

Captcha Recognition Using CNN

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Science and Engineering

by

M.Gautam

20BCI0245

Malladi Revanth

20BCI0254

P.Saitejeswarreddy

20BCI0272

Under the guidance of

Prof. N Kopperundevi

School of Computer Science and Engineering

VIT, Vellore.



May, 2024

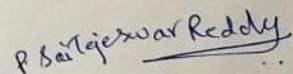
DECLARATION

I hereby declare that the thesis entitled "Captcha Recognition Using CNN " submitted by me, for the award of the degree of Bachelor of Technology in Computer Science and Engineering with specialization in Information Security" to VIT is a record of bonafide work carried out by me under the supervision of Prof . N. Kopperundevi.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree ordiploma in this institute or any other institute or university.

Place : Vellore

Date : 09-05-2024.


Signature of the Candidate

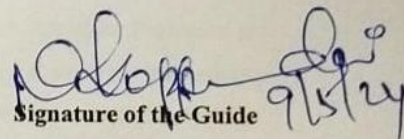
CERTIFICATE

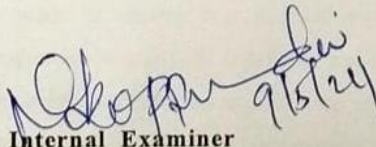
This is to certify that the thesis entitled "Captcha Recognition Using CNN" submitted by M.Gautam (20BCI0245), Malladi Revanth (20BCI0254), P.Saitejeswarreddy (20BCI0272), from SCOPE, VIT, for the award of the degree of Computer Science and Engineering with specialization in Information Security is a record of bonafide work carried out by him / her under my supervision during the period, 03. 01. 2024 to 09.05.2024, as per the VIT code of academic and research ethics.

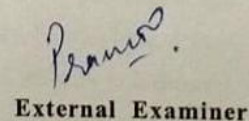
The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and, in my opinion, meets the necessary standards for submission.

Place : Vellore

Date: 09/05/2024


Signature of the Guide


Internal Examiner


External Examiner

Head of the Department

Programme

Head of the Department

Programme

ACKNOWLEDGEMENTS

I am immensely grateful to the honorable Chancellor, Vice Presidents, Vice Chancellor, PRO-VC, and Dean of SCOPE for fostering an environment conducive to growth and for their inspiring leadership during the course.

In this time of excitement, I would like to express my heartfelt gratitude to Professor Gopinath MP, Head of the Department of Information Security at SCOPE, and all of the teaching staff whose tireless efforts have served as pillars of support throughout my academic career. Their unselfish passion and timely encouragement have been vital in providing me with the knowledge required to effectively finish my course of study. I am grateful to my parents for their everlasting support.

It gives me great pleasure to sincerely thank Dr. N.Kopperundevi, Assistant Professor at SCOPE, Vellore Institute of Technology. Throughout my journey, her constant leadership, unceasing support, and profound understanding have been priceless. Most importantly, She taught me the value of patience, which has been essential to all I have done. My relationship with her goes beyond academics; working with someone so knowledgeable and accomplished in Computer Science and Engineering, especially Artificial Intelligence, has been an honor.

I also want to convey my deepest gratitude to my friends, whose persuasion and encouragement inspired me to take on and complete this work. Finally, but not least, I want to express my gratitude and appreciation to everyone who has helped, directly or indirectly, to the successful completion of this project.

Student Name

M. Gautam - 20BCI0245
Malladi Revanth - 20BCI0254
P.Saitejeswarreddy - 20BCI0272

Executive Summary

The thesis presents an extensive evaluation of three distinct models: simple CNN, VGG-16, and Siamese for CAPTCHA recognition across various types of CAPTCHAs. Results reveal the superiority of the simple CNN and Siamese models over VGG-16, particularly in accuracy across all CAPTCHA types. Notably, the simple CNN and Siamese models consistently outperform VGG-16, showcasing higher accuracy rates across different thresholds and CAPTCHA configurations. These findings underscore the significance of model selection and optimization strategies, with the simple CNN and Siamese models emerging as more effective choices for CAPTCHA recognition tasks. Future work will focus on enhancing the CNN model to achieve even greater accuracy, aiming to contribute to the advancement of CAPTCHA recognition technology.

	Page No.
CONTENTS	
Acknowledgement	i.
Executive Summary	ii.
Table of Contents	iii.
List of Figures	iv.
List of Tables	v.
1 INTRODUCTION	1
1.1 Objectives	1
1.2 Motivation	1
1.3 Background	1
2 PROJECT DESCRIPTION AND GOALS	3
2.1 Survey on Existing System	3
2.2 Research Gap	4
2.3 Problem Statement	5
3 TECHNICAL SPECIFICATION	6
3.1 Requirements	6
3.1.1 Functional	6
3.1.2 Non-Functional	6
3.2 Feasibility Study	6
3.2.1 Technical Feasibility	7
3.3 System Specification	7
3.3.1 Software Specification	7
3.3.2 Hardware Specification	8
4 DESIGN APPROACH AND DETAILS	9
4.1 System Architecture	9
4.2 Constraints, Alternatives and Tradeoffs	10
5 SCHEDULE, TASKS AND MILESTONES	11
5.1 Module Description	11
5.1.1 Module - 1	11
5.1.2 Module - 2	11
5.1.3 Module - 3	11
5.1.4 Module - 4	11
5.1.5 Module - 5	12

	5.1.6 Module - 6	12
6	PROJECT DEMONSTRATION	14
7	RESULT & DISCUSSION	16
8	SUMMARY	18
9	REFERENCES	19
	APPENDIX A – SAMPLE CODE	21

List of Figures

Figure No.	Title	Page No.
4.1	System Architecture	9
5.2	Architecture diagram of vgg16	12
6.1	Training dataset for CNN and VGG16 models	14
6.2	Training dataset for SNN model	15
7.1	Loss vs epoch graph for SNN model	16
7.2	SNN Model prediction	16

List of Tables

Table No.	Title	Page No.
7.1	CNN model results for different captcha	16
7.2	Vgg16 model results for different captcha	17

List of Abbreviations

CNN

Convolutional Neural Network

Vgg16

Visual Geometry Group 16

SNN

Siamese Neural Network

1. INTRODUCTION

1.1. OBJECTIVE

The primary objective of our project is to advance the field of artificial intelligence in tackling complex CAPTCHAs. Through the implementation of Convolutional Neural Networks (CNNs) and leveraging advanced neural network architectures, our goal is to develop an automated system with high efficiency and accuracy in predicting challenging CAPTCHAs. This project focuses on pushing the boundaries of AI capabilities to effectively solve intricate CAPTCHA tests, contributing to the ongoing evolution of sophisticated AI models for challenging online security measures. The primary objective of our project is to advance the field of artificial intelligence in tackling complex CAPTCHAs. Through the implementation of Convolutional Neural Networks (CNNs) and leveraging advanced neural network architectures, our goal is to develop an automated system with high efficiency and accuracy in predicting challenging CAPTCHAs. This project focuses on pushing the boundaries of AI capabilities to effectively solve intricate CAPTCHA tests, contributing to the ongoing evolution of sophisticated AI models for challenging online security measures.

1.2. MOTIVATION

The motivation behind the project on CAPTCHA recognition using Convolutional Neural Networks (CNNs) stems from the escalating need to fortify network security in the face of advancing internet technologies. CAPTCHA, a widely adopted tool to discern between human users and automated bots, faces mounting challenges due to the evolution of malicious computer attacks. While traditional CAPTCHA methods exhibit robustness, they often lack adaptability and suffer from poor performance in certain scenarios. By harnessing the power of CNNs, the project aims to enhance CAPTCHA recognition by leveraging deep neural networks' ability to learn intricate image features without the need for extensive preprocessing or artificial design elements. This endeavor seeks to evaluate the efficacy of various CAPTCHA designs in thwarting automated attacks and ultimately advance the field of artificial intelligence by pushing the boundaries of AI capabilities in solving complex CAPTCHA tests, thereby bolstering online security measures.

1.3. BACKGROUND

The background for the project on CAPTCHA recognition using Convolutional Neural Networks (CNNs) emerges from the imperative to address escalating network security concerns amidst rapid internet advancements. CAPTCHA, an essential tool in

differentiating between human users and automated bots, faces increasing challenges due to evolving malicious computer attacks. Although traditional CAPTCHA methods demonstrate resilience, they often lack adaptability and efficacy in certain scenarios. By harnessing CNNs, the project aims to bolster CAPTCHA recognition by capitalizing on the deep learning capabilities of neural networks, enabling them to discern intricate image features without extensive preprocessing or artificial design elements. This initiative seeks to evaluate the effectiveness of diverse CAPTCHA designs in thwarting automated attacks and ultimately enhance online security measures by advancing AI capabilities in solving intricate CAPTCHA tests.

2. PROJECT DESCRIPTION AND GOALS

2.1. SURVEY ON EXISTING SYSTEMS

1. TITLE: End-to-End Captcha Recognition Using Deep CNN-RNN Network

AUTHOR: Yujin Shu , Yongjin Xu

SUMMARY: The proposed CNN-RNN model utilizes a deep residual convolutional neural network to extract input captcha features accurately. It then employs a two-layer GRU network to further extract deep internal features from the captcha. Finally, the model outputs a sequence representing the four-character captcha. Experimental results demonstrate the end-to-end deep CNN-RNN network's high performance, achieving a remarkable 99% accuracy across various captcha datasets. This approach offers a robust solution for captcha recognition tasks, leveraging both CNNs and RNNs effectively.

2. TITLE: Captcha Recognition using convolutionalneural networks with low structural complexity

AUTHOR: Ponnusamy, P., Ghias, A. R., Guo, C., & Sarikaya, R.

SUMMARY: This project employs Convolutional Neural Networks (CNNs) to enhance captcha recognition while analyzing effective captcha characteristics. Captcha images are preprocessed by converting them to gray scale to reduce noise and background interference. Filenames are stored as strings for reference. Each image is resized and reshaped, and a 5x36 array is created to represent captcha characters at each position. Characters are identified from filenames and corresponding locations in the array are updated, aiming to improve recognition accuracy and understand captcha effectiveness.

3. TITLE: CAPTCHA Recognition Method Based on CNN with Focal Loss

AUTHOR: Zhong Wang, Peibei Shi

SUMMARY: In order to distinguish between computers and humans, CAPTCHA is widely used in links such as website login and registration. The traditional CAPTCHA recognition method has poor recognition ability and robustness to different types of verification codes. For this reason, the paper proposes a CAPTCHA recognition method based on convolutional neural network with focal loss function. This method improves the traditional VGG network structure and introduces the focal loss function to generate a new CAPTCHA recognition model.

4. TITLE: CAPTCHA Recognition Using Deep Learning with Attached Binary Images

AUTHOR: Prof. Ammar Hawbani

SUMMARY: This study addresses the challenge of CAPTCHA recognition amidst advancements in deep learning, proposing an efficient CNN model. By leveraging

attached binary images, the model enhances CAPTCHA recognition accuracy. The approach involves replicating the input CAPTCHA image for each character present and attaching distinct binary images to each copy. This novel technique enables the CNN model to effectively recognize CAPTCHAs, thereby enhancing website security against automated attacks.

5. TITLE: SiameseCCR: a novel method for one-shot and few-shot Chinese CAPTCHA recognition using deep Siamese network

AUTHOR: Zhe Chen, Weifeng Ma, Nanfan Xu, Caoting Ji, Yulai Zhang

SUMMARY: The article introduces a novel method utilizing a deep Siamese network for recognizing Chinese CAPTCHAs. Traditional methods struggle with evolving designs and require extensive labeled data. The proposed network excels in one-shot and few-shot scenarios, learning character embeddings to compare against unknown CAPTCHAs. It offers robustness to noise and distortions, outperforming traditional methods. Future directions include incorporating attention mechanisms and utilizing larger datasets for improved generalizability. This research presents a promising solution for efficient CAPTCHA recognition with minimal data requirements

6. TITLE: CAPTCHA Recognition with Active Deep Learning

AUTHOR: Fabian Stark, Caner Hazırba, s, Rudolph Triebel, and Daniel Cremers

SUMMARY: In CAPTCHA Recognition with Active Deep Learning, the focus lies on leveraging active learning techniques to improve model performance. This approach involves iteratively selecting informative data samples for annotation, thus maximizing model learning efficiency. By prioritizing uncertain or difficult instances during training, the model can better adapt to complex CAPTCHA variations. Active Deep Learning aims to reduce annotation costs and accelerate model convergence by intelligently selecting data for training.

2.2. GAPS IDENTIFIED

a) Limited Robustness: Traditional CAPTCHA recognition methods often lack robustness against evolving CAPTCHA designs. They may struggle with variations in font styles, distortions, noise, or complex backgrounds, leading to decreased accuracy in recognition.

b) Poor Stickiness: While traditional methods may achieve decent performance under specific conditions, their stickiness, or the ability to maintain performance across various types of CAPTCHAs, is often poor. They may fail when confronted with novel CAPTCHA designs or modifications.

c) Manual Feature Engineering: Traditional methods often rely on manual feature

engineering, requiring experts to design and extract relevant features from CAPTCHA images. This process can be time-consuming and may not capture all essential characteristics, limiting the system's adaptability to new CAPTCHA variations.

2.3. PROBLEM STATEMENT

As the Internet has grown, so too has the usage of captcha technology. Captcha technology is used to distinguish between humans and machines, namely Completely Automated Public Turing test to tell Computers and Humans Apart. This project builds an end-to-end deep CNN network model by analyzing the technology behind captcha recognition, which makes it possible to recognize text captchas with four.

3. TECHNICAL SPECIFICATION

3.1. REQUIREMENTS

3.1.1 Functional Requirements :

The CAPTCHA recognition project using Convolutional Neural Networks (CNNs) necessitates the development of a comprehensive CAPTCHA recognition system. This system should adeptly receive CAPTCHA images as input and undergo preprocessing to optimize recognition accuracy. Subsequently, the system should segment the characters within the CAPTCHA image and accurately recognize each character. The final output should comprise the recognized characters in the correct sequence. Implementation of CNNs forms a pivotal aspect of this project, requiring the creation and training of convolutional neural networks using a diverse dataset of CAPTCHA images. Furthermore, the system must undergo rigorous testing and evaluation, assessing its performance across various CAPTCHA designs and its ability to differentiate between human users and automated bots. Essential metrics such as accuracy, precision should be measured to gauge the system's efficacy accurately.

3.1.2 Non Functional Requirements

In addition to functional specifications, the CAPTCHA recognition project must meet several non-functional requirements to ensure its effectiveness, scalability, and reliability. Performance is paramount, necessitating high accuracy and efficient recognition within reasonable timeframes. The system's scalability is crucial, enabling it to accommodate a burgeoning volume of CAPTCHA requests and escalating computational demands as the user base expands. Robustness is another key consideration, requiring the system to perform consistently across diverse CAPTCHA designs and complexities. Usability concerns dictate the need for a user-friendly interface that provides clear feedback on recognition results, enhancing user experience. Security considerations encompass adherence to best practices to thwart unauthorized access and ensure data privacy throughout the recognition process. Reliability is imperative, necessitating minimal downtime and graceful error handling to maintain uninterrupted service. Lastly, maintainability is crucial, emphasizing modularity and extensibility for seamless updates and enhancements to the recognition algorithms and security measures, thereby ensuring the project's long-term viability and efficacy.

3.2. FEASIBILITY STUDY

3.2.1 Technical Feasibility

The technical feasibility of the CAPTCHA recognition project using Convolutional

Neural Networks (CNNs) is supported by several factors:

1. Availability of Data: Datasets containing CAPTCHA images for training CNNs are readily accessible. These datasets include various CAPTCHA designs and complexities, enabling comprehensive model training.
2. Advancements in Deep Learning Frameworks: Robust deep learning frameworks such as TensorFlow and PyTorch provide extensive support for building and training CNN models. These frameworks offer efficient tools for implementing complex neural network architectures and optimizing model performance.
3. Computational Resources: With the availability of powerful GPUs and cloud computing services, the computational requirements for training CNN models can be met effectively. These resources facilitate faster model training and experimentation, accelerating the development process.
4. Existing Research and Techniques: The field of CAPTCHA recognition using CNNs has seen significant advancements, with numerous research papers and techniques available for reference. This existing knowledge base provides valuable insights and methodologies for implementing and improving CAPTCHA recognition systems.
5. Open-Source Libraries and Tools: A plethora of open-source libraries and tools are available for image processing, data augmentation, and model evaluation, simplifying the implementation and experimentation process. These tools accelerate development and reduce the need for building components from scratch.

3.3. SYSTEM SPECIFICATION

3.3.1 Software Specification

For the software specifications of the CAPTCHA recognition project, Google Colab serves as the primary platform for development and execution. Google Colab provides a cloud-based environment with free access to GPU resources, making it ideal for training deep learning models such as Convolutional Neural Networks (CNNs). The project will leverage popular deep learning frameworks such as TensorFlow or PyTorch, both of which are fully compatible with Google Colab. These frameworks offer extensive support for building, training, and evaluating CNN models for image recognition tasks. Additionally, the project will utilize standard Python libraries for image processing, data manipulation, and model evaluation. Google Colab's collaborative features enable seamless sharing and collaboration among project team members, fostering efficient development workflows. Overall, the software specifications include the utilization of

Google Colab as the development platform, TensorFlow or PyTorch as the deep learning framework, and Python libraries for image processing and model evaluation.

3.3.2 Hardware Specification

In terms of hardware specifications, the project will leverage CUDA-enabled GPUs for accelerated model training and inference. CUDA is a parallel computing platform and programming model developed by NVIDIA, specifically designed for GPU-accelerated computing tasks. By utilizing CUDA-enabled GPUs, the project can significantly expedite the training process of CNN models, which typically require intensive computational resources. The choice of CUDA GPUs ensures compatibility with deep learning frameworks such as TensorFlow and PyTorch, which offer native support for GPU acceleration. Additionally, the project may utilize cloud-based GPU instances for scalability and flexibility in accessing computational resources. The hardware specifications encompass the use of CUDA-enabled GPUs to support efficient model training and inference, thereby enhancing the project's performance and scalability.

4. DESIGN APPROACH AND DETAILS

4.1 SYSTEM ARCHITECTURE

The system architecture for CAPTCHA recognition begins with data collection, where a diverse dataset of CAPTCHA images is gathered, incorporating various fonts, backgrounds, noise levels, and character distortions for robust model training. This dataset must cover all possible characters and symbols within the CAPTCHA system. Following data collection, preprocessing steps involve resizing images to a fixed size for uniformity, converting them to grayscale to reduce complexity, and normalizing pixel values to a 0-1 range. Segmentation techniques divide the CAPTCHA image into four equal-width segments, facilitating character extraction.

Moving to model architecture, a tailored CNN architecture is designed with convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification, incorporating dropout to prevent overfitting. Experimentation with various architectures and hyper parameters is crucial for optimal model selection. Once the architecture is established, the CNN model undergoes training using the dataset, employing data augmentation techniques to increase diversity and improve generalization. Performance monitoring on a validation set enables real-time adjustments to hyperparameters for optimized performance and robustness. This system architecture ensures efficient and accurate CAPTCHA recognition, enhancing online security against automated attacks.

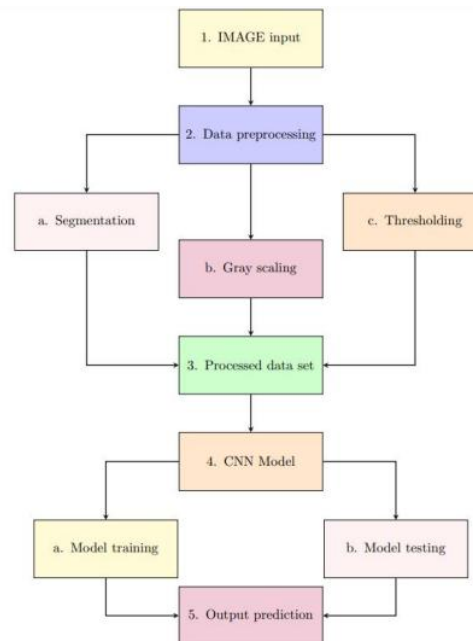


Figure 1: System Architecture

Fig-4.1: System Architecture

4.2. CONSTRAINTS AND ALTERNATIVES

1. **Data Availability:** Limited availability of diverse and labeled CAPTCHA datasets may restrict model training and performance.
Alternative: Data augmentation techniques can be employed to artificially expand the dataset.
2. **Computational Resources:** Constraints on computational resources, especially GPUs for model training, may impact the scalability and speed of the project.
Alternative: Utilize cloud-based GPU instances or distributed computing resources to mitigate resource limitations.
3. **Model Complexity:** Developing complex CNN architectures may lead to longer training times and increased computational costs.
Alternative: Consider using pre-trained models or simpler architectures to expedite development.
4. **Overfitting:** Complex models may be prone to over-fitting, especially with limited training data.
Alternative: Implement regularization techniques such as dropout and L2 regularization to mitigate over-fitting.
5. **Evaluation Metrics:** Determining appropriate evaluation metrics for CAPTCHA recognition, especially in the absence of ground truth labels, can be challenging.
Alternative: Use a combination of metrics such as accuracy, precision, recall, and F1-score, and consider human evaluation for qualitative assessment.

5. SCHEDULE, TASKS AND MILESTONES

5.1 MODULE DESCRIPTION

5.1.1 Module 1(Claptcha)

The Claptcha module is a Python library designed for generating CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) images. It offers customization options for the generated CAPTCHAs, including text, font, background color, noise, and distortion levels. Claptcha provides a straightforward API for integration into web applications, aiding in preventing automated bot submissions. With adjustable parameters for appearance and complexity, it enhances security by creating effective challenges for automated systems. The module typically relies on dependencies like PIL or Pillow for image processing tasks and is commonly distributed under an open-source license, ensuring flexibility and accessibility for developers.

5.1.2 Module 2(Keras)

The Keras module, renowned for its simplicity and versatility, offers a powerful platform for developing neural network architectures—a crucial component in combating automated attacks like CAPTCHA breaches. With its seamless integration with TensorFlow, Theano, or CNTK, Keras empowers developers to swiftly prototype and deploy robust machine learning models, including those geared towards CAPTCHA generation or recognition tasks. Leveraging its user-friendly interface and extensive documentation, Keras facilitates the rapid creation of intricate neural networks, ideal for enhancing security measures against automated bot submissions. Whether for image recognition, text analysis, or other intricate tasks, Keras stands as a cornerstone tool, empowering developers to fortify their systems against malicious activities with ease and efficiency.

5.1.3 Module 3(Loss function)

Categorical cross-entropy, a loss function commonly used in multi-class classification issues, quantifies the difference between the true probability distribution of the classes and the predicted probability distribution. It works especially well in situations where, every instance is a member of a single class. The following is the formula for categorical cross-entropy:

$$\text{Categorical Cross-Entropy Loss} = - \sum_{i=1}^C y_i \cdot \log(p_i) \quad -E_p(1)$$

5.1.4 Module 4(VGG16)

The VGG16 (Visual Geometry Group 16) model is a well-known convolutional neural network architecture for image categorization applications. The Oxford Visual Geometry Group designed the VGG16 set of 16 weight layers. These layers are mostly max-pooling and convolutional layers with tiny 3x3 filters. This consistent filter size keeps the structure

simple while encouraging a deeper network. Fully linked layers, which are in charge of higher-order reasoning, are stacked after the convolutional layers. VGG16's architecture exhibits a distinctive characteristic of maintaining a consistent 3x3 receptive field throughout the convolutional layers, leading to a more expressive and hierarchical feature representation. Despite its simplicity, VGG16 has demonstrated remarkable performance on various visual recognition benchmarks, becoming a foundational reference point in deep learning for image classification. Its intuitive design and robust performance have influenced subsequent convolutional neural network architectures and paved the way for advancements in image understanding tasks.

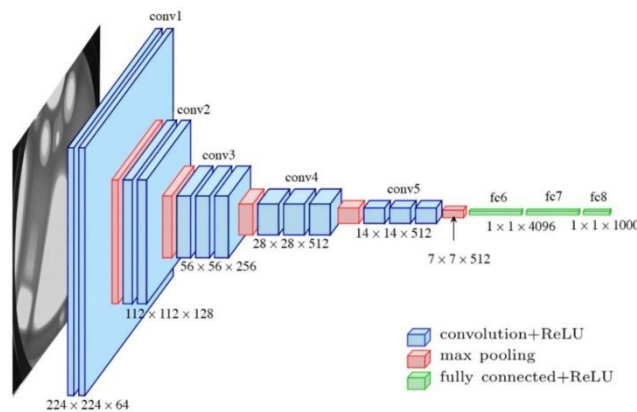


Figure 5.1: Architecture diagram of VGG16

5.1.5 Module 5(Siamese Network)

Siamese networks consist of twin sub-networks with identical parameters and weights, designed for tasks requiring similarity measurement like image identification and facial recognition. They generate fixed-length embeddings of input data, ensuring similar inputs yield close embeddings. These networks efficiently learn complex similarities and adapt well to input variations. One-shot learning strategies teach models to identify new classes with minimal labeled data, while few-shot learning techniques enable learning from a limited number of instances per class. Episodic training and meta-learning aid in this process, allowing models to quickly adapt to new tasks or generalize from limited samples.

5.1.6 Module 6(Contrastive Loss Function)

In siamese neural networks, the Contrastive Loss function is used to direct the embedding representation learning process for pairs of input data. Its goal is to push dissimilar inputs apart and encourage comparable inputs to have embeddings that are near together in the embedding space. The Euclidean distance metric is used in this loss function to calculate the distance between the embeddings of pairs. The network is penalized when the distance between embeddings of similar pairings is too great, and it is penalized when the distance

between embeddings of dissimilar pairs is less than a given margin. These two terms make up the loss. This margin helps ensure that dissimilar pairs are sufficiently separated in the embedding space. By optimizing this loss function during training, the siamese network learns to produce embeddings that effectively capture the underlying similarity or dissimilarity between pairs of input data.

$$L = \frac{1}{N} \sum_i (1 - y_i) \cdot d_i^2 + y_i \cdot \max(0, m - d_i)^2$$

L: the overall loss.

N : the batch size.

y_i : the ground truth label indicating similarity (0 for similar pairs, 1 for dissimilar pairs).

d_i : the Euclidean distance between the embeddings of the i th pair.

M: the margin hyper-parameter.

6.PROJECT DEMONSTRATION

6.1. DATASETS AND FEATURES

We generated datasets using the Claptcha Python package, which automatically produces PNG format images in RGB (3 channels). A ratio of 10:1 is used to split the dataset into training and testing sets, with 1000 samples set aside for training and 100 samples for testing. The characters utilized include numbers 0 to 9 and letters a-z and A-Z, rendered in the Freemono font. Additionally, dots and lines are added to the figures using the internally implemented package in Claptcha. CNN model and VGG16 model Dataset and preprocessing shown in Figure-3.

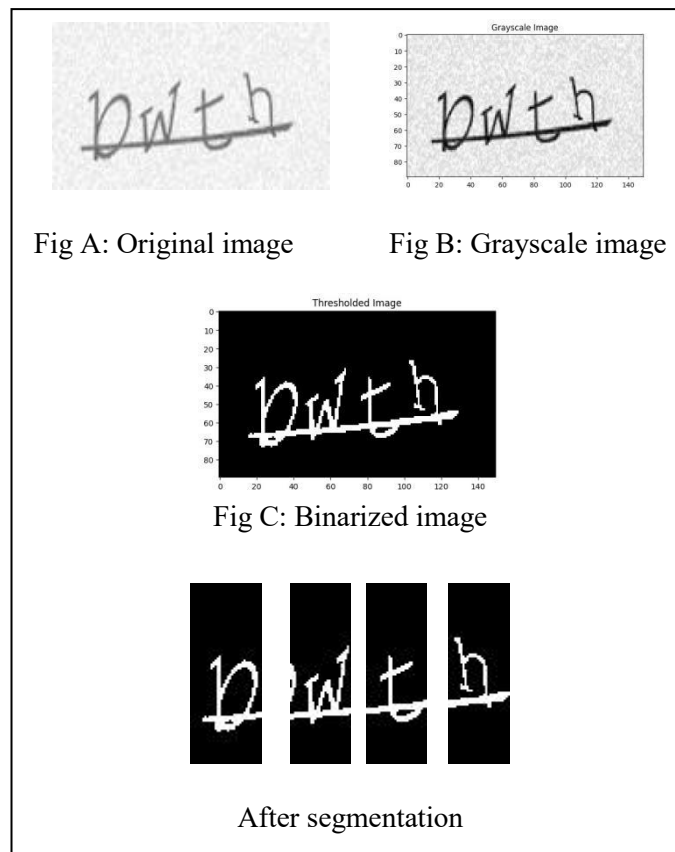


Figure 6.1: Training dataset Generation for CNN model

An image Folder Dataset, or simply a collection of photographs arranged into folders according on their classifications or categories, is the input used by the Siamese Network Dataset class. Additionally, the class permits the application of optional picture transformations, including scaling or normalization. In the process of creating the dataset, the class chooses one image (img0_tuple) at random for each index. Next, it chooses at random which image to choose from a different category (should_get_same_class = 0) or another image from the same category (should_get_same_class = 1). Upon making this determination, it proceeds to choose an additional image (img1_tuple) from either the same

category (should should_get_same_class == 1) or an alternate category (should should_get_same_class == 0). Following the selection of the image pairs, the matching images are opened and turned into grayscale ("L" mode). Any given transformations are applied to both pictures. Siamese network model Dataset and preprocessing shown in Figure-4.

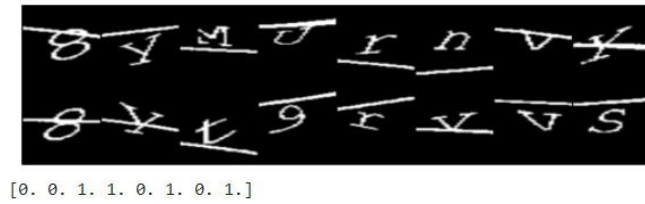


Figure 6.2: Training Dataset for Siamese network

7. RESULT & DISCUSSION

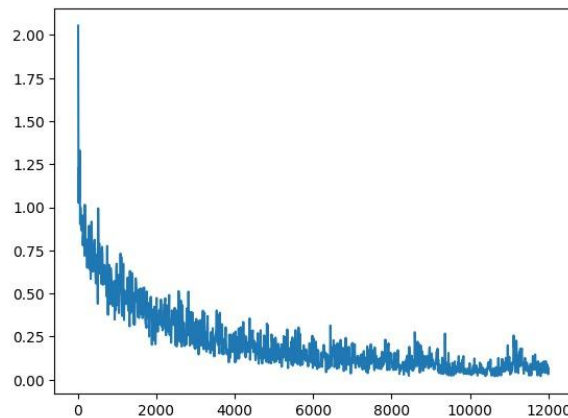


Figure 7.1: Loss vs Counter Graph for Siamese Model

In our study, we trained a Siamese network over 100 epochs, each consisting of 12 counters, resulting in a total of 1200 iterations. Through this training process, we observed a steady decrease in the loss function, reaching a final value of 0.0463. This reduction underscores the network's capability to effectively learn discriminative embeddings, indicative of its potential for accurate similarity detection in real-world applications. Siamese neural network prediction based on the dissimilarity score is shown in Figure-6.



Figure 7.2: Siamese Model prediction

Predicted class: 3

Dissimilarity score: 0.23833371698856354

Table -7.1 : CNN model results for different captcha

	Captcha with only digits	Captcha with only capital letters	Captcha with only small letters	Captcha with all digits + capital letters +small letters
0% correct	0	0	0	0
25% correct	0	0	2	4
50% correct	3	10	7	13
75% correct	13	38	34	39
100% correct	84	52	47	44
Test accuracy	84%	52%	47%	44%
Total images	100	100	100	100

- Each row represents a different accuracy threshold, ranging from 0% correct to 100% correct, along with the corresponding number of correctly identified CAPTCHAs.
- Columns represent different CAPTCHA types: "Captcha with only digits," "Captcha with only capital letters," "Captcha with only small letters," and "Captcha with all digits+capital letters+small letters."
- The "Test accuracy" row indicates the overall accuracy percentage achieved by each model across all CAPTCHA types.
- The "Total images" row denotes the total number of images tested for each CAPTCHA type.

Table -7.2 : VGG16 model results for different captcha

	Captcha with only digits	Captcha with only capital letters	Captcha with only small letters	Captcha with all digits+ capital letters +small letters
0% correct	2	11	3	24
25% correct	5	33	28	41
50% correct	18	31	34	24
75% correct	34	20	27	9
100% correct	41	5	8	2
Test accuracy	41%	5%	8%	2%
Total images	100	100	100	100

- Each row represents a different accuracy threshold, and each column represents a different CAPTCHA type.
- The numbers in each cell indicate the number of CAPTCHAs correctly identified by the model at the corresponding accuracy threshold.
- The "Test accuracy" row indicates the overall accuracy percentage achieved by each model across all CAPTCHA types.
- The "Total images" row denotes the total number of images tested for each CAPTCHA type.

8. SUMMARY

The experimental evaluation reveals significant disparities in performance between the simple CNN, VGG-16, and Siamese models across various types of CAPTCHAs. VGG-16 consistently performs poorly compared to the simple CNN and Siamese models. This is evident across all CAPTCHA types, where the simple CNN and Siamese models achieve higher accuracy rates. Specifically, for CAPTCHAs containing only digits, lowercase letters, uppercase letters, and combinations thereof, the simple CNN and Siamese models consistently outperform VGG-16, with substantially higher accuracy rates (Simple CNN: digits=84%, lowercase=52%, uppercase=47%, combination=44%; VGG-16: digits=41%, lowercase=8%, uppercase=5%, combination=2%).

Even when considering perfect accuracy (100% correct threshold), where all models are evaluated on their ability to predict all characters correctly, the simple CNN MODEL maintains a significant advantage over VGG-16. This is evidenced by the higher number of correct predictions achieved by the simple CNN model across all CAPTCHA types.

In summary, while both the simple CNN model performs well across various CAPTCHA types, the VGG-16 model lags behind in terms of accuracy. The discrepancy in performance underscores the importance of careful model selection and optimization strategies. Further efforts will be directed towards improving the CNN model to provide even higher accuracy in future iterations.

9. REFERENCES

- [1] Y. Shu and Y. Xu, "End-to-End CAPTCHA Recognition Using Deep CNN-RNN Network," presented at the 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 2019, pp. 54-58, doi:10.1109/IMCEC46724.2019.8983895.
- KolupaevA,OgijenkoJ.CAPTCHAs:Humansvs.bots[J].IEEE Security and Privacy, 2008, 6 (1) :68-70
- [2] Captcha Recognition using convolutional neural networks with low structural complexity, Haolin Yang.
- [3] CAPTCHA Recognition Using Deep Learning with Attached Binary Images by Alaa Thobhani, Mingsheng Gao, Ammar Hawbani, Safwan Taher Mohammed Ali, and Amr Abdussalam
- [4] Wang, Zhong & Shi, Peibei. (2021). CAPTCHA Recognition Method Based on CNN with Focal Loss. Complexity. 2021. 1-10. 10.1155/2021/6641329.
- [5] P. Wang, H. Gao, Z. Shi, Z. Yuan, and J. Hu, "Simple and easy: transfer learning-based attacks to text CAPTCHA," IEEE Access, vol. 8, pp. 59044–59058, 2020.
- [6] [6] Thobhani, Alaa & Gao, Mingsheng & Hawbani, Ammar & Mohammed, Safwan. (2020). CAPTCHA Recognition Using Deep Learning with Attached Binary Images. Electronics. 9. 10.3390/electronics9091522
- [7] Chen, Zhe & Ma, Weifeng & Xu, Nanfan & Ji, Caoting & Zhang, Yulai. (2020). SiameseCCR: a novel method for one-shot and few-shot Chinese CAPTCHA recognition using deep Siamese network. IET Image Processing. 14. 10.1049/ietipr.2019.0618.
- [8] M. Guerar, L. Verderame, M. Migliardi, F. Palmieri, and A. Merlo, "Gotta CAPTCHA 'em all: A survey of 20 years of the human-or-computer dilemma," ACM Comput. Surv., vol. 54, no. 9, Oct. 2021.
- [9] T. Kimbrough, P. Tian, W. Liao, E. Blasch, and W. Yu, "Deep CAPTCHA Recognition Using Encapsulated Preprocessing and Heterogeneous Datasets," presented at the IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, 2022, pp. 1-6, doi: 10.1109/INFOCOMWKSHPS54753.2022.9798233. Keywords: Computers, Training, Performance evaluation, Tracking, Conferences, Computer architecture, Software systems, Deep learning, CAPTCHA, Cybersecurity applications.

- [10] J. Wang, J. Qin, J. Qin, X. Xiang, Y. Tan, and N. Pan, "CAPTCHA recognition based on deep convolutional neural network," *Mathematical Biosciences and Engineering*, vol. 16, no. 5, pp. 5851–5861, 2019.
- [11] K. L. Wiggers, A. S. Britto, L. Heutte, A. L. Koerich and L. S. Oliveira, "Image Retrieval and Pattern Spotting using Siamese Neural Network," 2019 International Joint Conference on Neural Networks (IJCNN), Budapest, Hungary, 2019, pp. 1-8, doi: 10.1109/IJCNN.2019.8852197. keywords: {Image retrieval;Feature extraction;Neural networks;Task analysis;Computer architecture;Training;Deep learning;Siamese network;image retrieval;pattern spotting}
- [12] Li, Yikai & Chen, C. & Zhang, Tong. (2022). A Survey on Siamese Network: Methodologies, Applications and Opportunities. *IEEE Transactions on Artificial Intelligence*. PP. 1-21. 10.1109/TAI.2022.3207112.
- [13] Dr.N.Kopperundevi, et.al, (2023), "Sentiment Analysis for Online Product Reviews and Recommendation Using Deep Learning Based Optimization Algorithm", *International Journal on Recent and Innovation Trends in Computing and Communication* ISSN: 2321-8169 Volume: 11 Issue: 9, pp 3629-3640.
- [14] Dr.N.Kopperundevi, et.al, (2023), "Bio-Inspired Optimization with Transfer Learning Based Crowd Density Detection on Sparse Environment ", *Proceedings - 2023 3rd International Conference on Pervasive Computing and Social Networking, ICPCSN* , pp. 86 – 90.
- [15] N.Kopperundevi, S.Suresh Kumar (2017), "Biomedical Regularization Technique to Restore Woven Fabrics using Sparsity Through Wavelet Transforms", *Research journal of biotechnology*, Vol. 2017(2)Special Issue- II, pp. 42-46.

APPENDIX A – SAMPLE CODE

A)CNN Model:

```

classifier_0 = Sequential()
classifier_0.add(Conv2D(16, (5, 5), input_shape = (32, 32, 1), activation = 'relu'))
classifier_0.add(MaxPooling2D(pool_size = (2, 2),strides=(2,2)))
classifier_0.add(Conv2D(16, (5, 5), activation = 'relu'))
classifier_0.add(MaxPooling2D(pool_size = (2, 2),strides=(2,2)))
classifier_0.add(Flatten())
classifier_0.add(Dense(units = 250, activation = 'relu'))
classifier_0.add(Dense(units = 62, activation = 'softmax'))
classifier_0.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =
['accuracy'])

```

B)Siamese Network Model(SNN):

```

class SiameseNetwork(nn.Module):
    def __init__(self):
        super(SiameseNetwork, self).__init__()
        self.cnn1 = nn.Sequential(
            nn.Conv2d(1, 96, kernel_size=11,stride=4),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(3, stride=2),
            nn.Conv2d(96, 256, kernel_size=5, stride=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(2, stride=2),
            nn.Conv2d(256, 384, kernel_size=3,stride=1),
            nn.ReLU(inplace=True)
        )
        self.fc1 = nn.Sequential(
            nn.Linear(384, 1024),
            nn.ReLU(inplace=True),
            nn.Linear(1024, 256),
            nn.ReLU(inplace=True),
            nn.Linear(256,2)
        )
    def forward_once(self, x):
        output = self.cnn1(x)
        output = output.view(output.size()[0], -1)
        output = self.fc1(output)
        return output
    def forward(self, input1, input2):
        output1 = self.forward_once(input1)
        output2 = self.forward_once(input2)
        return output1, output2

```

C)VGG16 Model:

```

from tensorflow import keras
from keras.applications import VGG16
from keras.models import Model
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,

```

```
BatchNormalization
from keras.callbacks import EarlyStopping
from keras.optimizers import RMSprop
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,224,3))
for layer in base_model.layers:
    layer.trainable = False
x = base_model.output
x = Conv2D(128, (5, 5), activation='relu')(x)
x = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)
x = Flatten()(x)
x = Dense(units=512, activation='relu')(x)
x = Dense(units=62, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=x)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```


BEST PAPER AWARD

International Conference on “Innovations and Technology Development in Electronics, Computer and Communication (ITDECC-2024)”

