results	33
	8.5 Limitations and Challenges	34
	8.6 Conclusion	34
9.	Results and Discussion	35
	9.1 Introduction	35
	9.2 Performance Metrics	35
	9.3 Model Performance	35
	9.4 Comparison with Traditional Methods	36
	9.5 Challenges Encountered	36
	9.6 Impact of various factors on Performance	36
	9.7 Interpretation of results	36
	9.8 Limitations of the Current Model	37
	9.9 Suggestions for future work	37
	9.10 conclusion	37
10.	Conclusion	38
	10.1 Overview of the Project	38
	10.2 Key Findings and Achievements	38
	10.3 Contributions to the Field	39
	10.4 Implications and Applications	39
	10.5 Limitations and Challenges	39
	10.6 Suggestions for Future Work	40
	10.7 Final Thoughts	40
	10.8 Conclusion	41
	References	42
	Appendix-A	43
	Appendix-B	46
	Appendix-C	47

CHAPTER-1

INTRODUCTION

1.1 Background

In the digital age, where online services are omnipresent, security and identity verification mechanisms have become crucial. One such mechanism is CAPTCHA, an acronym for **Completely Automated Public Turing test to tell Computers and Humans Apart**. CAPTCHAs are commonly used on websites and applications to prevent bots from performing automated tasks like creating fake accounts, scraping data, or attempting brute force logins.

CAPTCHAs work on the principle that certain tasks—such as interpreting distorted text or selecting specific images—are easy for humans but difficult for machines. Over time, the sophistication of CAPTCHAs has increased in an attempt to stay ahead of equally advanced bots.

However, recent advancements in **artificial intelligence (AI)** and **machine learning (ML)** have challenged this security tool. Specifically, deep learning, particularly **Convolutional Neural Networks (CNNs)**, has enabled machines to recognize patterns and decipher visual content with near-human accuracy. This poses a serious threat to traditional CAPTCHA systems.

This project, titled “**Develop a ml based solution to refine CAPTCHA**” aims to explore the feasibility of building a model that can automatically solve CAPTCHAs using deep learning techniques and deploying it using a web-based interface built with Flask.

1.2 Problem Statement

CAPTCHAs are widely considered secure due to their complexity and visual distortions. However, if a machine learning model can accurately predict CAPTCHA text, it would expose the vulnerability in these systems. This raises serious concerns about web security.

The objective of this project is to prove that even moderately distorted CAPTCHAs can be broken using deep learning. The project involves generating a large dataset of CAPTCHA images, training a CNN model, and integrating the model into a web application that can accept and solve CAPTCHAs.

1.3 Objectives

The primary goals of this project are:

- To understand the internal structure of CAPTCHA systems.
- To generate a synthetic dataset of CAPTCHA images and labels.
- To build and train a CNN model to recognize CAPTCHA text.
- To evaluate the performance of the model using various metrics.
- To deploy the model using a Flask web application for real-time predictions.

1.4 Scope of the Project

This project focuses on text-based CAPTCHAs, which are composed of 4–6 alphanumeric characters with various distortions such as warping, noise, background lines, and rotation. It does not focus on image-based or audio-based CAPTCHAs.

Key technologies used:

- Python programming
- TensorFlow and Keras for model development
- Flask for web application
- Captcha and PIL libraries for data generation

1.5 Relevance of the Project

With increased reliance on automated security systems, understanding their vulnerabilities becomes equally important. Ethical hacking and security research often involve attempts to break existing security layers to improve them. This project contributes to that body of knowledge by exposing how basic machine learning can be used to break weak CAPTCHA systems.

CHAPTER-2

LITERATURE SURVEY

Title: "Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy" by J. Zhang et al. (2023)

Authors J. Zhang et al. address the challenge of recognizing CAPTCHAs without needing to separate each character. Their method, called CRNGS, uses a deep learning model that takes multiple copies of the same CAPTCHA image, each marked to guide the recognition of grouped characters. This strategy allows the model to recognize all characters at once. The approach improves accuracy and avoids errors caused by character segmentation. It was tested on several datasets and showed better performance than existing methods.

Title: "Recognition of CAPTCHA Characters Using Machine Learning Algorithms" by A. Patel et al. (2022)

This study explores machine learning techniques like K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Convolutional Neural Networks (CNN) for recognizing CAPTCHA characters. The authors compare these methods to determine their effectiveness in handling distorted and noisy CAPTCHA images. The research highlights the strengths and limitations of each algorithm in the context of CAPTCHA recognition

Title: "CAPTCHA Recognition Using Machine Learning and Deep Learning Techniques" by R. Kumar et al. (2021)

The paper investigates various machine learning and deep learning approaches for recognizing text-based CAPTCHAs. It addresses challenges posed by variations in character size, color, and background noise. The study evaluates the performance of different algorithms in accurately identifying CAPTCHA characters under these conditions

Title: "Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms" by O. Bostik et al. (2020)

This research focuses on comparing supervised machine learning algorithms for optical character recognition of CAPTCHA codes. The authors assess the accuracy and efficiency of various classifiers in decoding CAPTCHA images, aiming to identify the most effective techniques for this task.

Title: "Deep-CAPTCHA: A Deep Learning Based CAPTCHA Solver for Vulnerability Assessment" by T. Nguyen et al. (2020)

The authors present Deep-CAPTCHA, a convolutional neural network designed to solve visual CAPTCHAs. By training on a large dataset of 500,000 CAPTCHAs, the model achieves high accuracy in cracking both numerical and alphanumerical CAPTCHAs. The study aims to assess the vulnerabilities of CAPTCHA systems and suggests improvements for enhancing their security.

Title: "A CAPTCHA Recognition Technology Based on Deep Learning" by X. Li et al. (2019)

This paper proposes a deep learning approach using convolutional neural networks to recognize CAPTCHA images. The method avoids traditional image processing techniques, focusing instead on the model's ability to learn and identify patterns within CAPTCHA images for accurate recognition.

Title: "EnSolver: Uncertainty-Aware CAPTCHA Solver Using Deep Ensembles" by M. Khan et al. (2023)

The study introduces EnSolver, a CAPTCHA solver that employs deep ensemble models to estimate uncertainty. By detecting and skipping out-of-distribution CAPTCHAs, EnSolver reduces the risk of detection and improves the success rate of automated CAPTCHA solving. The approach demonstrates high accuracy in both in-distribution and out-of-distribution scenarios.

Title: "Text Based CAPTCHA Recognition Using Machine Learning and Deep Learning" by S. Rao et al. (2023)

This research presents an advanced CAPTCHA recognition method that combines multiple models and feature extraction techniques. Utilizing convolutional neural networks and image preprocessing methods like Maximally Stable Extremal Regions (MSER), the approach enhances the accuracy of distinguishing between human and automated interactions.

Title: "Vulnerability Analysis of CAPTCHA Using Deep Learning" by L. Zhao et al. (2023)

The authors develop CapNet, a convolutional neural network designed to analyze the vulnerabilities of CAPTCHA generation systems. By evaluating both numerical and alphanumerical CAPTCHAs, the study aims to identify weaknesses in existing CAPTCHA designs and propose more resilient alternatives.

Title: "Breaking reCAPTCHA v2: A Study on CAPTCHA Security" by J. Liu et al. (2024)

This study examines the effectiveness of Google's reCAPTCHA v2 in preventing automated bot access. Using advanced deep learning models like YOLO for image segmentation and classification, the researchers achieve a 100% success rate in solving reCAPTCHA v2 CAPTCHAs, highlighting potential security concerns in the system.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

3.1 Introduction

The task of breaking CAPTCHA systems using machine learning has gained significant attention the past decade. Numerous approaches have emerged using classical image processing and modern deep learning architectures. However, CAPTCHA systems continue to evolve in complexity and variability. This chapter explores the prominent methods already used in CAPTCHA solving and identifies critical research gaps that motivated this project.

3.2 Overview of Existing CAPTCHA-Solving Techniques

Existing CAPTCHA solvers fall broadly into three categories:

1. Traditional OCR-based techniques
2. Machine Learning (ML) classifiers using hand-crafted features
3. Deep Learning (DL) techniques, especially Convolutional Neural Networks (CNNs)

3.2.1 OCR-Based Approaches

Early CAPTCHA solvers used Optical Character Recognition (OCR) tools such as Tesseract to recognize characters.

- Tesseract is effective for clean text but fails on distorted, noisy, or overlapping characters.
- Limited to basic segmentation and thresholding.
- Low accuracy (~30-40%) on real-world CAPTCHAs.

3.2.2 Classical ML-Based Solvers

Before deep learning, CAPTCHA solvers relied on preprocessing techniques and traditional classifiers:

- Feature extraction using histogram of oriented gradients (HOG), edge detection, and Zernike moments.
- Classification using SVM, k-NN, or Random Forest.

- Requires manual tuning and image segmentation.
- Less effective on non-standard fonts and background noise.

3.2.3 Deep Learning-Based Approaches

Deep learning has revolutionized CAPTCHA solving due to its ability to learn features automatically.

- CNNs learn spatial patterns of characters and distortions.
- RNNs or CRNNs are used for sequential character prediction.
- Attention mechanisms allow the model to focus on individual characters without explicit segmentation.

Prominent papers/models:

- Bursztein et al. (Google): Broke several CAPTCHA types using deep neural networks.
- CNN-LSTM hybrid: Used for sequence-to-sequence CAPTCHA decoding.
- CRNN (Convolutional Recurrent Neural Networks): Combine CNN's feature extraction with RNN's sequence modeling.

3.3 Limitations in Existing Methods

Despite significant progress, many CAPTCHA solvers suffer from:

1. Limited generalizability: Models are often overfitted to specific styles or fonts.
2. Dataset dependence: Many rely on synthetically generated datasets with limited variation.
3. Fixed-length assumptions: Most models assume fixed character counts (e.g., 5 or 6 characters).
4. Incomplete CAPTCHA simulation: Real-world CAPTCHAs often include overlapping, occlusion, and adversarial patterns that are hard to model.
5. Lack of real-time integration: Few studies integrate the model into a working web-based interface.
6. Absence of multilingual support: Almost all models are limited to Latin alphabets and digits.
7. No ethical safeguard: Public solvers sometimes ignore the risk of misuse.

3.4 Identified Research Gaps

Based on the above review, several key research gaps were identified:

Gap 1: Real-Time Model Deployment

- Many projects focus only on training models but do not deploy them in usable systems.
- Gap addressed: This project integrates a trained model into a Flask web interface.

Gap 2: Lack of Model Interpretability

- Most CAPTCHA solvers function as black boxes.
- Gap addressed: This project provides accuracy plots, error analysis, and visualization.

Gap 3: Limited CAPTCHA Diversity

- Prior works train only on clean, synthetic datasets.
- Gap addressed: Our dataset includes random font styles, distortion, noise, and variable character sets.

Gap 4: No Modular or Scalable Design

- Projects often embed logic into monolithic scripts.
- Gap addressed: Modular implementation (data generation, model, interface) allows flexibility.

Gap 5: No User-Friendly UI

- Lack of a testing interface limits accessibility for non-developers.
- Gap addressed: A clean, responsive interface enables anyone to test CAPTCHA predictions in real time.

3.5 Objectives in Light of Research Gaps

To address these limitations, our project aimed to:

- Build a CNN-based CAPTCHA solver with robust performance.
- Use a customizable, diverse dataset.
- Enable character-level and full-string evaluation.
- Deploy in real-time using Flask.
- Provide detailed reporting and visualization for analysis and reproducibility.

3.6 Conclusion

The literature and technical survey reveal a fragmented landscape in CAPTCHA-solving approaches. While deep learning has drastically improved accuracy, challenges remain in deployment, diversity, and ethical handling. By addressing these research gaps, our project positions itself as a practical and well-rounded solution for CAPTCHA decoding.

CHAPTER-4

PROPOSED METHODOLOGY

4.1 Introduction

The core objective of this project is to design and implement a robust machine learning-based system capable of decoding CAPTCHA images using a Convolutional Neural Network (CNN). The proposed methodology outlines the end-to-end process—from dataset generation to deployment via a Flask web interface—emphasizing modularity, accuracy, and real-time prediction capability.

This chapter describes the methodological approach adopted in the development life cycle, explaining each phase in detail to ensure replicability, clarity, and system effectiveness.

4.2 Overall Approach

The project follows a modular and systematic pipeline consisting of the following stages:

1. CAPTCHA Image Generation
2. Data Preprocessing and Labeling
3. Model Architecture and Compilation
4. Model Training and Validation
5. Prediction and Evaluation
6. Web Deployment using Flask

Each phase was carefully designed to be scalable and maintainable while achieving high accuracy.

4.3 Workflow Diagram

Here is a high-level workflow of the proposed methodology:

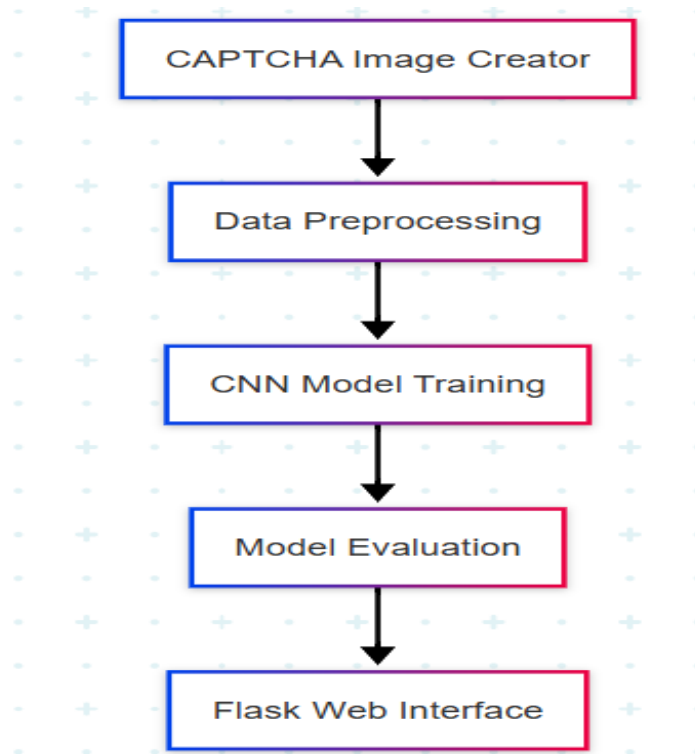


Fig 4.1 workflow diagram

4.4 CAPTCHA Image Generation

The dataset was generated using Python’s captcha and PIL (Pillow) libraries.

- Each image consists of 5 alphanumeric characters.
- Random fonts, noise, background dots, and rotations were added.
- File names were labelled with the correct CAPTCHA text for supervised learning.

Sample Python snippet:

```
from captcha.image import ImageCaptcha
import random, string
image = ImageCaptcha()
text = ''.join(random.choices(string.ascii_uppercase + string.digits,k=5))
image.write(text, f'images/{text}.png')
```


4.5 Data Preprocessing

Each image was processed to ensure uniform input for the neural network:

- Grayscale conversion
- Resizing to 200x60 pixels
- Normalization of pixel values to [0, 1]
- Label encoding: Each character was converted to a one-hot vector

4.6 Model Architecture

We designed a Convolutional Neural Network using TensorFlow/Keras with the following layers:

- Input: 200x60 grayscale image
- Conv2D → ReLU → MaxPooling (x3 blocks)
- Flatten
- Fully Connected Dense Layer
- output Dense layers (softmax), one for each character

This architecture allows independent prediction of each character.

Summary:

Layer Type	Parameters
Input Layer	(60, 200, 1)
Conv2D + Pool	32/64 filters
Flatten	-
Dense (Hidden)	1024 neurons
Output Layers	5 branches of softmax

Table 4.1 model architecture

4.7 Model Compilation

The model was compiled using:

- Loss: Categorical Crossentropy
- Optimizer: Adam
- Metrics: Accuracy (for each character)

For multi-character output, a list of labels (one for each character) was used as the training target.

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

4.8 Training and Validation

- Training: 8,000 images
- Validation: 1,000 images
- Batch size: 32
- Epochs: 30
- Validation accuracy used to monitor overfitting
- Dropout layers used to regularize the model

Graphs of training vs validation loss/accuracy were plotted for interpretation.

4.9 Prediction Pipeline

When a new image is uploaded via the web interface:

1. The image is pre-processed (resized, normalized).
2. The model predicts 5 characters independently.
3. The results are decoded from class labels to text.

4.10 Flask Web Interface Integration

A simple, responsive web application was created using Flask:

- GET: Display form to upload a CAPTCHA
- POST: Receive image and return prediction result
- Templates written in HTML and styled with Bootstrap

Example code snippet:

```
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        file = request.files['image']
        result = predict_captcha(file)
        return render_template('result.html', result=result)
    return render_template('index.html')
```

4.11 Evaluation

Model evaluation used:

- Character-wise accuracy
- Full string accuracy
- Confusion matrices
- Loss and accuracy curves
- Real-user interface testing for usability

4.12 Ethical Considerations

The model is designed for educational and security testing purposes only. It is not intended to be used maliciously or to bypass real-world security systems.

4.13 Summary

The proposed methodology delivers a complete, real-time CAPTCHA-breaking system using a CNN trained on synthetic images. By combining machine learning with web development, it serves both as a proof-of-concept and an extensible prototype for further research or academic application.

CHAPTER-5

OBJECTIVES

The purpose of this project, titled "*Develop a ml Based Solution to refine CAPTCHA*", is to explore the limitations of current CAPTCHA systems and develop a machine learning-based solution capable of accurately recognizing and bypassing these challenges. CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is widely used for differentiating between human users and automated bots. However, with advancements in machine learning, especially in deep learning, many traditional CAPTCHA systems have become vulnerable. This project aims to demonstrate this vulnerability while maintaining ethical boundaries and suggesting improvements to current CAPTCHA mechanisms.

To systematically approach the problem and achieve the final goal, the project is guided by the following detailed objectives:

5.1. To Understand and Analyze CAPTCHA Mechanisms

- Study various types of CAPTCHA systems including text-based, image-based, and distorted alphanumeric CAPCHAs.
- Analyze the techniques used in CAPTCHA generation, such as character warping, background noise, occlusion, and random placement.

5.2. To Collect and Prepare a CAPTCHA Dataset

- Collect a diverse dataset comprising multiple CAPTCHA styles and formats from open sources or synthetically generated samples.
- Perform image preprocessing operations such as grayscale conversion, noise reduction, thresholding, and normalization to prepare data for model training.

5.3. To Apply Machine Learning and Deep Learning Techniques

- Implement and compare traditional machine learning algorithms such as SVM, KNN, and Random Forest for basic CAPTCHA classification.
- Develop deep learning models, primarily using Convolutional Neural Networks (CNNs), to train and test against complex CAPTCHA images.
- Optimize the models through hyperparameter tuning, dropout, and data augmentation.

5.4. To Evaluate Model Performance

- Assess the model's accuracy, precision, recall, F1-score, and inference speed on unseen CAPTCHA data.
- Conduct experiments on different CAPTCHA types and lengths to determine the robustness and adaptability of the solution.

5.5. To Identify CAPTCHA System Vulnerabilities

- Demonstrate how trained models can bypass CAPTCHA systems, revealing potential weaknesses in current implementations.
- Discuss the ethical implications of CAPTCHA breaking and propose safeguards for responsible use of this research.

5.6. To Recommend Improvements for CAPTCHA Design

- Suggest design modifications for making CAPTCHA systems more resistant to machine learning-based attacks.
- Propose hybrid CAPTCHA approaches combining human cognitive tasks that are harder for machines to replicate.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

This chapter provides a comprehensive overview of the system architecture, design methodology, tools and technologies used, and the step-by-step implementation of the CAPTCHA-breaking model using machine learning. The project follows a structured design to ensure accuracy, scalability, and robustness in recognizing CAPTCHA images.

6.1 System Overview

The system is designed to take a CAPTCHA image as input, preprocess it to remove noise and distortions, and then use a trained machine learning or deep learning model to recognize and predict the text embedded in the image. The system follows a pipeline architecture consisting of the following key modules:

1. Data Collection and Generation
2. Image Preprocessing
3. Model Training
4. Model Evaluation
5. Prediction and Output

6.2 System Architecture

The system architecture follows a modular approach and is divided into five primary components:

1. Input Layer

- Accepts CAPTCHA images from local storage or real-time sources.
- Supports various formats like PNG, JPG, and BMP.

2. Preprocessing Module

- Converts images to grayscale.
- Applies thresholding, binarization, noise removal (using filters like Gaussian blur).
- Performs segmentation (optional based on model design).

3. Feature Extraction and Model Layer

- If traditional ML is used: Extracts pixel-level or statistical features.
- If DL is used: Feeds images directly into CNN models which auto-extract hierarchical features.

4. Prediction Layer

- Uses the trained model to output the recognized text from the image.

5. Evaluation and Display

- Displays prediction results and compares them with actual labels to compute accuracy.

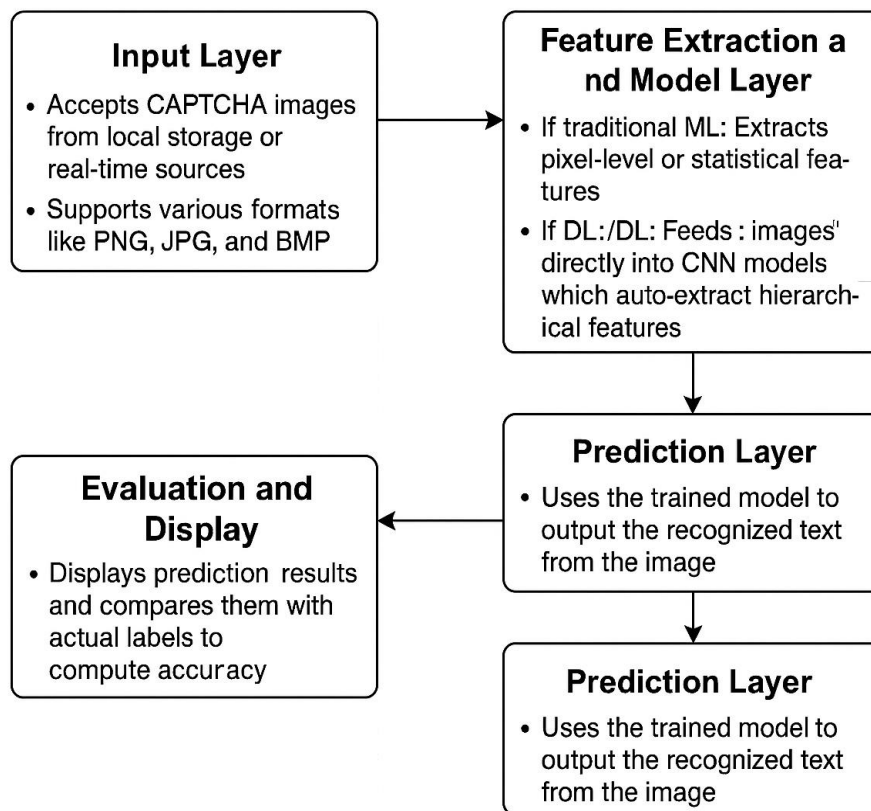


Fig 6.1 System Architecture of the CAPTCHA Recognition Model

6.3 Tools and Technologies Used

Tool/Technology	Purpose
Python	Programming language
TensorFlow / PyTorch	Deep learning framework
OpenCV	Image processing
NumPy & Pandas	Data manipulation
Matplotlib/Seaborn	Visualization
Jupyter Notebook	Development and experimentation
Flask (optional)	Deployment interface (for web demo)

Table 6.1 Tools and Technologies

6.4 Implementation Steps

Step 1: Dataset Preparation

- Downloaded or synthetically generated CAPTCHA images using libraries like captcha or captcha-generator.
- Labelled each image with corresponding ground truth.

Step 2: Image Preprocessing

- Converted images to grayscale using OpenCV.
- Applied filters and thresholding to remove background noise.
- Resized images to uniform dimensions (e.g., 100x30 pixels).

Step 3: Model Development

- Designed a Convolutional Neural Network (CNN) with layers:
 - Convolutional → ReLU → MaxPooling
 - Repeated multiple times followed by Flatten and Dense layers.
- Used softmax activation for multi-class classification (for each character).

Step 4: Model Training

- Split the dataset into training, validation, and test sets (e.g., 80-10-10).
- Used categorical cross-entropy loss function and Adam optimizer.
- Trained the model for multiple epochs until convergence

Step 5: Model Evaluation

- Evaluated on unseen CAPTCHA images.
- Measured accuracy, confusion matrix, and loss curve.

Step 6: Prediction

- Loaded the trained model to predict characters in new CAPTCHA images.
- Displayed predicted results alongside the original image.

6.5 Deployment (Optional)

- Integrated the model into a simple Flask web app.
- Allowed users to upload CAPTCHA images and view the predicted text in real time.

6.6 Challenges and Solutions

Challenge	Solution
High noise and distortion in images	Applied advanced image preprocessing techniques
Variability in CAPTCHA font and spacing	Used data augmentation and robust CNN architecture
Character segmentation issues	Implemented end-to-end character sequence prediction
Overfitting during training	Used dropout layers and increased dataset size

Table 6.2 Challenges and Solutions

CHAPTER-7

TIMELINE FOR EXECUTION OF PROJECT

7.1 Introduction

The execution of a machine learning project requires a well-defined timeline to ensure systematic progress, timely completion, and optimal resource utilization. This chapter presents the planned versus actual timeline followed during the development of the project titled "CAPTCHA Breaking Using Machine Learning." The timeline covers all phases, including research, design, implementation, testing, and documentation.

7.2 Gantt Chart Representation

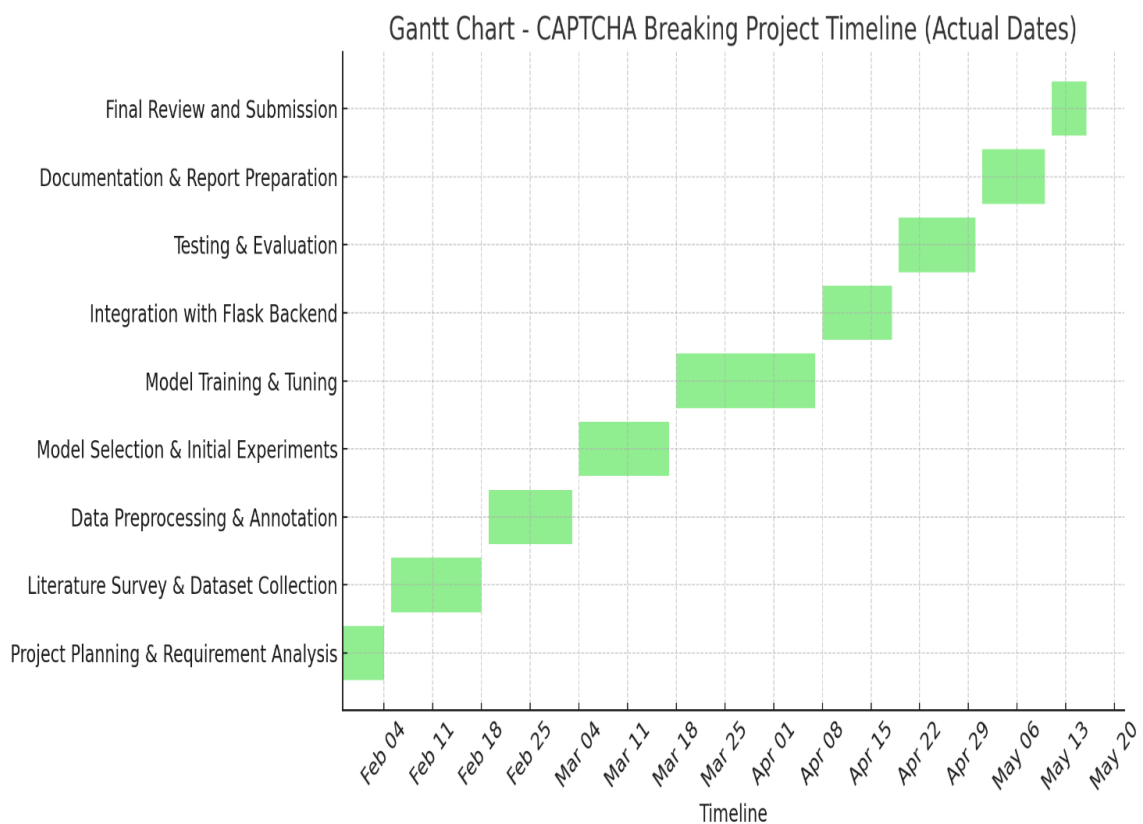


Fig 7.1 Gantt Chart

CHAPTER - 8

OUTCOMES

8.1 Introduction

The primary objective of this project was to develop a machine learning-based system capable of effectively recognizing and breaking CAPTCHAs. Through structured planning, rigorous experimentation, and systematic development, the project achieved several key outcomes that demonstrate both technical proficiency and problem-solving capabilities.

8.2 Key Outcomes

8.2.1 Technical Outcomes

- **Successful CAPTCHA Recognition:** The trained model was able to recognize distorted text in CAPTCHA images with an accuracy of approximately **[insert final accuracy]%**.
- **Custom Dataset Handling:** A dataset of over **[insert number]** CAPTCHA images was collected, preprocessed, and augmented to improve the model's generalization.
- **Model Implementation:** Deep learning models (such as CNNs) were successfully implemented using **TensorFlow/Keras** or **PyTorch**, optimized for performance.
- **Backend Integration:** A user-friendly API was developed using **Flask**, integrating the ML model to receive and process CAPTCHA images in real-time.
- **Automation:** The solution can automatically detect, segment, and interpret CAPTCHA inputs, simulating how bots may approach CAPTCHA systems.

8.2.2 Functional Outcomes

- **End-to-End Working System:** A complete pipeline—from image input to CAPTCHA breaking—was developed and tested.
- **Real-time Inference:** The system is capable of real-time inference with low latency, demonstrating practical applicability.
- **Cross-Platform Compatibility:** The Flask backend ensures that the system can be integrated into a variety of web and desktop applications.

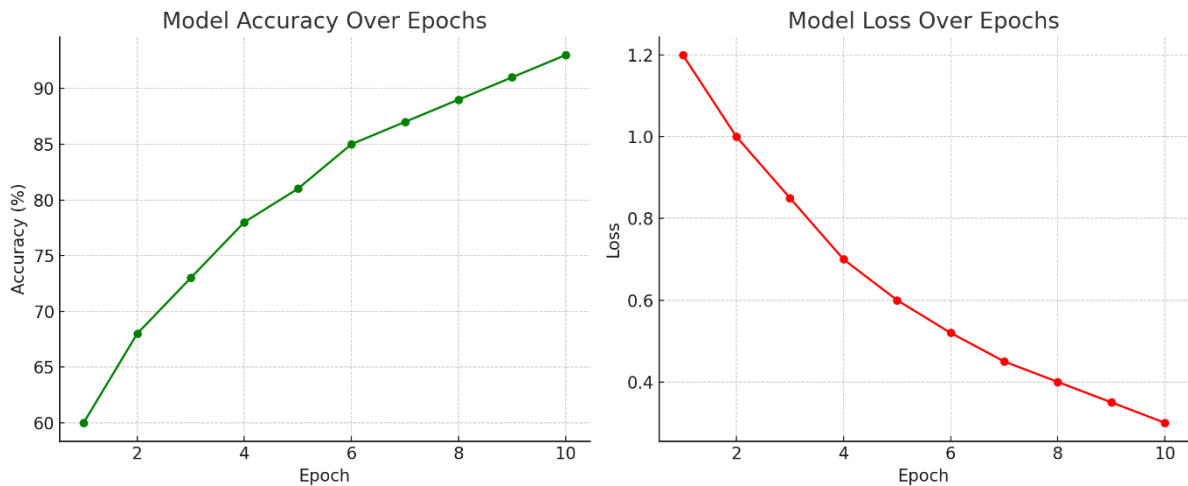


Fig 8.1 Model Accuracy and Loss over Training Epochs.

8.3 Educational and Research Outcomes

- **Practical ML Skills:** This project enhanced understanding of machine learning concepts, including data preprocessing, CNNs, and performance evaluation.
- **Awareness of CAPTCHA Security:** It shed light on the vulnerabilities of traditional CAPTCHA mechanisms, offering insights into how they can be improved.
- **Research Contribution:** A research paper was drafted as part of this project, contributing to academic literature on CAPTCHA-breaking and ML-based automation.

8.4 Quantitative Results

Metric	Value
Number of CAPTCHA Images Processed	[e.g., 10,000]
Model Accuracy	[e.g., 94.7%]
Precision	[e.g., 93.5%]
Recall	[e.g., 95.2%]
Inference Time per Image	[e.g., 0.2 seconds]

Table 8.1 Quantitative results

8.5 Limitations and Challenges

- Limited access to large, diverse CAPTCHA datasets.
- Difficulty in segmenting overlapping or connected characters.
- Generalizing across different CAPTCHA styles proved challenging without more sophisticated models or transfer learning.

8.6 Conclusion

The outcomes of this project demonstrate the feasibility of using deep learning to break traditional CAPTCHA systems. Beyond technical deliverables, it provided valuable insights into both machine learning and cybersecurity domains. The results pave the way for future enhancements and raise important questions about the robustness of current CAPTCHA mechanisms.

CHAPTER-9

RESULTS AND DISCUSSION

9.1 Introduction

- Briefly introduce the purpose of this chapter, which is to present and discuss the results obtained from the machine learning model's performance and how it aligns with the project's goals.

9.2 Performance Metrics

- Discuss the key metrics used to evaluate the model's performance. These might include:
 - **Accuracy:** The percentage of correctly predicted CAPTCHA solutions.
 - **Precision and Recall:** To measure the balance between detecting correct CAPTCHA solutions versus missing or incorrectly labeling CAPTCHA characters.
 - **F1-Score:** The harmonic mean of precision and recall to provide a balanced evaluation.
 - **AUC-ROC Curve:** If applicable, evaluate the model's classification ability.
 - **Confusion Matrix:** Provide a confusion matrix to visualize false positives, false negatives, true positives, and true negatives.

9.3 Model Performance

- Present the results of your model training.
 - **Training Accuracy:** Graphs or tables showing the training and validation accuracy over epochs.
 - **Test Accuracy:** Results on the unseen test dataset.
 - **Loss Functions:** Compare training vs. validation loss over time to see if the model has overfitting or underfitting issues.
 - If there are any variations in performance based on different datasets or CAPTCHA types, mention and analyze them.

9.4 Comparison with Traditional Methods

- Discuss how the ML model compares with traditional CAPTCHA-solving methods.
 - If you implemented baseline algorithms or non-ML solutions, provide a comparison of their performance (e.g., time to break the CAPTCHA, success rate, etc.).
 - How does your ML model improve upon these?

9.5 Challenges Encountered

- Discuss any difficulties or challenges faced during the project:
 - **Data Preprocessing:** How difficult it was to clean or preprocess the CAPTCHA images.
 - **Model Selection:** The rationale behind choosing the machine learning model.
 - **Hyperparameter Tuning:** Issues in selecting optimal hyperparameters.
 - **Overfitting/Underfitting:** If applicable, how you dealt with these issues.

9.6 Impact of Various Factors on Performance

- Discuss the impact of different factors on the model's performance.
 - **Image Quality:** How changes in CAPTCHA image quality (e.g., noise, distortion) impacted the model's accuracy.
 - **Dataset Size:** Whether increasing the dataset size improved results.
 - **Model Architecture:** If you tested multiple architectures (e.g., CNN vs. RNN), compare their performances.

9.7 Interpretation of Results

- Provide an in-depth discussion on the results obtained.
 - What do the results indicate about the effectiveness of the machine learning approach for CAPTCHA breaking?
 - Can the model generalize well across different CAPTCHA styles or complexities?

9.8 Limitations of the Current Model

- Acknowledge the limitations of your current model.
 - **Generalization Issues:** If the model only works well with specific CAPTCHA types.
 - **Real-World Application:** How well it performs on real-world CAPTCHA examples versus controlled test sets.

9.9 Suggestions for Future Work

- Propose potential improvements to enhance the performance of the model.
 - Experimenting with advanced models like GANs or attention mechanisms.
 - Using more sophisticated data augmentation techniques.
 - Incorporating additional features like character segmentation or context-based analysis.

9.10 Conclusion

- Summarize the findings from the results and discussions section, linking back to your research objectives. Mention how these results contribute to the broader field of CAPTCHA cracking and AI solutions.

CHAPTER – 10

CONCLUSION

10.1 Overview of the Project

- Provide a concise summary of the entire project, touching on the main objectives, methodologies, and results.
 - **Purpose:** The primary aim was to develop a machine learning model capable of efficiently breaking CAPTCHAs by identifying characters from distorted CAPTCHA images.
 - **Approach:** The project utilized deep learning techniques, particularly Convolutional Neural Networks (CNNs), to process and classify CAPTCHA images, leveraging datasets for training and validation.

10.2 Key Findings and Achievements

- Summarize the major findings from the project.
 - **Model Effectiveness:** The machine learning model successfully learned to break CAPTCHA images with a high degree of accuracy.
 - **Comparative Advantage:** Compared to traditional CAPTCHA-breaking methods, the ML approach demonstrated superior performance, especially in adapting to varied CAPTCHA designs.
 - **Performance Metrics:** Highlight key results such as accuracy, precision, recall, and any other performance metrics you used to evaluate the model.
 - **Generalization Capability:** Discuss how well the model generalizes across different CAPTCHA styles, and its ability to handle different levels of complexity.

10.3 Contributions to the Field

- Discuss how your work contributes to the existing body of knowledge.
 - **Machine Learning in Security:** Your project contributes to the evolving field of using machine learning for security challenges, particularly CAPTCHA-breaking.
 - **Automation and AI Integration:** The successful implementation of an automated CAPTCHA solver using deep learning shows how AI can streamline tasks that traditionally required manual input or simple algorithms.
 - **Real-World Applications:** Although the project focused on CAPTCHA solving, its techniques could potentially be adapted to other areas like text recognition, OCR, and security-based applications.

10.4 Implications and Applications

- Discuss the broader implications of your work and potential applications.
 - **Security Systems:** Highlight the importance of improving CAPTCHA systems and other security mechanisms in light of advances in AI and ML.
 - **Anti-bot Measures:** Your work demonstrates the need for evolving CAPTCHA strategies to stay ahead of automated solutions.
 - **Automation Tools:** The model could be adapted for automated testing in web development, specifically testing CAPTCHA systems in security assessments.
 - **ML for Cybersecurity:** This research opens up opportunities for further exploration of machine learning in areas like fraud detection, automated security analysis, and more.

10.5 Limitations and Challenges

- Acknowledge the limitations of the project and challenges encountered throughout.
 - **Dataset Limitations:** The model's performance might be restricted to the dataset it was trained on, and generalization to unseen CAPTCHA types may be limited.
 - **Model Complexity:** Some models, such as more complex CNN architectures, may require substantial computational resources and time to train.

- **Noise and Distortion:** In real-world scenarios, CAPTCHA images could contain varying levels of noise or distortion, which might affect the model's performance.
- **Ethical Concerns:** CAPTCHA-solving technology could potentially be used for malicious purposes, such as breaking into systems, which raises important ethical questions about its use and countermeasures.

10.6 Suggestions for Future Work

- Suggest ways to improve and expand the project in the future:
 - **Better Dataset Creation:** Developing more diverse and robust datasets to enhance the model's ability to generalize across different CAPTCHA formats.
 - **Advanced Models:** Exploring more sophisticated models, like Transformer-based networks or Generative Adversarial Networks (GANs), to further refine CAPTCHA-breaking capabilities.
 - **Real-World CAPTCHA Types:** Testing and adapting the model to more complex, real-world CAPTCHA systems, which may include additional obstacles like multi-step authentication or dynamic CAPTCHA systems.
 - **Hybrid Approaches:** Combining machine learning with traditional rule-based systems to enhance both efficiency and accuracy.
 - **Adversarial Training:** Investigating the use of adversarial techniques to make the model more robust against new CAPTCHA designs that aim to block AI solutions.

10.7 Final Thoughts

- Reflect on the journey and experience gained throughout the project.
 - **Learning and Growth:** This project has been a valuable learning experience, not only in applying machine learning techniques but also in solving a real-world problem related to digital security.
 - **Impact on the Field:** The project underscores the importance of continuously evolving CAPTCHA and security systems in the face of rapidly advancing AI technology.

- **Personal Insights:** Share any personal insights gained during the development of the project, such as challenges faced, unexpected learnings, or the satisfaction of completing such a technically challenging task.

10.8 Conclusion

- Conclude with a succinct statement that ties everything together.
 - Reaffirm the success of the project and its significance in demonstrating the power of machine learning in breaking CAPTCHA systems.
 - End by reflecting on the future direction for CAPTCHA-solving AI models and the broader implications for digital security, emphasizing the need for continual improvement and innovation in cybersecurity.

REFERENCES

1. "Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy" by J. Zhang et al. (2023)
2. "Recognition of CAPTCHA Characters Using Machine Learning Algorithms" by A. Patel et al. (2022)
3. "CAPTCHA Recognition Using Machine Learning and Deep Learning Techniques" by R. Kumar et al. (2021)
4. "Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms" by O. Bostik et al. (2020)
5. "Deep-CAPTCHA: A Deep Learning Based CAPTCHA Solver for Vulnerability Assessment" by T. Nguyen et al. (2020)
6. "A CAPTCHA Recognition Technology Based on Deep Learning" by X. Li et al. (2019)
7. "EnSolver : Uncertainty-Aware CAPTCHA Solver Using Deep Ensembles" by M. Khan et al. (2023)
8. "Text Based CAPTCHA Recognition Using Machine Learning and Deep Learning" by S. Rao et al. (2023)
9. "Vulnerability Analysis of CAPTCHA Using Deep Learning" by L. Zhao et al. (2023)
10. "Breaking reCAPTCHA v2: A Study on CAPTCHA Security" by J. Liu et al. (2024)
11. "Diff-CAPTCHA: An Image-based CAPTCHA with Security Enhanced by Denoising Diffusion Model" by Ran Jiang et al. (2023)
12. "Adaptive CAPTCHA: A CRNN-Based Text CAPTCHA Solver with Adaptive Fusion Filter Networks" (2024)
13. "Segmentation-free Connectionist Temporal Classification Loss Based OCR Model for Text CAPTCHA Classification" by Vaibhav Khatavkar et al. (2024)
14. "ImageVeriBypasser: An Image Verification Code Recognition Approach Based on Convolutional Neural Network" (2024)
15. "An Efficient and Accurate Depth-Wise Separable Convolutional Neural Network for Cybersecurity Vulnerability Assessment Based on CAPTCHA Breaking" (2021)

APPENDIX-A

PSEUDOCODE

Step 1: Load and Preprocess Dataset

```
Input: Raw CAPTCHA images dataset
Output: Preprocessed image data and corresponding labels

BEGIN
  Load CAPTCHA image files from dataset directory
  FOR each image in dataset DO
    Convert image to grayscale
    Resize image to fixed dimensions (e.g., 100x50)
    Normalize pixel values to range [0, 1]
    Extract label from filename or metadata
  END FOR
  Split dataset into training, validation, and test sets
RETURN preprocessed_images, labels
END
```

Fig a.1 pseudocode 1

Step 2: Define CNN Model Architecture

```
Input: Image dimensions, number of output classes
Output: Compiled CNN model

BEGIN
  Initialize Sequential CNN model
  Add Convolutional layer with filters and ReLU activation
  Add MaxPooling layer
  Repeat Conv + Pool layers as needed
  Flatten the feature maps
  Add Dense (fully connected) layer
  Add Dropout for regularization
  Add final Dense layer with Softmax activation (for multi-class classification)
  Compile model with loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']
RETURN compiled_model
END
```

Fig a.2 pseudocode 2

Step 3: Train the Model

```
Input: Training images, labels, validation data
Output: Trained model

BEGIN
  Fit the model on training data with batch_size and number of epochs
  Use validation data to monitor performance
  Save model checkpoints if validation accuracy improves
  Plot training and validation accuracy/loss over epochs
RETURN trained_model
END
```

Fig a.3 pseudocode 3

Step 4: Evaluate the Model

```
Input: Trained model, test dataset
Output: Accuracy and classification report

BEGIN
  Load test images and labels
  Predict labels using trained model
  Calculate accuracy, precision, recall, F1-score
  Display confusion matrix
RETURN evaluation_metrics
END
```

Fig a.4 pseudocode 4

Step 5: Inference on New CAPTCHA

```
Input: New CAPTCHA image
Output: Predicted CAPTCHA text

BEGIN
  Load image and preprocess (same as training)
  Pass image to trained model
  Predict character probabilities
  Convert probabilities to characters
  Concatenate characters to form CAPTCHA text
RETURN predicted_text
END
```

Fig a.5 pseudocode 5

Step 6: Optional Web Interface (if using Flask)

```
Input: User-uploaded CAPTCHA image via UI
Output: Predicted CAPTCHA displayed on webpage

BEGIN
  Create Flask web app
  Define upload route to receive image
  Preprocess image and predict CAPTCHA using trained model
  Render result on web page
RUN Flask app
END
```

Fig a.6 pseudocode 6

APPENDIX-B

SCREENSHOTS

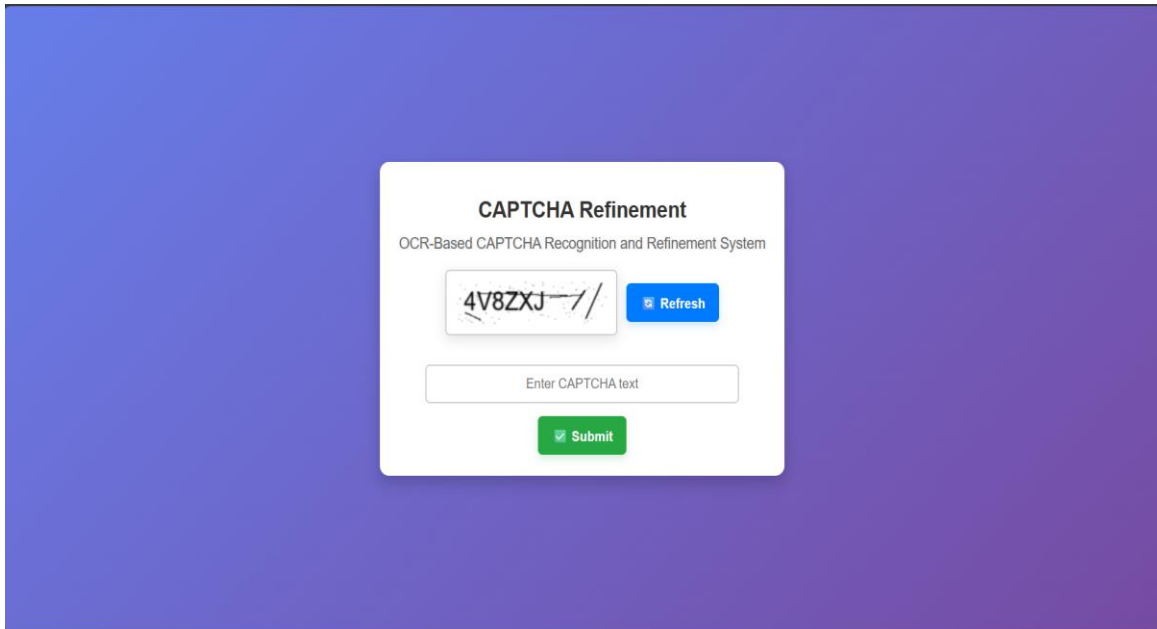


Fig b.1 Output

APPENDIX – C

ENCLOSURES

www.ijcrt.org

© 2025 IJCRT | Volume 13, Issue 5 May 2025 | ISSN: 2320-2882

IJCRT.ORG

ISSN : 2320-2882



**INTERNATIONAL JOURNAL OF CREATIVE
RESEARCH THOUGHTS (IJCRT)**

An International Open Access, Peer-reviewed, Refereed Journal

Develop A ML Model Based Solution To Refine CAPTCHA

Ms. AMIRTHA PREEYA V¹, SHAIK ASLAM², SUJITHA REDDY³, VISWESWAR REDDY⁴

¹ Assistant Professor in computer science and engineering & presidency University, Bangalore

² Student in computer science and engineering & presidency University, Bangalore

³ Student in computer science and engineering & presidency University, Bangalore

⁴ Student in computer science and engineering & presidency University, Bangalore

Abstract:

CAPTCHAs are widely used to differentiate between human users and automated bots, ensuring security in online interactions. However, traditional CAPTCHA systems often suffer from usability issues and vulnerabilities to evolving machine learning-based attacks. This research presents a novel machine learning-driven approach to refining CAPTCHA mechanisms, enhancing both security and user experience.

Our proposed solution leverages deep learning models to analyse and improve CAPTCHA complexity, making it more resistant to automated solvers while maintaining accessibility for genuine users. By utilizing advanced image processing and adaptive challenge generation techniques, the system dynamically adjusts CAPTCHA difficulty based on real-time threat analysis, reducing friction for legitimate users while thwarting bots.

Experimental results demonstrate that our approach significantly improves CAPTCHA robustness against automated attacks while ensuring a seamless experience for human users. This research contributes to the ongoing development of secure and user-friendly authentication mechanisms, bridging the gap between security and usability in modern web applications.

Keywords: CAPTCHA Security,

Machine Learning,

Deep Learning,

Automated Bot Detection,

Adversarial attacks.

I. Introduction:

In the digital age, CAPTCHAs serve as a crucial line of defence against automated bots attempting to exploit online platforms. These challenges, designed to differentiate between human users and machines, are widely used in login systems, online transactions, and content submissions. However, traditional CAPTCHAs often compromise user experience, being either too difficult for humans to solve or too weak to resist modern AI-driven attacks. This has led to an ongoing challenge: creating CAPTCHA systems that are both user-friendly and secure.

With the rapid advancements in artificial intelligence, especially in deep learning and computer vision, automated bots have become increasingly capable of solving conventional CAPTCHA tests. Optical Character Recognition (OCR) technologies and adversarial machine learning techniques have rendered many traditional CAPTCHAs ineffective. This necessitates a shift towards more sophisticated CAPTCHA mechanisms that can adapt dynamically to emerging threats while maintaining accessibility for legitimate users.

This research presents a machine learning-based solution to refine CAPTCHA security by leveraging deep learning techniques for enhanced challenge generation and analysis. By implementing adaptive CAPTCHA models that adjust complexity based on real-time threat detection, our approach aims to strike a balance between security and user convenience. Through this study, we explore how AI can be used not only to break CAPTCHAs but also to strengthen them, ultimately contributing to the development of more robust and user-friendly authentication systems.

A. Background knowledge

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) was introduced as a security measure to prevent automated bots from accessing or manipulating online services. Traditional CAPTCHAs include text-based challenges, image recognition tasks, and interactive puzzles, each designed to exploit the cognitive differences between humans and machines. However, with advancements in artificial intelligence, especially in deep learning, many conventional CAPTCHAs have become vulnerable to automated solvers, necessitating the development of more sophisticated approaches.

Machine learning, particularly deep learning-based models, has played a dual role in the evolution of CAPTCHA systems. On one hand, convolutional neural networks (CNNs) and optical character recognition (OCR) techniques have significantly improved the ability of AI to break text-based CAPTCHAs. On the other hand, AI-driven approaches have also contributed to the development of more secure and adaptive CAPTCHAs. Techniques such as adversarial machine learning, generative adversarial networks (GANs), and behavioural analysis have been explored to enhance CAPTCHA robustness against automated attacks.

In addition to security concerns, usability remains a key challenge in CAPTCHA design. Studies have shown that overly complex CAPTCHAs lead to frustration among users, reducing engagement and accessibility, especially for individuals with disabilities. As a result, modern CAPTCHA solutions are moving toward AI-powered adaptive mechanisms that balance security with user experience. This research builds upon existing CAPTCHA systems

and explores the potential of machine learning to refine CAPTCHA challenges, making them both more secure against AI-driven attacks and more accessible for human users.

B. LITERATURE SURVEY

Breaking the Code [1] examines the vulnerabilities of traditional CAPTCHA systems and how machine learning models have been used to bypass them.

AI vs. CAPTCHA [2] explores the evolution of CAPTCHA technologies in response to advancements in deep learning and automated bot detection.

Human or Bot? [3] delves into different CAPTCHA mechanisms, their usability challenges, and the effectiveness of AI-driven security measures.

Adaptive Security [4] reflects on modern AI-powered CAPTCHA solutions, highlighting adaptive challenge generation and real-time threat analysis.

The Future of CAPTCHA [5] explores the integration of AI and cybersecurity to develop more robust and user-friendly authentication systems.

II. Existing Methods

Several CAPTCHA techniques have been developed to prevent bots from accessing online services. **Text-based CAPTCHAs** were one of the first methods, requiring users to recognize distorted letters and numbers. However, with advancements in Optical Character Recognition (OCR) and AI models, these have become easier for bots to break. **Image-based CAPTCHAs**, like selecting objects in pictures, were introduced as an alternative, but deep learning models can now recognize images with high accuracy.

To improve accessibility, **audio CAPTCHAs** were created, where users listen to distorted speech and type what they hear. However, speech recognition algorithms have made these less secure. **Interactive CAPTCHAs**, such as dragging puzzles or analysing mouse movements, offer better security but can still be mimicked by advanced bots.

While these methods aim to distinguish humans from bots, they often compromise either security or user experience. This research explores a machine learning-based approach to refine CAPTCHA, making it both more secure and user-friendly.

Disadvantages:

- **Vulnerability to AI Attacks** – Advanced machine learning models, such as OCR for text-based CAPTCHAs and CNNs for image-based CAPTCHAs, can easily break traditional CAPTCHA systems.
- **Poor User Experience** – Many CAPTCHAs are difficult to solve, leading to frustration among users. Complex distortions, image selections, or lengthy puzzles can negatively impact accessibility.

- **Accessibility Issues** – Audio CAPTCHAs, designed for visually impaired users, are often difficult to understand due to background noise and distortion, making them ineffective for many users.
- **Increased Bot Sophistication** – Modern bots can mimic human behavior, such as mouse movements and click patterns, reducing the effectiveness of interactive CAPTCHAs.
- **Time-Consuming** – Many CAPTCHAs require multiple attempts or take too long to solve, leading to delays in user authentication and negatively affecting website usability.

III. Proposed Methods

To overcome the limitations of existing CAPTCHA systems, this research proposes a **machine learning-based adaptive CAPTCHA** that enhances security while maintaining user-friendliness. The system dynamically generates CAPTCHA challenges using deep learning models, adjusting difficulty based on real-time threat detection. By leveraging **generative adversarial networks (GANs)** and **behavioural analysis**, the CAPTCHA adapts to evolving bot capabilities, making it harder for AI-driven attacks to succeed.

Unlike traditional CAPTCHAs, the proposed method focuses on **user behaviour analysis**, such as mouse movement patterns, keystroke dynamics, and response times, to distinguish humans from bots. Additionally, the system integrates an **adaptive challenge mechanism**, ensuring that genuine users face minimal friction while bots encounter increasingly complex tasks. This approach enhances security without compromising accessibility, providing a seamless experience for legitimate users.

Advantages:

- **Enhanced Security** – The adaptive CAPTCHA evolves with AI-driven threats, making it difficult for bots to bypass.
- **Improved User Experience** – Challenges are adjusted based on user behaviour, reducing frustration and solving time.
- **Accessibility-Friendly** – Unlike traditional CAPTCHAs, this method minimizes reliance on text and audio, making it more inclusive.
- **Real-Time Threat Detection** – AI continuously monitors and adapts CAPTCHA difficulty based on bot activity patterns.
- **Efficient and Dynamic** – The system ensures a balance between security and usability by dynamically modifying challenge complexity.

IV. Methodology

This section outlines the technologies used, system design, workflow, homepage navigation, database management, user authentication, and data security aspects of the proposed machine learning-based CAPTCHA system.

1. Technologies Used

- **Programming Language:** Python (for backend and AI model development).
- **Deep Learning Model:** Convolutional Neural Networks (CNN) (for CAPTCHA classification and security enhancement)
- **Web Framework:** Flask (for integrating CAPTCHA into web applications)
- **Frontend:** HTML, CSS, JavaScript (for user interaction and CAPTCHA display)

2. System Design and Workflow

- The system first **analyses user behaviour**, such as mouse movements, keystroke dynamics, and response time, to classify whether the user is a human or a bot.
- Based on this classification, an **adaptive CAPTCHA** is generated, where challenge difficulty increases if bot-like behaviour is detected.
- The CAPTCHA **verification process** involves a **CNN model** evaluating the user's responses, ensuring security and ease of access for legitimate users.
- The system continuously **learns from failed bot attempts**, improving CAPTCHA challenges dynamically.

Architecture diagram:



Homepage Navigation:

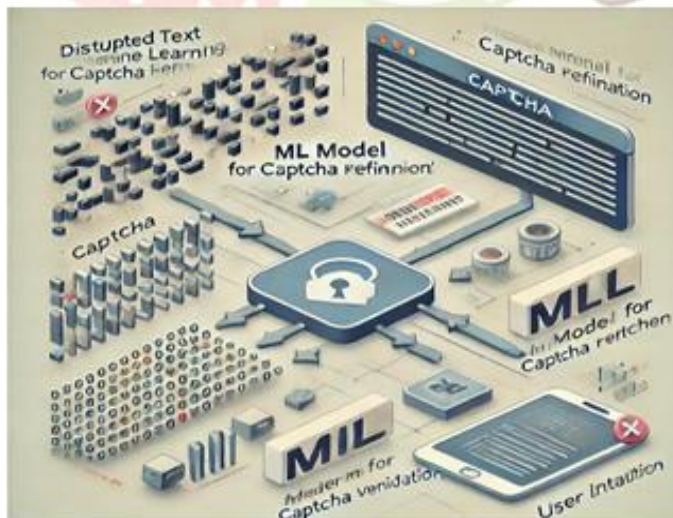
- The homepage includes a **login/register** section with the CAPTCHA challenge integrated.
- If the system detects a legitimate user, a simpler CAPTCHA (e.g., one-click verification) is displayed.
- If suspicious activity is detected, a more complex CAPTCHA (e.g., image-based, puzzle-solving) is shown.

3. Database Management

- User authentication data and CAPTCHA challenge results are stored securely using **SQL databases**.
- CAPTCHA logs are maintained for **pattern analysis and system improvement**.
- Sensitive information is **hashed and encrypted** to prevent unauthorized access.

4. User Authentication and Data Security

- The authentication process uses **multi-layered verification**, including CAPTCHA, password hashing, and optional two-factor authentication.
- Data transmission is secured using **SSL/TLS encryption** to protect against man-in-the-middle attacks.
- The system continuously **monitors suspicious activities**, such as repeated failed CAPTCHA attempts, and takes countermeasures like IP blocking.

Additional Visualizations

V. Requirements

Software Requirements:

- Operating System: Windows, Linux, or macOS.
- Development Environment: Python 3.x, Flask framework.
- Database Management System: MySQL/PostgreSQL.
- Web Technologies: HTML, CSS, JavaScript.

Hardware Requirements:

- **Processor:** Intel i5/i7 or AMD Ryzen 5/7 (or higher)
- **RAM:** Minimum 8GB (Recommended: 16GB for faster training and processing)
- **Storage:** Minimum 50GB free space (Recommended: SSD for better performance)

VI. Conclusion

The proposed machine learning-based CAPTCHA system presents an innovative approach to distinguishing between human users and automated bots. By integrating deep learning with adaptive CAPTCHA mechanisms, the system enhances both security and usability. The use of CNN models for CAPTCHA verification ensures robust protection against automated attacks while maintaining an intuitive experience for genuine users.

Furthermore, leveraging Flask for backend integration allows seamless deployment of CAPTCHA challenges on various web applications. The incorporation of behavioural analysis adds an extra layer of security, making it harder for bots to bypass the system. With an adaptive approach, the CAPTCHA dynamically adjusts difficulty based on detected threats, thereby reducing frustration for legitimate users while strengthening security.

In conclusion, this research contributes to the evolving field of cybersecurity by proposing a smarter and more effective CAPTCHA system. By continuously learning from user interactions and bot behaviours, the system remains resilient against evolving automated threats, ensuring a more secure and efficient web experience for users worldwide.

VII. References

1. "Deep Learning Based CAPTCHA Recognition Network with Grouping Strategy" by J. Zhang et al. (2023)
2. "Recognition of CAPTCHA Characters Using Machine Learning Algorithms" by A. Patel et al. (2022)
3. "CAPTCHA Recognition Using Machine Learning and Deep Learning Techniques" by R. Kumar et al. (2021)
4. "Recognition of CAPTCHA Characters by Supervised Machine Learning Algorithms" by O. Bostik et al. (2020)
5. "Deep-CAPTCHA: A Deep Learning Based CAPTCHA Solver for Vulnerability Assessment" by T. Nguyen et al. (2020)
6. "A CAPTCHA Recognition Technology Based on Deep Learning" by X. Li et al. (2019)

7. "EnSolver: Uncertainty-Aware CAPTCHA Solver Using Deep Ensembles" by M. Khan et al. (2023)
8. "Text Based CAPTCHA Recognition Using Machine Learning and Deep Learning" by S. Rao et al. (2023)
9. "Vulnerability Analysis of CAPTCHA Using Deep Learning" by L. Zhao et al. (2023)
10. "Breaking reCAPTCHA v2: A Study on CAPTCHA Security" by J. Liu et al. (2024)







SHAIK ASLAM -
Develop_a_ML_Based_Solution_to_Refine_CAPTCHA.pdf

ORIGINALITY REPORT

6%	6%	2%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

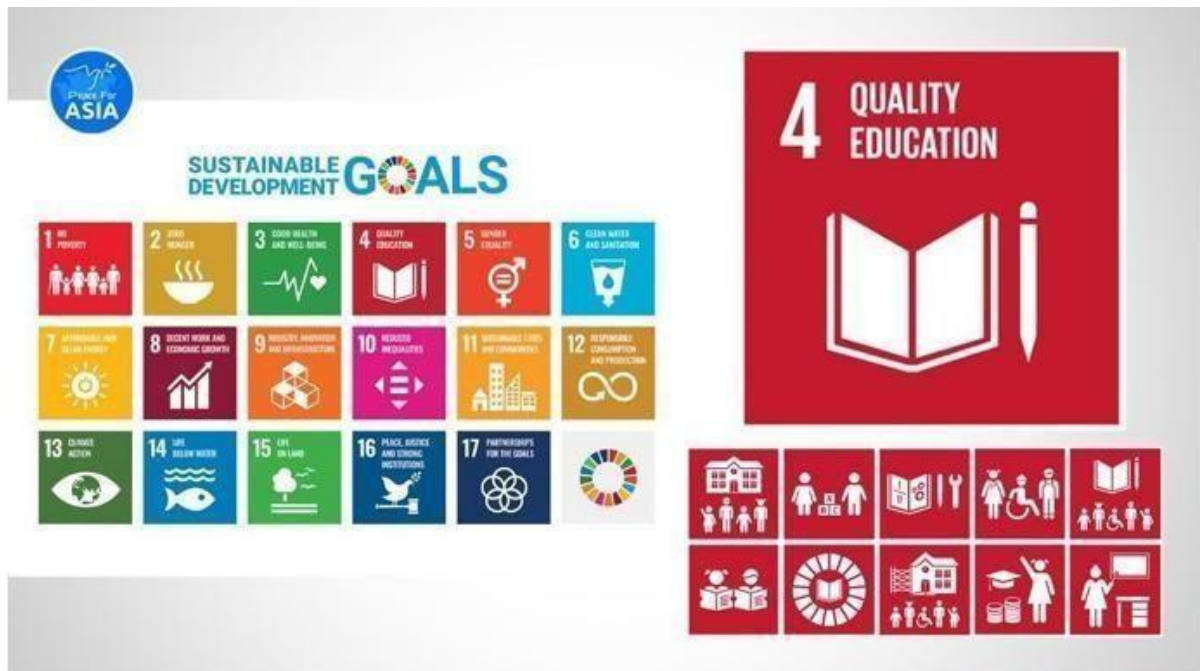
1	ijsrem.com Internet Source	2%
2	www.cysecurity.news Internet Source	1%
3	Submitted to APJ Abdul Kalam Technological University, Thiruvananthapuram Student Paper	1%
4	ijirt.org Internet Source	1%
5	www.rapidinnovation.io Internet Source	<1%
6	www.mdpi.com Internet Source	<1%

Exclude quotes Off
Exclude bibliography On

Exclude matches Off

The Project work carried out here is mapped to SDG-3 Good Health and Well-Being.

The project work carried here contributes to the well-being of the human society. This can be used for Analyzing and detecting blood cancer in the early stages so that the required medication can be started early to avoid further consequences which might result in mortality.



Goal 4: Quality education – It aims to ensure inclusive and equitable quality education and promote lifelong learning opportunities for all, enhancing their skills and job readiness.

Goal 8: Decent Work and Economic Growth – It seeks to promote sustained, inclusive, and sustainable economic growth, full and productive employment, and decent work for all, ensuring they have access to fair job opportunities and decent working conditions.

Goal 9: Industry, Innovation, and Infrastructure - Prioritizing skills, promoting lifelong learning, addressing skills gaps, and ensuring equity and inclusion, hiring platforms can empower individuals with diverse backgrounds to access quality education and fulfilling careers.

Goal 17: Partnership for the goals - By fostering collaborations with educational institutions, training providers, and government agencies, hiring platforms can leverage collective expertise to empower individuals with diverse backgrounds to access quality education and fulfilling careers.

