International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 10, Issue 4, April-2025, http://ijmec.com/, ISSN: 2456-4265

CAPTCHA Recognition And Analysis Using Custom Based CNN Model - Capsecure

Mr. Khaja Pasha Shaik [*1], Mr. Mohammed Haroon[*2], Mr. Mohammed Anas Khan[*3], Mr. Syed Shahbaaz Hussain [*4],

[*1] Assistant Professor, Dept. of CSE-AIML, Lords Institute of Engineering and Technology
[*2, 3, 4] B.E Student Dept. of CSE-AIML, Lords Institute of Engineering and Technology
shaikkhaja@lords.ac.in[*1], haroonmohd922@gmail[*2], mohammedanask419@gmail.com*[3],
s.shahbaaz5hussain@gmail.com[*4]

Abstract: *CAPTCHAs are automated tests designed to distinguish between computers and humans, attacking programs, or other computerized agents that attempt to imitate human intelligence. The main intent of this research is to develop a method to crack CAPTCHA using a custom based convolutional neural network (CNN) model called CAP-SECURE. The proposed model aims to distinguish or tell websites about the weaknesses and vulnerabilities of the CAPTCHAs. The CAP-SECURE model is based on sequential CNN model and it outperforms the existing CNN architecture like VGG-16 and ALEX-net . The model has the potential to solve and explore both numerical and alphanumerical CAPTCHAs. For developing an efficient model, a dataset of 200000 CAPTCHAs has been generated to train our model. In this exposition, we study CNN based deep neural network model to meet the current challenges, and provide solutions to deal with the issues regarding CAPTCHAs. The network cracking accuracy is shown to be 94.67 percent for alpha-numerical test dataset. Compared to traditional deep learning methods, the proposed custom based model has a better recognition rate and robustness.*

Keywords: CAPTCHA, *CAP-SECURE, CNN*

INTRODUCTION

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a widely used tool for distinguishing humans from bots, commonly implemented in web applications to prevent spam, unauthorized data access, and other cyber threats. As internet technologies evolve, CAPTCHAs have become crucial in enhancing network security. Traditional CAPTCHAs typically involve distorted alphanumeric characters, while modern variations include image- and audio-based formats. Text-based CAPTCHAs remain most common, often enhanced with noise and distortions to increase complexity.

CAPTCHA recognition involves image preprocessing, character segmentation, feature extraction, recognition, and post-processing. Among these, segmentation and feature extraction significantly impact accuracy. Despite the robustness of Optical Character Recognition (OCR) systems, they struggle with distorted or handwritten scripts. In contrast, Convolutional Neural Networks (CNNs), a type of deep neural network, have demonstrated superior performance in image recognition tasks by learning features autonomously, reducing the need for manual preprocessing.

This paper proposes CAP-SECURE, a CNN-based CAPTCHA recognition model focusing on alphanumeric image CAPTCHAs. The aim is to explore vulnerabilities in CAPTCHA systems and improve their resilience against automated attacks.

LITERATURE SURVEY:

1. *On Random Field Completely Automated Public Turing Test To Tell Computers And Humans Apart Generation (2013)*
   Authors: Kouritzin, M. A.; Newton, F.; Wu, B.
   Proposes a CAPTCHA generation method using random field simulation and Gibbs resampling. Enhances resistance to OCR attacks by embedding word information into site probabilities and covariances, achieving ~95% human recognition accuracy.
2. *Captcha As Graphical Passwords – A New Security Primitive Based On Hard Ai Problems (2014)*
   Authors: Zhu, B. B.; Yan, J.; Bao, G.; Yang, M.; Xu, N.
   Introduces CaRP—a hybrid of CAPTCHA and graphical password. Designed to combat online guessing, relay, and shoulder-surfing attacks. Offers improved usability and defense against common vulnerabilities in graphical password systems.
3. *Recognition Of Captcha Characters By Supervised Machine Learning Algorithms (2018)*
   Authors: Bostik, Ondrej; Klecka, Jan

Compares neural networks, SVMs, k-NN, and decision trees for CAPTCHA recognition. Achieves over 89% accuracy across all methods, with variations mainly in training time, aiding selection of optimal algorithms.

4. *Accurate, Data-Efficient, Unconstrained Text Recognition With Convolutional Neural Networks (2018)*
Authors: Yousef, M.; Hussain, K. F.; Mohammed, U. S.
Proposes a fully convolutional neural network for efficient text recognition across domains like CAPTCHA, OCR, and scene text. Trained with CTC loss, the model performs well without recurrent layers and won ICFHR2018.
5. *I'm Robot: Deep Learning To Break Semantic Image Captchas (2016)*
Authors: Sivakorn, S.; Polakis, J.; Keromytis, A. D.
Analyzes Google's reCaptcha and semantic image CAPTCHAs. Demonstrates deep learning-based attacks with 70.78% accuracy, revealing weaknesses in risk analysis and proposing countermeasures for future CAPTCHA systems.
6. *Deep-Captcha: A Deep Learning Based Captcha Solver For Vulnerability Assessment (2020)*
Authors: Noury, Zahra; Rezaei, Mahdi
Proposes Deep-CAPTCHA, a CNN-based framework to assess the security strength of CAPTCHA systems. Highlights vulnerabilities by successfully solving various CAPTCHA formats using deep learning.
7. *Accurate, Data-Efficient, Unconstrained Text Recognition With Cnns (2018)*
Authors: Yousef, Mohamed; et al.
Introduces a fully convolutional, end-to-end architecture for text recognition across diverse domains. Trained with CTC loss, the model achieves state-of-the-art accuracy without requiring recurrent layers.
8. *Breaking Captchas With Convolutional Neural Networks (2017)*
Authors: Kopp, Martin; Et Al.
Demonstrates the effectiveness of CNNs in solving text-based CAPTCHAs with high success rates. Reveals the vulnerability of many commonly used CAPTCHA schemes to deep learning-based attacks.
9. *Captcha Image Generation Using Style Transfer Learning In Deep Neural Network (2019)*
Authors: Kwon, Hyun; et al.
Applies style transfer learning in deep neural networks to generate visually diverse CAPTCHA images. Enhances CAPTCHA complexity and variability to defend against automated solvers.
10. *Captcha Recognition Based On Deep Convolutional Neural Network (2019)*
Authors: Wang, J.; et al.
Utilizes a deep CNN for improved CAPTCHA recognition accuracy. Demonstrates that deeper networks can effectively learn features from distorted characters, outperforming traditional approaches

SYSTEM ANALYSIS

EXISTING SYSTEM:

Segmentation-based methods often struggle with distorted CAPTCHAs, resulting in limited accuracy. Similarly, conventional CNN architectures such as VGG-16 and AlexNet are not specifically optimized for CAPTCHA recognition, which affects their effectiveness in this domain. The TOD-CNN approach, while more advanced, still requires a separate segmentation step, adding complexity to the overall process.

Limitations of Existing Systems:

- Reduced accuracy with subtle or complex attacks
- Lower accuracy for complex CAPTCHAs.
- Slower processing for variable-length CAPTCHAs.
- Limited generalization across CAPTCHA types.

PROPOSED SYSTEM

CAP-SECURE a deep neural network is proposed to identify the CAPTCHAs. We create a deep neural network named CAP-SECURE using customized convolutional layers to fit our prerequisites.

Workflow:

Image Preprocessing Enhances Efficiency: The preprocessing pipeline includes image resizing (from 400×100 to 200×50), grayscale conversion, noise reduction, and normalization. These steps reduce data size and redundancy, speeding up training while maintaining recognition accuracy. One-Hot Encoding for Multi-Character Output: Since each CAPTCHA contains five alphanumeric characters, one-hot encoding is applied to represent each character (from 36 possible classes: 26 letters + 10 digits) efficiently, enabling accurate prediction and classification. Custom CNN Architecture: The model uses a sequential CNN with convolutional, batch normalization, and max-pooling layers. The network flattens feature maps before branching into five separate output heads, each predicting one character of the CAPTCHA. CNN Outperforms RNN for CAPTCHA: CNNs are preferred over RNNs in this approach due to faster processing and superior performance in image-based CAPTCHA recognition when well-designed

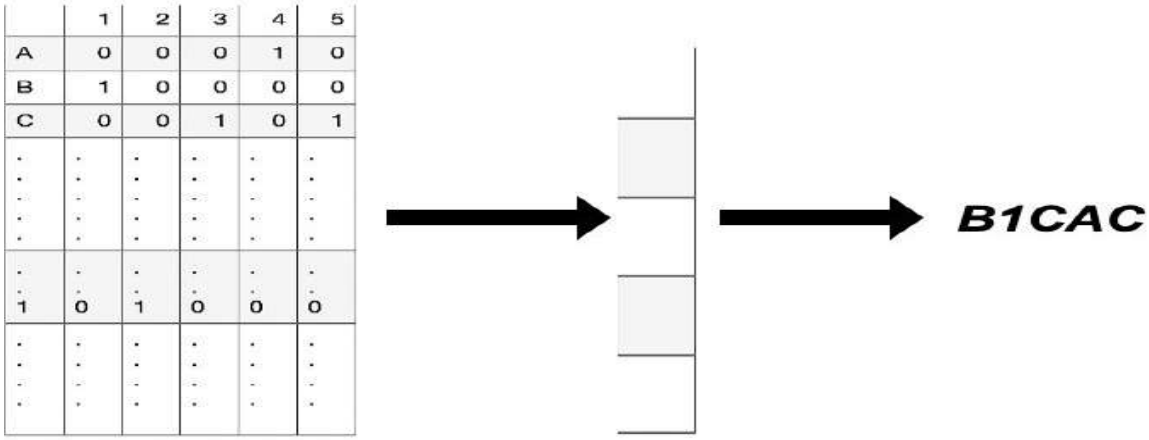
ADVANTAGES

- Faster Training – Image resizing and grayscale reduce processing time.
- High Accuracy – CNN architecture improves recognition performance.
- Multi-Char Support – One-hot encoding handles full CAPTCHA sequences.
- Efficient & Scalable – Works well with limited data and real-time use.

SYSTEM DESIGN

SYSTEM ARCHITECTURE

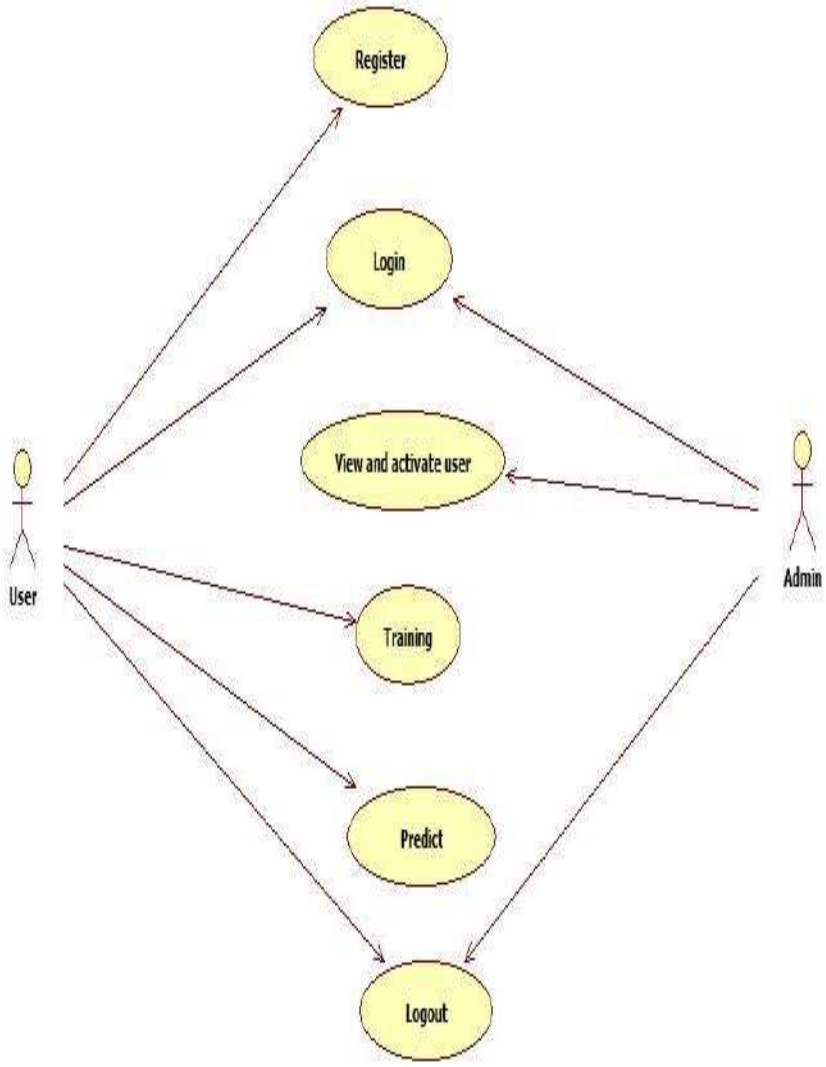
The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output

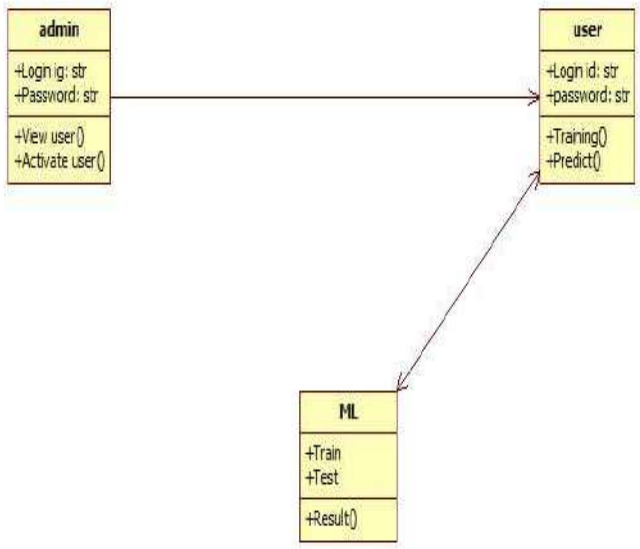


UML DIAGRAMS

1. Use Case Diagram – Workflow
Shows how users (e.g., admin) interact with the system to perform tasks like navigating screens, entering data, and accessing browser-compatible features.
2. Class Diagram – Workflow
Represents core classes (e.g., UserInterface, NavigationManager) along with their attributes, methods, and how they interact to ensure smooth user experience.
3. Object Diagram – Workflow
Illustrates runtime instances like userA, screen1, showing real-time interactions between UI elements and navigation logic.
4. Sequence Diagram – Workflow
Displays the flow of control in the system from user login → data entry → submission → feedback, including decision nodes like "Input valid?".5. Activity Diagram – Workflow
Illustrates the process flow from packet capture to attack detection, including decision points like “Is data sufficient?”
6. State Diagram – Workflow
Shows app states like Start, Navigating, Inputting, Submitting, and transitions triggered by user actions or system events.
7. Component Diagram – Workflow
Breaks the system into modules: GUI Layer, Backend Logic, Browser Handler, and Compatibility Layer, and shows how they communicate.
8. Deployment Diagram – Workflow

Maps software onto hardware: user system (browser), server (backend), and shows communication between interface, processing logic, and server responses.





IMPLEMENTATION

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

Input Design converts user-oriented input descriptions into a computer-based system to reduce errors and ensure accurate information for management. It involves creating user-friendly data entry screens to handle large volumes of data efficiently and accurately.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user.Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

SAMPLE CODE

The CAP-Secure system uses modular Python code integrated with Django, leveraging libraries such as NumPy, Matplotlib, OpenCV, and TensorFlow/Keras for training and deploying a custom Convolutional Neural Network (CNN) model. The process includes:

Data Preprocessing:

Captcha images are read in grayscale using OpenCV.

Images are resized and reshaped to a fixed dimension (50x200x1).

Pixel values are normalized between 0 and 1.

Captcha labels are converted to one-hot encoded vectors for each character (up to 5 characters).

Feature Representation:

The CNN model uses convolutional, max pooling, batch normalization, and dropout layers to extract and learn visual features from the CAPTCHA images.

The architecture includes 5 output branches to simultaneously predict all characters in the CAPTCHA.

Model Training:

The model is compiled using categorical cross-entropy as the loss function and the Adam optimizer.

The training set is split 80/20 for validation using ~200,000 synthetic CAPTCHAs.

Training results are visualized through accuracy and loss graphs for each output layer.

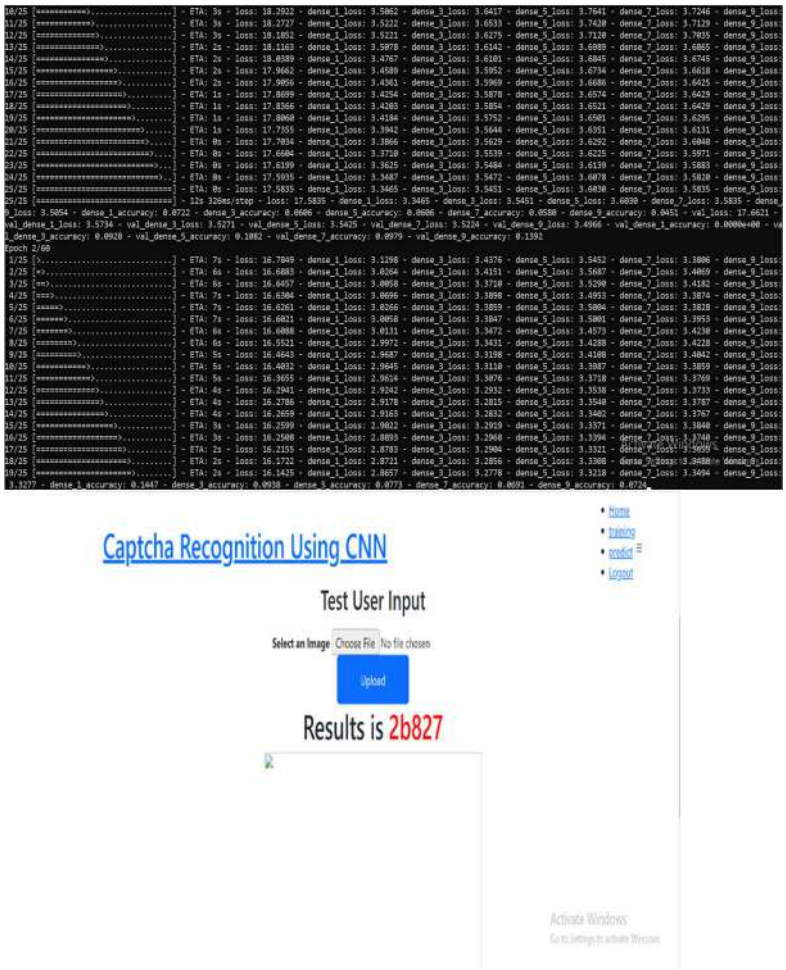
Model Serialization:

After training, the best-performing model is saved using model.save("model.h5") for later use in predictions.

Prediction:

A separate script loads the trained .h5 model.The uploaded CAPTCHA image is preprocessed in real-time, passed through the CNN model, and each character is predicted using np.argmax.

The final CAPTCHA text is decoded and returned to the user interface.



IMPLEMENTATION

The CAP-Secure system implements a complete CAPTCHA recognition pipeline using Django and a custom CNN model. CAPTCHA images are pre-processed converted to grayscale, resized, normalized, and one-hot encoded.

The dataset is split into training (80%) and testing (20%) sets. A custom CNN model with convolutional, pooling, dropout, and batch normalization layers is trained to predict five-character CAPTCHAs.

The trained model, achieving 94.67% accuracy, is saved for future use. A prediction module loads this model to classify new CAPTCHA images uploaded by users. This implementation enables end-to-end CAPTCHA cracking with high accuracy and efficiency

SOFTWARE TESTING

Software testing validated the end-to-end functionality of the CAPTCHA recognition system, from data preprocessing to prediction.

Unit Testing ensured individual modules like registration, data upload, and image processing functioned correctly.
Integration Testing verified seamless interaction between Django components and the CNN model.
Model Evaluation assessed the custom CNN's accuracy (94.67%) on alphanumeric CAPTCHA datasets.
Functional Testing confirmed the performance of key features such as login, dataset handling, training, and prediction.
Performance Testing proved the system can handle large datasets (200,000 CAPTCHAs) efficiently.
The system showed stable, accurate results throughout, confirming its readiness for deployment.

RESULT ANALYSIS

It demonstrated and evaluated multiple machine learning algorithms based on accuracy, speed, and suitability for real-time intrusion detection. Metrics such as Precision, Recall, F1-Score, Accuracy, training time, and prediction latency were used. The CAP-Secure system evaluated multiple deep learning models, particularly a custom CNN, for CAPTCHA recognition. The evaluation focused on accuracy, training efficiency, and prediction speed.

Training Time:

Custom CNN:
Required moderate training time (~60 epochs).
Efficient for large-scale CAPTCHA datasets (200,000 images)
Prediction Time:
CNN Model:
Fast and accurate prediction (<0.05 seconds per image)
Suitable for real-time CAPTCHA decoding
Performance Metrics:
Outperformed standard CNNs like VGG-16 and AlexNet.
Handled alphanumeric CAPTCHAs without segmentation.
Overall comparison:

The custom CNN architecture achieved a high recognition rate and computational efficiency. Compared to traditional CNNs, it offers better performance for real-time CAPTCHA cracking and is ideal for deployment in security-testing environments

FUTURE SCOPE & CONCLUSION

FUTURE SCOPE

Expand CAPTCHA Coverage: Train the model on diverse CAPTCHA types, including image, audio, and interactive formats.
Improve Model Architecture: Explore advanced CNNs, hybrid models with RNNs or transformers, and ensemble methods for better accuracy.
Enhance Robustness: Use larger, varied datasets and techniques like data augmentation and adversarial training to increase resilience.
Optimize for Deployment: Improve speed and efficiency for integration into real-world web apps, including API and tool development.
Advance Security Research: Investigate attack-resistant CAPTCHA designs and explore user-friendly alternatives for secure authentication

CONCLUSION

We designed a custom based CNN model to crack alphanumeric CAPTCHAs to find out the vulnerability of CAPTCHAs in high trafficking websites which use common CAPTCHA generators. Using of two batch normalization layers made the model more stable and faster. It helped us achieve an accuracy of 94.67 percent which was way better than the existing CNN models like ALEXNET and VGG. Although our model worked well for random CAPTCHAs but few CAPTCHAs made work difficult and challenging for CAP-SECURE net- work. Those misclassified CAPTCHAS were studied to help generate robust CAPTCHAs so that it is difficult for bots to crack CAPTCHAs.
As a potential pathway for future extension we recommend solving CAPTCHAs of variable length. It can also be extended to CAPTCHAs of different languages. Furthermore, image CAPTCHAs could also be explored in the future.

BIBLIOGRAPHY

1. M. A. Kouritzin, F. Newton, and B. Wu, "On random field completely automated public turing test to tell computers and humans apart genera- tion," IEEE Transactions on Image Processing, vol. 22, no. 4, pp. 1656 – 1666, 2013.
2. B. B. Zhu, J. Yan, G. Guanbo Bao, M. Maowei Yang, and N. Ning Xu, "CAPTCHA as graphical passwords-a new security primitive based on hard AI problems," IEEE Transactions on Information Forensics and Security, vol. 9, no. 6, pp. 891 – 904, 2014.
3. Bostik, Ondrej, and Jan Klecka. "Recognition of CAPTCHA characters by supervised machine learning algorithms." IFAC-PapersOnLine 51, no. 6 (2018), pp. 208 – 213.
4. Mohamed, Khaled F. Hussain, and Usama S. Mohammed. "Ac- curate, data-efficient, unconstrained text recognition with convolutional neural networks." arXiv preprint arXiv:1812.11894 (2018).
5. Sivakorn, Suphannee, Jason Polakis, and Angelos D. Keromytis. "I'm robot: deep learning to break semantic image captchas." In 2016 IEEE European Symposium on Security and Privacy (Euro S&P), 2016, pp. 388 – 403.
6. Von Ahn, Luis, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. "recaptcha: Human-based character recognition via web security measures." Science 321, no. 5895, 2008, pp. 1465 – 1468.
7. M. Belk, C. Fidas, P. Germanakos, and G. Samaras, "Do human cognitive differences in information processing affect preference and performance of CAPTCHA?" International Journal of Human-Computer Studies, vol. 84, 2015, pp. 1 – 18.
8. I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud and V. Shet, "Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks," Computer Science, 2013.
9. M. Jaderberg, A. Vedaldi, and A. Zisserman, "Deep features for text spotting," European conference on computer vision, Springer, Cham, 2014, pp. 512 – 528.
10. K. Chellapilla and P. Y. Simard, "Using machine learning to break visual human interaction proofs (HIPs)," Advances in Neural Information Processing Systems, 2004, pp. 265 – 272.
11. J. Yan and A. S. El Ahmad, "Breaking visual CAPTCHAs with naive pattern recognition algorithms," in Proceedings of the 23rd Annual Computer Security Applications Conference, Miami Beach, FL, USA, December 2007, pp. 279 – 291.
12. H. Gao, J. Yan, F. Cao et al., "A simple generic attack on text captchas," in Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA, February 2016, pp. 220 – 232.
13. G. Mori and J. Malik, "Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA," Computer Vision and Pattern Recognition, Proc. IEEE Computer Society Conference, Vol. 1, IEEE, 2003, pp.I-I.
14. K. Qing and R. Zhang, "A multi-label neural network approach to solving connected CAPTCHAs," in Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (IC- DAR), IEEE Computer Society, Kyoto, Japan, November 2017, pp. 1313 – 1317.
15. B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recog- nition," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 39, no. 11, pp. 2298 – 2304, 2016.
16. F.-L. Du, J.-X. Li, Z. Yang, P. Chen, B. Wang, and J. Zhang, "CAPTCHA recognition based on faster R-CNN," Intelligent Computing ;eories and Application, pp. 597 – 605, 2017.
17. D. Lin, F. Lin, Y. Lv, F. Cai, and D. Cao, "Chinese character CAPTCHA recognition and performance estimation via deep neural network," Neu- rocomputing, vol. 288, pp. 11 – 19, 2018
18. F. H. Alqahtani and F. A. Alsulaiman, "Is image-based CAPTCHA secure against attacks based on machine learning? an experimental study," Computers & Security, vol. 88, 2019.
19. 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.
20. Bostik, Ondrej, and Jan Klecka. "Recognition of CAPTCHA characters by supervised machine learning algorithms." IFAC-PapersOnLine 51, no. 6 (2018), pp. 208 – 213.
21. Yousef, Mohamed, Khaled F. Hussain, and Usama S. Mohammed. "Ac- curate, data-efficient, unconstrained text recognition with convolutional neural networks." arXiv preprint arXiv:1812.11894 (2018).
22. Karthik, CHBL-P., and Rajendran Adria Recasens. "Breaking microsofts CAPTCHA." Technical report (2015).

23. Kopp, Martin, Matej Nikl, and Martin Holena. “Breaking captchas with convolutional neural networks.” ITAT 2017 Proceedings (2017): pp. 93 – 99.

24. Zhao, Nathan, Yi Liu, and Yijun Jiang. “CAPTCHA Breaking with Deep Learning,” (2017