International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362
IJARST

Analysis and Recognition of CAPTCHAs with the Use of a Custom-Based CNN Model – Capsecure

[1]Mrs. Ch. Prashanthi, [2]Pasalapudi Pavan Kumar, [3]P.S.K.B. Subrahmanyam, [4]Vadapalli Nahum,

[1]Assistant Professor, Dept. of CSE, Rajamahendri Institute of Engineering & Technology, Bhoopalapatnam, Near Pidimgoyyi, Rajahmundry, E.G. Dist. A.P. 533107.

[2,3,4] Students, Dept. of CSE, Rajamahendri Institute of Engineering & Technology, Bhoopalapatnam, Near Pidimgoyyi, Rajahmundry, E.G. Dist. A.P. 533107.

Abstract—

CAPTCHAs are automated tests designed to distinguish between computers and humans, attacking programs, or other computerized agents that attempt to imitate human intel- ligence. 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.

I. INTRODUCTION

The Completely Automated Public Turing Test for Human-Computer Assignment, more often known as CAPTCHA, is a system that uses automated testing to differentiate between people and robots [2]. Applications like information security and network protection have made extensive use of it. Concerns about network security are only becoming worse as internet technology develops further. In order to protect the privacy of users and their online services from a variety of cyber threats, assaults, and other intrusions, CAPTCHAs are used on the internet [1]. Quite a few prominent mainstream websites have started utilizing it recently. To be more precise, these assaults frequently result in scenarios where computer programs take the role of people. Computer programs attempt to automate services in order to send a flood of unwanted emails, get access to databases, or manipulate online pools. The integration of CAPTCHA recognition with AI and image processing is crucial to the advancement of AI technology; it has several applications in areas such as optical character recognition, handwriting recognition, license plate recognition, and more. The most common kind of CAPTCHA relies on visual cues such as strings of numbers or alpha-numerical characters, audio, or image sets. At the same time, CAPTCHAs that rely on text are widely utilized in practice. The types and instances of alpha-numerical CAPTCHAs are shown in Figure 1. Numerous techniques exist for enhancing CAPTCHAs with the addition of efficient noise and disruptions, making them more challenging [20], [4]. Adding other kinds of noise, such as zigzag lines across the alphanumeric CAPTCHA, can improve the security of CAPTCHA methods. Figure 2 shows a few examples of distorted CAPTCHAs caused by noise. There are two types of CAPTCHAs that are gaining popularity: text-based and image-based. The purpose of these CAPTCHAs is to test the user's ability to recognize a certain object from a set of example photos, which may include traffic lights, automobiles, street signs, landscapes, or sculptures. See Figure 3 for a picture of a CAPTCHA in action. Common procedures for traditional CAPTCHA detection include preprocessing the picture, segmenting the characters, recognizing them, extracting features, and post-processing. Segmentation and feature extraction are crucial steps in these models that determine their accuracy and efficiency. This page primarily aims to shed light on commonly used CAPTCHA graphics, which are made up of random English letters and numbers. On top of that, making them is a breeze. Also, another use of CAPTCHA that deserves note is in OCR (Optical Character Recognition). While current optical character recognition algorithms are very reliable, they do not fully account for a variety of hand-written scripts or weakened handwritings [6]. Convolutional neural networks (CNNs) have recently outperformed more conventional approaches to picture identification, demonstrating its potential as a type of deep neural networks. Traditional methods of pixel point extraction and template matching are limited to detecting basic CAPTCHAs. One major benefit of CNN over more conventional pattern recognition methods is its ability to actively learn features without the need for artificial design. Without the issues of data preparation, the retrieved picture characteristics are very expressive. There is a dearth of research on the recognition impact of complicated CAPTCHA, despite the fact that CNN has produced some findings [7]. Then, in order to detect the CAPTCHAs, a CNN algorithm is suggested by [8].

In this research, we present CAP-SECURE, a CNN-based picture CAPTCHA recognition model. Picture CAPTCHAs include both English characters and random numbers. You can make it with little effort, and raw power is tough to overcome. The purpose of this study is to investigate CAPTCHA recognition and identify its potential flaws. So that these websites can withstand intelligence assaults and bots, there has been an attempt to enhance CAPTCHA technology and create more robust CAPTCHAs.

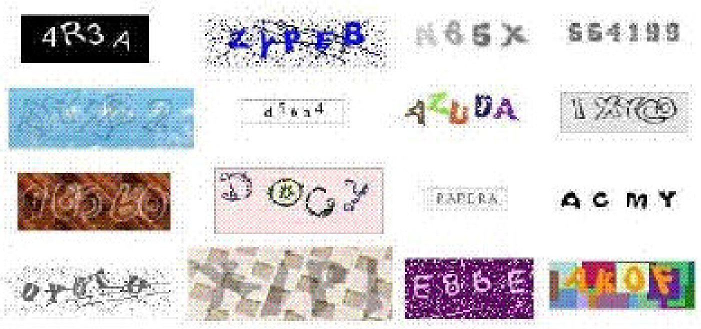


Fig. 1: Examples of various alpha-numeric CAPTCHAs

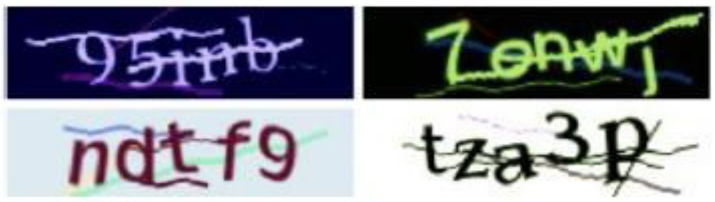


Fig. 2: Examples of five-digit distorted CAPTCHAs

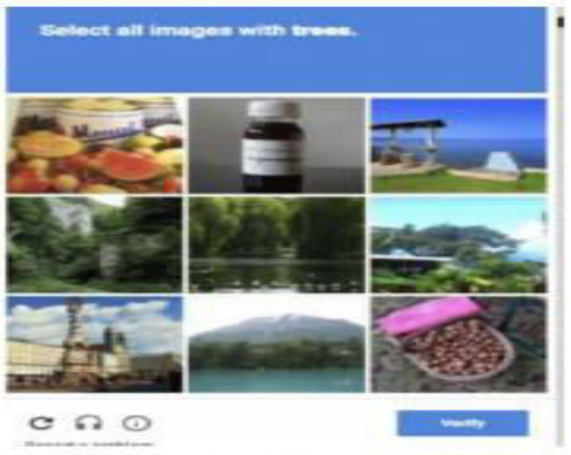


Fig. 3: Image based Captcha

Here is the paper's outline: In Section II, we take a look at the written works that have contributed to our understanding of CAPTCHA and its evolution. In Section III, we lay out the specifics of our suggested approach. Section IV discusses the experimental outcomes. Section V concludes with a few last thoughts.

II. BACKGROUND

In this part, we will go over what is known about CAPTCHA and what has happened recently in the area.
One popular approach to CAPTCHA recognition is segmentation recognition. The standard procedure for locating a specific region in a picture that contains a number or character, then

segmenting the image and identifying the individual characters [9]. Take the authors of [10] as an example; they demonstrated how character segmentation in CAPTCHA may greatly enhance recognition accuracy. This type of CAPTCHA has been deciphered by the developers of [11] using pixel calculations. The same author went on to successfully segment it using the projection method, achieving a success rate of 60% in cracking and an accuracy of up to 90% in the process. Then, in [12], the character strokes are extracted using the Gambor filtering approach. They discovered the best combination for character segmentation using the graph search technique, and CAPTCHA breaking accuracy was 77%. The use of a shape context technique for CAPTCHA picture identification is covered in [13]. Researchers have taken a shine to deep learning-based CAPTCHA identification systems in the past several years. For text-based CAPTCHA recognition without segmentation, the authors of [14] have suggested a multilabel CNN. Character distortion and complicated CAPTCHA have both been improved upon. A convolutional recurrent neural network was suggested by the authors of [15] to achieve general CAPTCHA recognition by combining CNN with RNN. In [16], you can find the fast region-based CNN for overall recognition that works better with CAPTCHA sequences of varying lengths.

In order to learn CAPTCHA's stroke and character properties, the creators of [17] employed a convolutional neural network. The accuracy of CAPTCHA detection with rotation, distortion, and background noise has been significantly enhanced. Deep neural networks can efficiently increase classification and recognition compared to older approaches, and they have greater learning abilities overall [18]. To increase the complexity of CAPTCHA patterns' security, writers in previous publications have proposed adding noises such line cross noise or point-based scattering noise [20] and [21]. From [22] to [25], we may find approaches that are based on CNNs. For improved outcomes, Kwon et al. [26] have used CNN with transfer style. In contrast to DenseNet [28], a CNN with a few tweaks was analyzed in [19]. An earlier study by Garg and Pollett [29] investigated the use of a deep neural network trained in Python to decipher fix-lengthening CAPTCHAs. Two Convolutional Maxpool layers are being investigated for the network. Stochastic Gradient Descent (SGD) with Nesterov momentum, which shows recurrent layers instead of fundamental dense layers, is the basis of their concept. Dense layers are also shown to be more accurate. Afterwards, a web-browser-based system for illuminating photo CAPTCHAs was proposed by Sivakorn et al. [5]. The problem has been solved by the authors of [25] using a CNN. After two thick layers, they utilised three convolutional ones. In addition, a technique to decrease the size of the necessary training dataset has been employed by the writers. A technique based on the local minimum and minimum projection values was presented by Lu Wang et al. [30] for the identification of merged characters. In addition, one can eliminate implausible terms by using a dictionary. To overcome CAPTCHAs, for instance, Mori and Malik [18] suggested a strategy that involves a dictionary containing all 411 words.

III. PROPOSED METHOD

Deep learning has recently grown in significance among the scientific community. Among its many successful applications are natural language processing, target detection, picture identification, and speech recognition.
One major benefit of deep learning is its ability to actively learn characteristics without the need for artificial design. Websites with a lot of traffic utilize CAPTCHAs, which are easy targets for bot assaults that attempt to decipher them automatically. A deep neural network called CAP-SECURE is suggested to detect CAPTCHAs based on the aforementioned observations and ideas. Using specialized convolutional layers that meet our requirements, we construct a CAP-SECURE deep neural network. Presented here is the comprehensive approach of managing, recognizing, and dividing the alphanumerical Captcha images. It is necessary to pre-process the picture, encode the output, and design the network in order for the suggested approach to directly use images as input information.
Section A. Prefixing Reduced picture size, greyscaled images, one-hot encoding, and noise reduction filtering are all part of the pre-processing procedures that raise the network's accuracy. This study makes use of picture data that was originally 400 × 100 pixels in size. According to our findings, the model's performance remains unchanged when the picture size is reduced to 200 × 50 pixels, yielding almost identical outcomes. The training procedure can become significantly faster with this size reduction since it decreases the data without substantial deduction in the data entropy. Greyscale image processing was another preprocessing step. Greyscale versions of the captcha pictures were created. With this method, we may decrease the data size without sacrificing detection accuracy. Both the prediction and training processes are made easier, and the quantity of redundant data is decreased, thanks to this. Since color is not essential for CAPTCHAs based on alphanumeric text, converting the source image to greyscale has no effect on the outcomes. Normalization was the last preprocessing method we employed. Dividing the pixel values by the highest value a pixel may take—255 in our case—is how a picture is normalized. In order to make the model run faster, normalization is applied to make sure that each input parameter has the same distribution of data. The training of the model becomes more efficient as a result. A. One-Time Encoding
When dealing with CAPTCHAs, the number of classes varies depending on the quantity of alphabetical letters and numerical digits, unlike to other classification problems where several classes are to be predicted. The duration of the CAPTCHAs is another factor. For this project, we have used a five-character CAPTCHA. Depending on the amount of classes that need to be discovered, this leads to exponential development. For a CAPTCHA challenge involving five number digits, we have thus far explored around 100,000 distinct permutations. Hence, we have to encode the output data so it may fit into just one neural network. Our data set includes one hot-encoding that was created using the image CAPTCHA module in Python. Machine learning algorithms often use one hot encoding to transform category data into binary numbers. Using 26 letters and 10 numerals has increased the prediction accuracy. Every one of the 36 possible

combinations can be a CAPTCHA unit. Take the letter B as an example; it would be encoded with 1 and the remaining 35 alternatives would be encoded with 0. The remaining four CAPTCHA units are treated in the same way. The input data is categorical in nature, thus it's important to employ one-hot encoding to ensure accurate prediction. Hence, our model is able to make more accurate predictions with the aid of encoding. C. ONE HOT ENCODING ALGORITHM: This is the algorithm that is utilized for one-hot encoding. First Method

```
$targs \leftarrow np.zeros((5, num_symbols))$ ;
$pictarget \leftarrow 5$ ;
while $j, l \neq pictarget$ do
    $ind \leftarrow symbols.find(l)$ ;
    $targs[j, ind] \leftarrow 1$ ;
end
$X[i] \leftarrow img$ ;
$y[:, i] \leftarrow targs$ ;
```

Section D: Model Architecture Though RNN is one potential solution for CAPTCHA prediction and cracking. Since sequential CNN models are both quicker and more effective than RNNs when trained correctly, we have employed them in our work to break CAPTCHA. Figure 5 shows the architecture of the network that we have suggested. A convolutional layer with 16 input neurons, the ReLU activation function (1), and 3×3 Kernels are the initial components of the network. After this layer, there is a 2 × 2 Max-Pooling layer. Afterwards, there are two sets of convolutional, batch normalisation, and Max-Pooling layers. Each set has 32 neurons and uses the ReLU activation function (1). It should be mentioned that the "same" padding parameter is used by all the convolutional layers. Following these layers, a flatten layer is used to merge all the 2-dimensional arrays that are collected from the pooled feature maps into one long continuous linear vector. This flatten layer now splits into five separate branches, one for each of the five alphanumeric characters used in the CAPTCHA. There is a dense-dropout-dense layer at the base of every branch. With 64 neurons, the network is ready to go. There are two dense layers: one with an activation function called ReLU and another with an activation function called

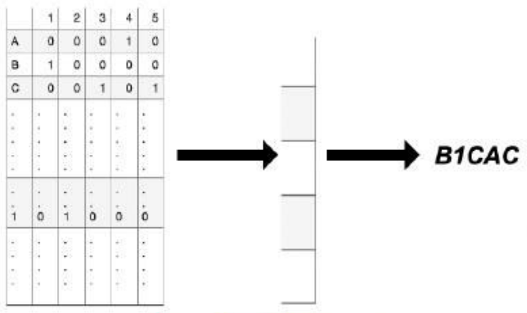


Fig. 4: One-hot encoding

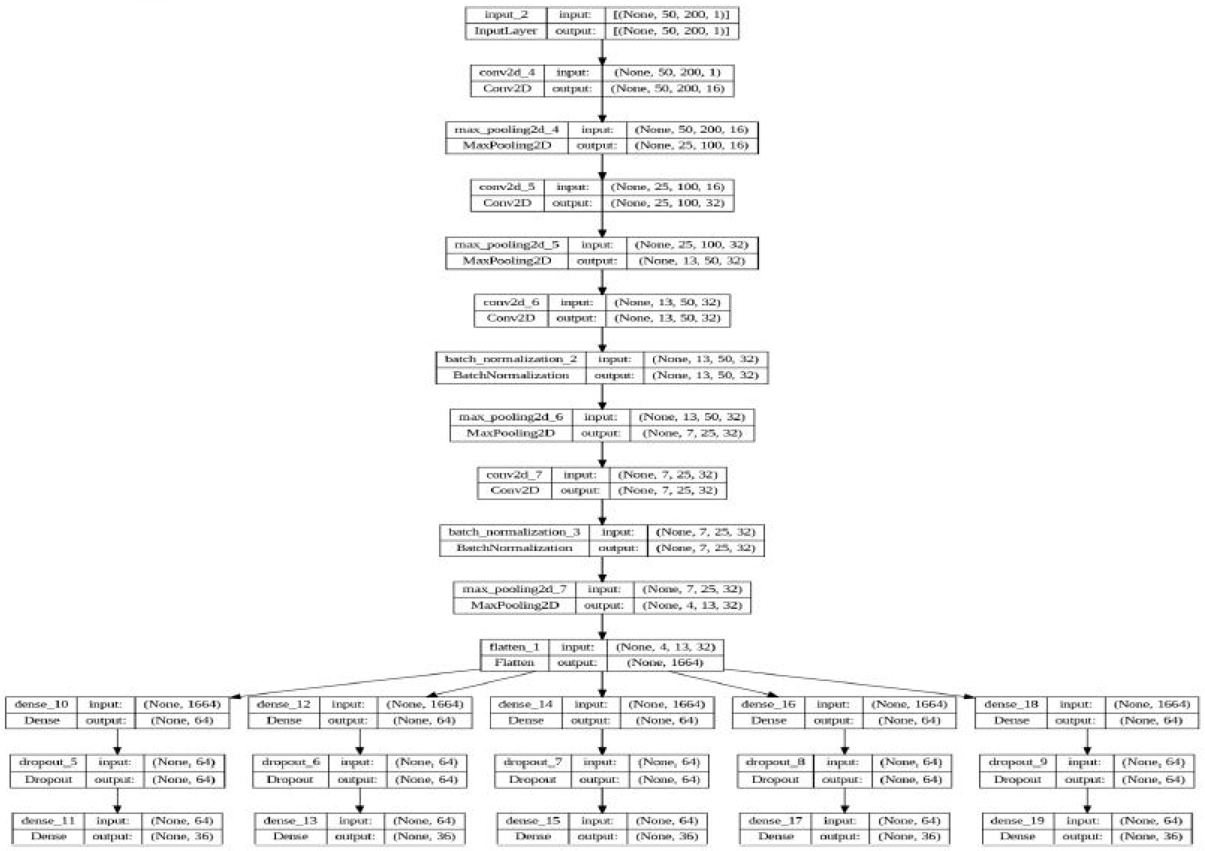

input_2 | InputLayer | input: [(None, 50, 200, 1)] | output: [(None, 50, 200, 1)]
conv2d_4 | Conv2D | input: (None, 50, 200, 1) | output: (None, 50, 200, 16)
max_pooling2d_4 | MaxPooling2D | input: (None, 50, 200, 16) | output: (None, 25, 100, 16)
conv2d_5 | Conv2D | input: (None, 25, 100, 16) | output: (None, 25, 100, 32)
max_pooling2d_5 | MaxPooling2D | input: (None, 25, 100, 32) | output: (None, 13, 50, 32)
conv2d_6 | Conv2D | input: (None, 13, 50, 32) | output: (None, 13, 50, 32)
batch_normalization_2 | BatchNormalization | input: (None, 13, 50, 32) | output: (None, 13, 50, 32)
max_pooling2d_6 | MaxPooling2D | input: (None, 13, 50, 32) | output: (None, 7, 25, 32)
conv2d_7 | Conv2D | input: (None, 7, 25, 32) | output: (None, 7, 25, 32)
batch_normalization_3 | BatchNormalization | input: (None, 7, 25, 32) | output: (None, 7, 25, 32)
max_pooling2d_7 | MaxPooling2D | input: (None, 7, 25, 32) | output: (None, 4, 13, 32)
flatten_1 | Flatten | input: (None, 4, 13, 32) | output: (None, 1664)
dense_10 | Dense | input: (None, 1664) | output: (None, 64)
dense_12 | Dense | input: (None, 1664) | output: (None, 64)
dense_14 | Dense | input: (None, 1664) | output: (None, 64)
dense_16 | Dense | input: (None, 1664) | output: (None, 64)
dense_18 | Dense | input: (None, 1664) | output: (None, 64)
dropout_5 | Dropout | input: (None, 64) | output: (None, 64)
dropout_6 | Dropout | input: (None, 64) | output: (None, 64)
dropout_7 | Dropout | input: (None, 64) | output: (None, 64)
dropout_8 | Dropout | input: (None, 64) | output: (None, 64)
dropout_9 | Dropout | input: (None, 64) | output: (None, 64)
dense_11 | Dense | input: (None, 64) | output: (None, 36)
dense_13 | Dense | input: (None, 64) | output: (None, 36)
dense_15 | Dense | input: (None, 64) | output: (None, 36)
dense_17 | Dense | input: (None, 64) | output: (None, 36)
dense_19 | Dense | input: (None, 64) | output: (None, 36)

Fig. 5: Architecture of the proposed CAP-SECURE Network

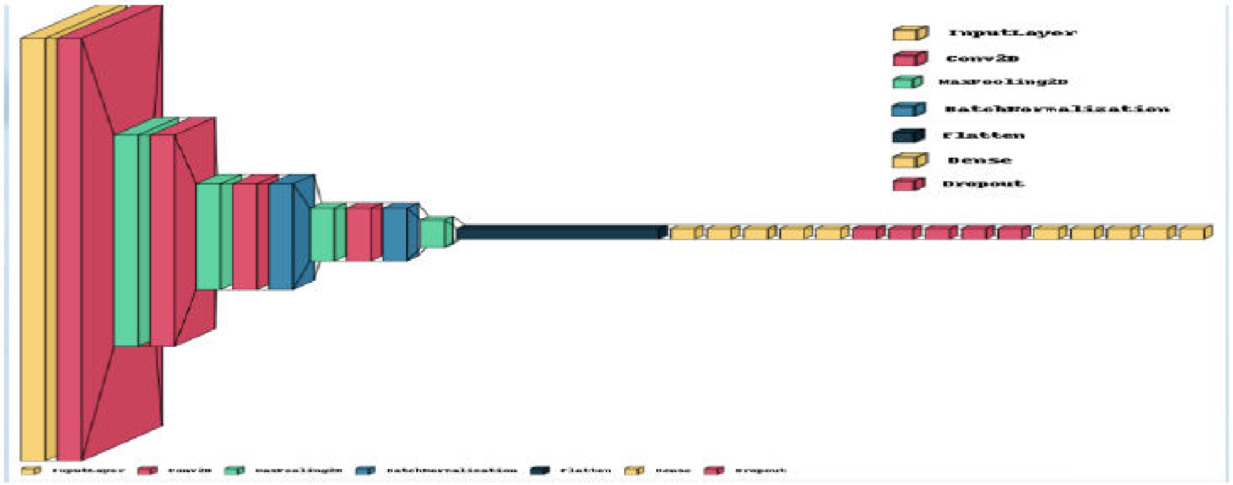


Fig. 6: 3D view of the proposed CAP-SECURE Network

It is a Sigmoid layer (2). Set to 0.5, or 50%, is the dropout rate for the dropout layer. The mathematical expressions of the ReLU (f(x)) and the Sigmoid activation (σ(x)) functions are as

$$f(x) = \max(0, x), \qquad (1)$$

and

$$\sigma(x) = \frac{1}{1+e^{-x}} \qquad (2)$$

in that order. Since we must compare both binary matrices together, the Binary cross entropy serves as the loss function (3) of the proposed network.

$$\sum_{i=1}^{N} -(y_i \log(p(x_i)) + (1-y_i)\log(1-p(x_i)), \qquad (3)$$

in where xi and yi are the input CAPTCHA for the ith sample, and N is the number of samples. The label might be either 0 or 1, therefore only one portion of the equation would be active, hence the binary cross entropy is used.
Equations (4) through (8) outline our model, which makes use of the ADAM optimizer. Here, m(t) and v(t) stand for the sum of squared previous gradients and the aggregate of gradients at time T, respectively.

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1)[\delta(L)/\delta(w_t)], \qquad (4)$$

and

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2)[\delta(L)/\delta(w_t)]^2 \qquad (5)$$

in that order. In this context, β1 and β2 are adjustable constants called moving average parameters, δ(L)/δ(wt) is the optimizing function's gradient, and t is the learning iteration. Here are the momentary values of m and v as determined in Equations (6) and (7):

$$\hat{m}_t = m_t/(1-(\beta_1)^t). \qquad (6)$$

$$\hat{v}_t = v_t/(1-(\beta_2)^t). \qquad (7)$$

The best value of the function is finally obtained by computing the value of θt using equation (8). You can find ˆmt and ˆvt by referring to equations (6) and (7). In our method, the step size, which is also called the learning rate, is set at p =.0001.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\hat{m}_t. \qquad (8)$$

A decent amount of time may be spent training the network, which is why ADAM optimizer is used. It converges quicker than SGD and produces superior outcomes. Figure 7 compares the outcomes. Our model was trained for 200 epochs with 32-batch sizes after many tests. It is evident from Figure 7 that the network continues to exhibit satisfactory convergence even after 100 epochs. Therefore, it appears that 200 epochs are enough to ensure the network's steady performance. We used the Python Image Captcha Library to train our model on 200,000 randomly generated CAPTCHAs after we developed the model mentioned above [31]. Some of the numerical CAPTCHAs that are created at random and have a set length of five digits are shown in Figure 2.

IV. EXPERIMENTAL RESULTS

Section A: Analyzing Performance Our model is compared to several well-known CNN models and previous research in this field in this section. Table I displays the comparative findings, while subsequent talks focus on the methods' specifications. Our method relies on convolutional neural networks (CNNs), which include

TABLE I: Accuracy metric of the dataset for each digit and the complete CAPTCHA as a set of 5 integrated digits.

	Train	Test
Digit1	99.4%	99.4%
Digit2	98.9%	94.3%
Digit3	98.7%	94.3%
Digit4	97.03%	94.3%
Digit5	98.19%	97.9%
CAPTCHA	96.45%	94.67%

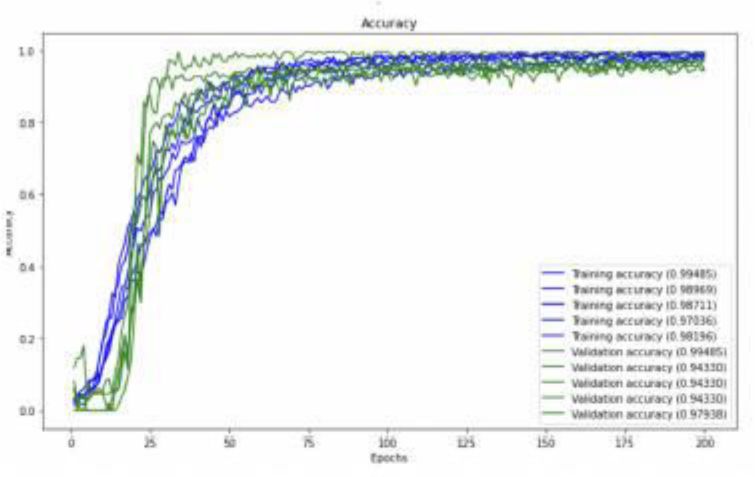
TABLE II: Loss and Accuracy of the Model

	LOSS	ACCURACY
Train	.023	96%
Test	.095	94.67%

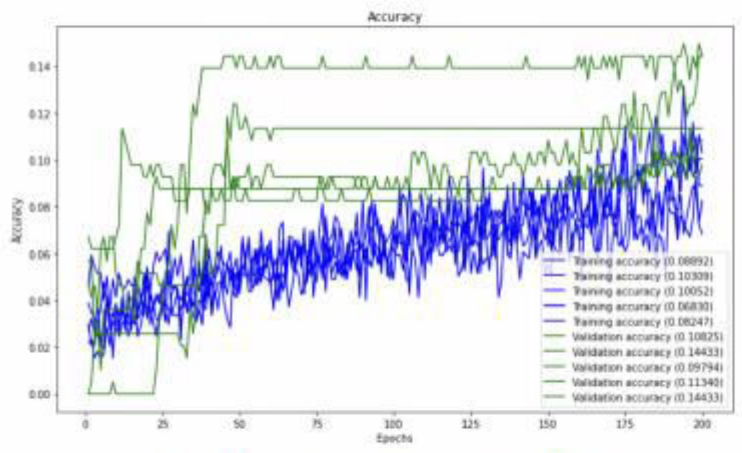
(2) Convolutional BatchNormalization-MaxPool layers, followed by (2) Convolutional-MaxPool pairs.

After going over a very flat stratum, it splits into five different branches. Lastly, Adam optimizer is used to train the network. Batch normalization allowed us to rescale the data to a mean of 0 and standard deviation of 1 for our network, making it quicker and more reliable. This eliminates the negative impact of the internal covariance shift and helps us attain a fixed distribution of inputs. Our model's predictions for CAPTCHAs extracted from random websites are displayed in Figure 8. However, we created a separate network that can solve both numerical and alphanumerical CAPTCHAs, as many current algorithms are compatible with both. We used 200,000 alphanumerical CAPTCHAs to train the network. We evaluated our performance against two well-known CNN architectures, VGG and Alex-net. The training accuracy for VGG-16 was 98.4 percent, and the testing accuracy was 87.1 percent. Alex-Net achieved 95% accuracy during training and 85% accuracy during testing. Alex-Net and VGG-16 are both surpassed by our model. Just like TOD-CNN [10], which uses a CNN model trained on a 60,000-item dataset, has also employed segmentation techniques to detect the characters. The approach segments the picture and its characters using a TensorFlow Object Detection (TOD) methodology. Its accuracy was 92.37 percent. Part B: Analyzing Vulnerabilities We sketched out a few suggestions that might help us develop more robust CAPTCHAs after carefully visualizing the misclassified ones. It is possible to make the CAPTCHA generator less susceptible by paying attention to a few key elements. The model's misclassified CAPTCHAs would be easy for a normal person to spot. It is possible to make more resilient CAPTCHAs by keeping a few things in mind.

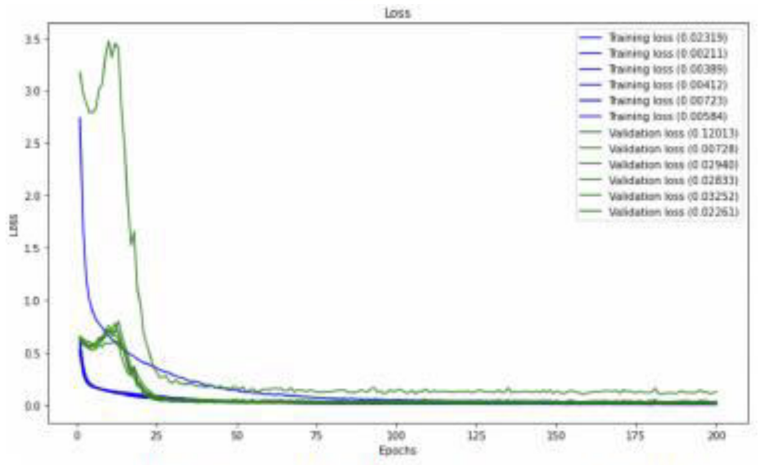
1) The grey scale intensity was low in 86% of the misclassified CAPTCHAs compared to the average intensity. 2) The numbers 8 and 3 were the cause of misclassification in 58% of the instances. Thirdly, a 12 degree or greater rotation was applied to the numbers or letters in 80% of the cases. 4) The pairs 1 and l, as well as g and 8, were frequently confused.



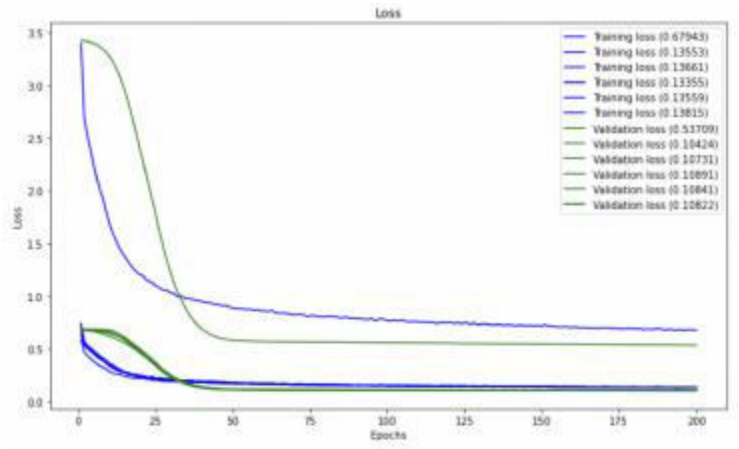
(a) Character accuracy using ADAM



(b) Character accuracy using SGD



(c) Training and validation loss using ADAM



(d) Training and validation loss using SGD

Fig. 7: Comparison of results between ADAM and SGD

To make CAPTCHAs more secure and resistant to machine and bot attacks, we recommend adding the numerals 8, 3, and the characters g and l. Because of this, machines will have a harder time deciphering CAPTCHAs that humans can readily solve.

V. CONCLUSION AND EXTENSION

If we wanted to know how susceptible the CAPTCHAs on popular websites that employ generic CAPTCHA generators are, we built a custom-based CNN model to decipher alphanumeric CAPTCHAs. The model's stability and speed were both improved by using two batch normalisation layers. It aided in the attainment of precision

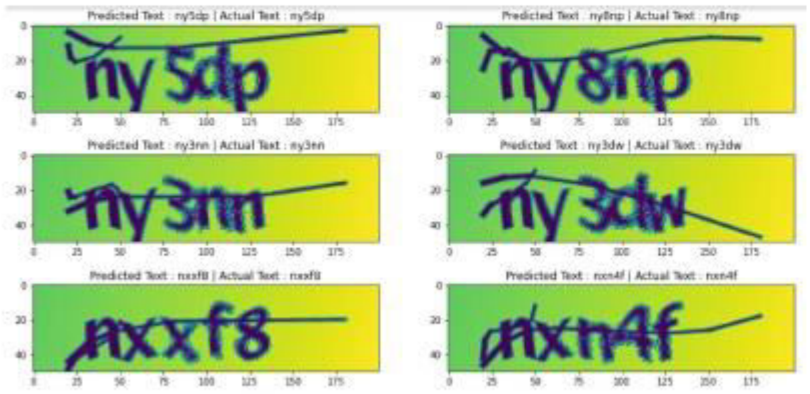


Fig. 8: CAP-SECURE model predicting CAPTCHAs taken from random websites.

of 94.67%, significantly outperforming state-of-the-art CNN algorithms such as ALEXNET and VGG. The CAP-SECURE network had a tough time with the few CAPTCHAs, even if our model performed well with random CAPTCHAs. In order to make CAPTCHAs that are tough for bots to break, researchers looked into such misclassified CAPTCHAs. We suggest solving CAPTCHAs of varying lengths as a possible future expansion avenue. Various language CAPTCHAs may also be added to it. Additionally, future research may potentially investigate the use of image-based CAPTCHAs.

REFERENCES

[1] M. A. Kouritzin, F. Newton, and B. Wu, “On random field completely automated public turing test to tell computers and humans apart generation,” 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] Yousef, Mohamed, Khaled F. Hussain, and Usama S. Mohammed. “Accurate, 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] Agrawal, K. K. ., P. . Sharma, G. . Kaur, S. . Keswani, R. . Rambabu, S. K. . Behra, K. . Tolani, and N. S. . Bhati. "Deep Learning-Enabled Image Segmentation for Precise Retinopathy Diagnosis". *International Journal of Intelligent Systems and Applications in Engineering*, vol. 12, no. 12s, Jan. 2024, pp. 567-74, https://ijisae.org/index.php/IJISAE/article/view/4541.
[11] 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.
[12] J. Yan and A. S. El Ahmad, "Breaking visual CAPTCHAs with naïve pattern recognition algorithms," in Proceedings of the 23rd Annual Computer Security Applications Conference, Miami Beach, FL, USA, December 2007, pp. 279 – 291.
[13] 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.
[14] 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.
[15] 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 (ICDAR), IEEE Computer Society, Kyoto, Japan, November 2017, pp. 1313 – 1317.
[16] 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 recognition," IEEE Transactions on Pattern Analysis & Machine Intelligence, vol. 39, no. 11, pp. 2298 – 2304, 2016.
[17] 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.
[18] D. Lin, F. Lin, Y. Lv, F. Cai, and D. Cao, "Chinese character CAPTCHA recognition and performance estimation via deep neural network," Neurocomputing, vol. 288, pp. 11 – 19, 2018.
[19] Samota, H. ., Sharma, S. ., Khan, H. ., Malathy, M. ., Singh, G. ., Surjeet, S. and Rambabu, R. . (2024) "A Novel Approach to Predicting Personality Behaviour from Social Media Data Using Deep Learning", *International Journal of Intelligent Systems and Applications in Engineering*, 12(15s), pp. 539–547. Available at: https://ijisae.org/index.php/IJISAE/article/view/4788

[20] 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.
[21] 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.
[22] Bostik, Ondrej, and Jan Klecka. "Recognition of CAPTCHA characters by supervised machine learning algorithms." IFAC-PapersOnLine 51, no. 6 (2018), pp. 208 – 213.
[23] Yousef, Mohamed, Khaled F. Hussain, and Usama S. Mohammed. "Accurate, data-efficient, unconstrained text recognition with convolutional neural networks." arXiv preprint arXiv:1812.11894 (2018).
[24] Karthik, CHBL-P., and Rajendran Adria Recasens. "Breaking microsofts CAPTCHA." Technical report (2015). [25] Kopp, Martin, Matej Nikl, and Martin Holena. "Breaking captchas with convolutional neural networks." ITAT 2017 Proceedings (2017): pp. 93 –99.
[26] Zhao, Nathan, Yi Liu, and Yijun Jiang. "CAPTCHA Breaking with Deep Learning," (2017).
[27] Stark, Fabian, Caner Hazrbas, Rudolph Triebel, and Daniel Cremers. "Captcha recognition with active deep learning." In Workshop new challenges in neural computation, vol. 2015, p. 94. Citeseer, 2015.
[28] Kwon, Hyun, Hyunsoo Yoon, and Ki-Woong Park. "CAPTCHA Image Generation Using Style Transfer Learning in Deep Neural Network." In International Workshop on Information Security Applications, pp. 234– 246. Springer, Cham, 2019.
[29] Zahra Noury and Mahdi Rezaei, "Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment," arXiv preprint arXiv:2006.08296, 2020.
[30] Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. "Densely connected convolutional networks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4700 – 4708. 2017.
[31] Garg, Geetika, and Chris Pollett. "Neural network captcha crackers." In 2016 Future Technologies Conference (FTC), pp. 853 – 861. IEEE, 2016.
[32] L. Wang, R. Zhang and D. Yin, "Image verification code identification of hyphen," Comput. Eng. Appl., 28 (2011), pp. 150 – 153.
[33] Python Image Captch Librady https://github.com/lepture/captcha, Last access: 15 June 2020.