ISSN: 2583-9292 (ONLINE)
DOI: 10.xx/ijdsml.2023.xx

CAPTCHA RECOGNITION USING MACHINE LEARNING AND DEEP LEARNING TECHNIQUES

S. Arivazhagan, M. Arun, F. Ruby Ranjitha Mary and B. Shamyughtha Bala
Department of Electronics and Communication Engineering, Mepco Schlenk Engineering College, India

Abstract

CAPTCHAs are widely used on the internet to determine whether a user is a human, and text-based CAPTCHAs are mostly used. The study on CAPTCHA recognition is meant for detecting the vulnerabilities in their security for preventing any malicious intrusion in the network. In this article, the Segmentation-based method and Segmentation-free method are used for recognition. In segmentation-based technique, text CAPTCHAs are segmented using contours and bounding-box method, SIFT and KAZE features are extracted and Support Vector Machine (SVM) and modified LeNet-5 model is used for recognition. In Segmentation-free approach, we propose a customized Convolutional Neural Network (CNN) for recognition. Really simple CAPTCHAs dataset and Vulnerable CAPTCHAs dataset achieved a highest recognition rate of 96.74% and 92.36% with pixel features using SVM. Also, in modified LeNet-5 the highest recognition rates achieved for these two datasets are 97.6% and 91.33% respectively. Using the customized CNN without segmentation, these two datasets achieved 99.6% and 25.31% success rates. Also, some three different complex 5-letter CAPTCHAs called ECE_1373_dataset, Captcha dataset and Captcha_2000 are tested in this model and achieved 82.09%, 62.96% and 61.33% accuracy rates.

Keywords:

CAPTCHA, SVM, CNN, LeNet-5

1. INTRODUCTION

CAPTCHA stands for Completely Automated Public Turing test to tell Computers and Humans Apart [1]. CAPTCHAs are used as security tools in multimedia for differentiating humans who are real users from any other automated users like robots. They have a wide range of applications in protecting the network and information security. They are mainly used as a part of securing the websites from any malicious program attacks by hackers such as automatic website registrations, online polling, dictionary attacks, spamming the blogs and so on. There are many different types of CAPTCHAs. Some of them include text CAPTCHA, image CAPTCHA, audio CAPTCHA, math CAPTCHA and reCAPTCHA with text CAPTCHA being the most popular [2]. Generally, the recognition rate of CAPTCHAs for human beings should be at least 85% and computers should have the least recognition rate of about 0.01%.

Even though there are various types of CAPTCHAs, many websites still rely on text CAPTCHAs for protection as a part of their security mechanism. But with the development of machine learning and deep learning techniques [33], it has become much easier to break these text CAPTCHAs. Now-a-days, machines are trained to recognize them with nearly equal recognition rates as humans.

Text CAPTCHAs are primarily made up of some random combinations of numbers (0-9) and English alphabets (a-z and A-Z). They may also use known words or phrases. Visual interference effects like rotation, twisting, adhesion, overlapping, variation in size, colour, shape of digits and characters with many background noises, dots and lines ensure the security of text CAPTCHAs. Some types of text CAPTCHAs include solid CAPTCHA, hollow CAPTCHA, three-dimensional CAPTCHA, and animated CAPTCHA. These text CAPTCHAs can be recognized using the available effective methods in machine learning and deep learning.

Many text-based CAPTCHA recognition algorithms generally available are divided into two categories. They are: (i) segmentation-based approach (ii) segmentation-free approach [3].

In the segmentation step, the CAPTCHA image is segmented into individual characters, and these individual characters are recognized using character recognition module [4]. The segmentation step is the most essential since it has a significant impact on the overall accuracy and efficiency of the system. CAPTCHA recognition is currently dominated by segmentation-free models. They don't need to segment CAPTCHAs into individual characters to recognise and classify the characters in input CAPTCHAs. Furthermore, most segmentation-free models have a high level of accuracy and efficiency [5].

In this paper, using both machine learning and deep learning approaches, the recognition rates of different 4-letter and 5-letter text-based CAPTCHAs are examined. For identifying CAPTCHAs, a suitable segmentation approach is employed and two different character recognition module is used in segmentation-based approach. A customized CNN is constructed for directly identifying CAPTCHAs in a segmentation-free approach. The accuracy rates in both techniques are then compared.

2. RELATED WORKS

Chellapilla and Simard [6] were the first to propose an attack on text-based CAPTCHA schemes using machine learning techniques. They mainly used Google/Gmail and Yahoo CAPTCHAs and achieved success rates of 89.3% and 95.3% respectively. It is solved CAPTCHAs using basic pre-processing techniques like thresholding, dilation and erosion. They used fill-flood segmentation and K-Nearest Neighbour (KNN) classifier and Principal Component Analysis (PCA) for recognizing CAPTCHAs with a success rate of 95% and 65% for several real-world examples like Rogers, PHPBB, PirateBay, Digg and Watercap [7]. Fiot and Paucher [8] used Support Vector Machines with polynomial kernel to break Gmail, Yahoo and Hotmail CAPTCHAs. Their method achieved a success rate of about 92.2% [8]. Zhang and Wang [9], used vertical and horizontal projection profile method for segmenting CAPTCHAs and used KNN for classifying the characters with 4 x 4 coarse grid features and all-points feature [9]. They achieved a success rate of about 95-98%. Burstein et al. (2011) [10] used CAPTCHAs from Gibbs, Slashdot, eBay and observed that SVM with linear kernel performed well with distorted characters and KNN performed well with a mix of five complex fonts achieving success rates of 61% and 62% respectively. Cruz-Peres et al. [11] solved the reCAPTCHA in websites using a heuristic segmentation by three-color bar code, cropped them using vertical segmentation with 82% success rate and implemented SVM-based learning classifier with Gaussian kernel and Sequential minimal optimization (SMO) algorithm feature vector for recognition of combinations of characters with a success rate of about 94% [11]. Bursztein et al. [12], introduced a new technique in which for segmentation they used contains a cut-point detector, slicer, scorer and arbiter and for classification and recognition SVM has been used. Their success rates against reCAPTCHA and Baidu are 33.34% and 38.68%. Starostenko et al. [13] achieved a segmentation success rate up to 82% for reCAPTCHA of version 2011 and 95.5% for reCAPTCHA of version 2012 with a recognition success rate of about 94% using SVM-based classifier.

Li et al. [14], proposed a new set of image processing and pattern recognition techniques to recognize e-banking CAPTCHA schemes using Complex Wavelet Structural Similarity (CW-SSIM) based pattern recognition and K-means layer segmentation with success rate nearly equal to 100%. Gao et al. [15] implemented an attack on hollow CAPTCHAs using Colour Filling Segmentation (CFS) for segmentation and CNN for recognition phase. They also introduced a novel technique called graph-based segmentation and recognition. Using this new technique, they have broken the hollow schemes deployed by Yahoo, Tencent, Sina, CmPay and Baidu and achieved a success rate of 36%, 89%, 59%, 66% and 51% respectively. Rui et al. [16], used two-dimensional Long Short-Term Memory – Recurrent Neural Network (LSTM-RNN) to recognize text-based CAPTCHAs and improved the recognition rate by about 55% for merged-type CAPTCHAs. Stark et al. [17], used “Alex Net” as their base CNN for recognition phase. Dileep et al. [18] solved reCAPTCHA using Recursive Cortical Network (RCN) for recognizing CAPTCHAs and achieved 57% to 66% success rates. Ye et al. [19], used Generative Adversarial Networks (GAN) for solving CAPTCHAs with 87% to 90% success rates [19]. Chen et al. [20] solved the hollow CAPTCHAs by pre-processing them using thinning operation followed by inner-outer contour filling algorithm and a customised CNN is used for recognition. Zi et al. [21], introduced an end-to-end attack on text CAPTCHAs without any pre-processing through a CNN and an attention-based RNN with success rates ranging from 74.8% to 97.3%.

In this paper, we use two different 4-letter CAPTCHAs named Really Simple CAPTCHAs and Vulnerable CAPTCHAs are used in both segmentation-based and segmentation-free approach. Also, three different 5-letter CAPTCHAs namely ECE_1373_dataset, Captcha dataset and Captcha_2000 are used for recognition in our proposed customized CNN. As per the authors knowledge, no research work has been done in these datasets and we are the first to use them.

3. MATERIALS AND METHODS

Different types of 4-letter and 5-letter CAPTCHAs used in our method is discussed below:

3.1 REALLY SIMPLE CAPTCHAS (DS1)

The 4-letter, white background text CAPTCHA containing numbers from 2-9 and capital letters from A-Z excluding I and O [27]. There is a total of 9,830 images each of size 24 x 72. Some sample images are shown in Fig.1

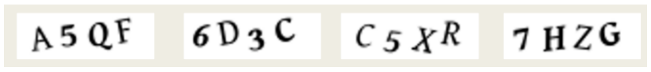


Fig.1. Sample images of DS1

3.2 VULNERABLE CAPTCHAS (DS2)

4-letter, colour background text CAPTCHA with some interference lines consisting of numbers from 2-9 and capital letters from A-Z excluding I and O [28]. Total number of images present is equal to 362 each of size 50 x 330. Some example images are shown in Figure: 2.

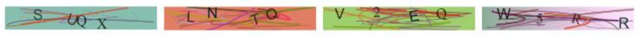


Fig.2. Sample images of DS2

3.3 ECE_1373_DATASET (DS3)

5-letter colour text CAPTCHA with some background dots and lines made up of digits and lowercase characters. Only 19 characters are present (2, 3, 4, 5, 6, 7, 8, b, c, d, e, f, g, m, n, p, w, x, y) [29]. There is a total of 74,831 images present each of size 50 x 150. Some sample images are shown in Fig.3

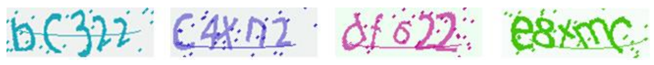


Fig.3. Sample images of DS3

3.4 CAPTCHA DATASET (DS4)

5-letter text CAPTCHA with some background lines made up of digits and lowercase characters. Only 19 characters are present (2, 3, 4, 5, 6, 7, 8, b, c, d, e, f, g, m, n, p, w, x, y) [30]. There is a total of 1,070 images present each of size 50 x 200. Some example images are shown in Fig.4

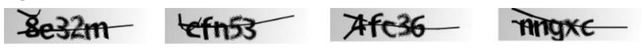


Fig.4. Sample images of DS4

3.5 CAPTCHA_2000 (DS5)

5-letter text CAPTCHA with some background noise like dots and lines consisting of numbers from 1-9 and uppercase characters from A-Z excluding 0 and O [31]. 2,000 images made up of 32 characters each of size 50 x 180 are present. Some example images are shown in Fig.5.

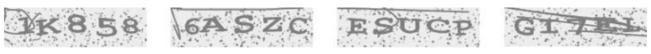


Fig.5. Sample images of DS5

3.6 CHARS74K DATASET (TD1)

Chars74K dataset [32] is taken as training dataset-1 in our approach since DS1 and DS2 contains only these valid characters. This dataset with individual images of letters from A-Z, a-z numbers from 0-9 contains a total of 74,000 images belonging to 64 classes each of size 128 x 128. Out of these, 40,433 images containing characters from A-Z excluding I and O and numbers from 2-9 excluding 0 and 1 belonging to 32 classes which are resized to 32 x 32 is used for training. Some of the images in this dataset is shown in Fig.6.

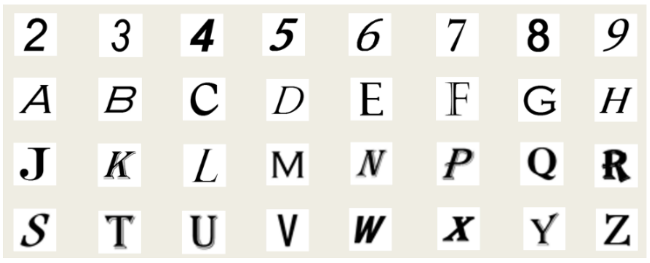


Fig.6. Sample images of Chars74K dataset

4. PROPOSED METHODOLOGY

The block diagram of segmentation-based approach which is our proposed work-1 for recognizing CAPTCHAs is shown in the Fig.7. TD1, DS1 and DS2 are used in this approach. DS1 is a grayscale image and DS2 is a HSV image. They are then pre-processed using Otsu's thresholding method. For segmenting the characters, contours and bounding box method is used. For character recognition, two models are used. They are: (i) SVM (machine-learning based) and (ii) modified LeNet-5 CNN (deep-learning based). Finally, recognition result is obtained.

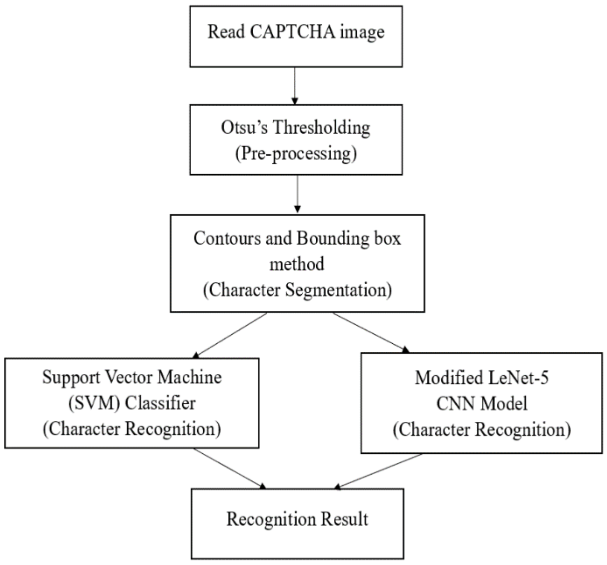

Read CAPTCHA image
Otsu's Thresholding (Pre-processing)
Contours and Bounding box method (Character Segmentation)
Support Vector Machine (SVM) Classifier (Character Recognition)
Modified LeNet-5 CNN Model (Character Recognition)
Recognition Result

Fig.7. Segmentation-based recognition block diagram

DS1 and DS2 are almost similar. The only difference is their size and background colour. DS1 images are read in grayscale format. DS2 images are read in HSV format since it gives a more logical way of grouping colours. Here, only Value channel (V-channel) image is taken since the background intensity in lower than the foreground intensity.

4.1 PRE-PROCESSING

Otsu's thresholding method is used for pre-processing the input image [9]. Thresholding is the process of separating the foreground pixels from the background pixels. Otsu's method iteratively searches over the space for finding the threshold value which minimizes the within-class variance of both the classes (background and foreground). The formula for finding the threshold value is given below:

$$\sigma_w^2 = w_0(t)\sigma_0^2 + w_1(t)\sigma_1^2 \tag{1}$$

where, ω_1 and ω_1 are the probabilities of classes separated by threshold t; σ_0^2 and σ_1^2 are the variances of two classes.

The images of DS1 and DS2 after applying Otsu's thresholding is shown in Fig.8 and Fig.9, respectively.

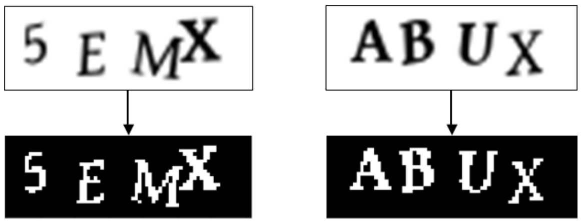


Fig.8. DS1 after thresholding

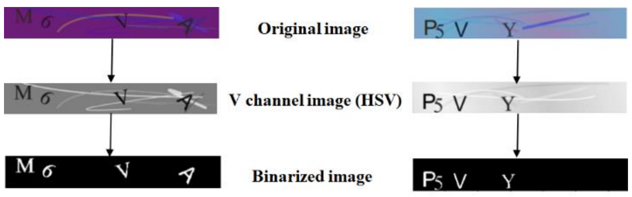


Fig.9. DS2 after thresholding

4.2 CHARACTER SEGMENTATION

Contours and Bounding box method [34] is used for segmenting the characters. The curves joining all the continuous points that are along the boundary of the object with same colour and intensity are termed as contours. After finding the coordinates like x-coordinate, y-coordinate, width, and height of the external contour imaginary rectangular boxes are drawn around each detected character and they are cropped into four individual images. These boxes determined by the upper-left corner and lower-right corner coordinates describing the spatial location of the objects are called bounding boxes. If the (width / height) ratio is more than 1.25, the width is divided into half otherwise same value is maintained for DS1. If (width * height) value, that is the area of the contour is below 100, it is ignored otherwise it is considered for DS2. Achieved nearly 100% and 83.7% success rate in segmenting the characters for dataset-1 and dataset-2 respectively. DS1 after segmentation are shown in Fig.10. Fig.11 (a), 11 (b) show some correctly and wrongly segmented images of DS2 respectively. The Table.1 shows the segmentation accuracy rates for both the datasets.

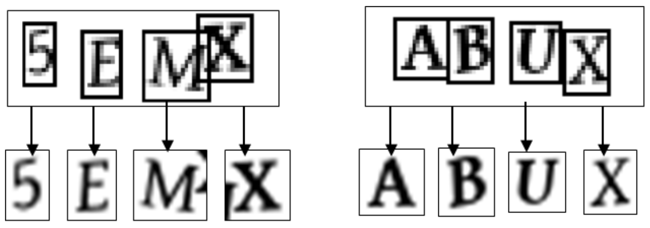


Fig.10. Segmented characters of DS1

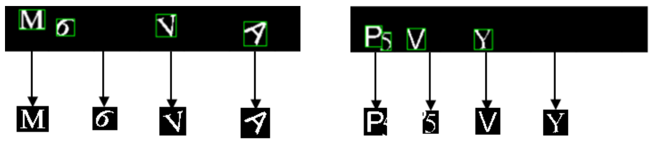


Fig.11(a). Correctly segmented characters of DS2

Table.1. Segmentation accuracy for DS1 and DS2

Dataset	Total no. of images	Correctly segmented	Wrongly segmented	Segmentation accuracy (%)
DS1	9,830	9,830	-	100
DS2	362	303	59	83.70

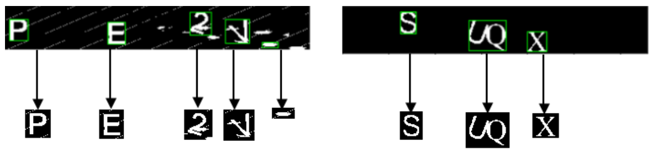


Fig.11(b). Wrongly segmented characters of DS2

4.3 FEATURE EXTRACTION

While working with very large number of datasets, in order to avoid overfitting, some feature extraction techniques are used which are also called dimensionality reduction techniques. Here two types of feature extraction techniques are used. They are: (i) SIFT feature extraction (ii) KAZE feature extraction.

First the pixels are directly used as features. Extracting the pixels as features from a grayscale image, involves simply reshaping it to the desired size and returning as the array form of the image. Here, all the segmented characters are reshaped to (32 x 32) pixel size.

4.3.1 SIFT Feature Extraction:

SIFT stands for Scale Invariant Feature Transform, developed by Lowe in 1999, which provides an image with a set of effective features which do not suffer from any complications like scale changes and rotation changes [24]. This algorithm has four main steps: (i) Scale-space peak selection (ii) Key point localization (iii) Orientation assignment (iv) Key point descriptor. SIFT feature extracted images of DS1 and DS2 is shown in Fig.12.

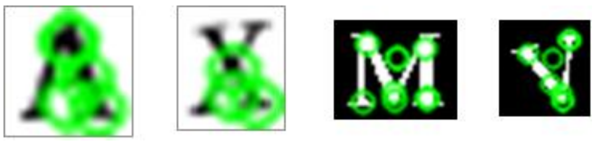


Fig.12. SIFT feature extraction for DS1 and DS2

4.3.2 KAZE Feature Extraction:

KAZE is a 2D feature extraction algorithm developed by Pablo Fernandez et al. in the year 2012 [25]. This algorithm is slightly different from SIFT feature extraction and has the following three steps: (i) Non-linear scale space computation (ii) Feature detection (iii) Feature description. KAZE feature extracted images of DS1 and DS2 is shown in Fig.13.

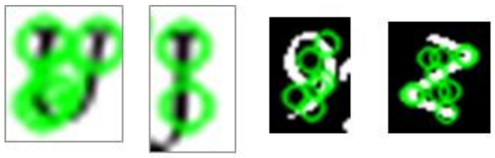


Fig.13. KAZE feature extraction for DS1 and DS2

4.4 CHARACTER RECOGNITION

The process of recognizing the text inside the images is called character recognition. Here, two approaches for recognizing the characters in CAPTCHA are used. They are: (i) Support Vector Machine (machine-learning based approach) (ii) Modified LeNet-5 CNN (deep-learning based approach).

4.4.1 Traditional Approach (using SVM):

SVM is used for classification of characters [22]. Here 32 classes of numbers and letters containing characters (A-Z) and numbers (0-9) except I, O 1 and 0 are present. The idea of SVM is to separate these classes using a suitable hyperplane or kernel. Radial Basis Function (RBF) kernel is chosen to separate the classes. The formula for RBF kernel is given below as in Eq.(2).

$$K(x, x') = e^{-\gamma\|x-x'\|^2} \tag{2}$$

where, $\|x-x'\|^2$ is the squared Euclidean distance between two feature vectors and γ is the scalar value which defines how much impact a single training example has.

PCA [7] is used to normalize the pixels of the characters and to normalize against linear transformations. It is used before using SVM classifier. All the data are fitted in PCA transformation after extracting the pixel features. 99% variance is maintained with the pixel features.

After extracting pixel features from the training dataset (TD1), PCA is used to normalize them and then passed onto SVM with RBF kernel for classification. The segmented characters from DS1 and DS2 are used for testing and prediction result is got. For SIFT and KAZE feature extraction, to train the classifier with the extracted feature descriptors involves certain steps. From the extracted feature descriptors which contains each feature as a point in the coordinate space of the descriptor, a vocabulary or codebook is created with n random sample points of interest chosen randomly by the detector. K-means clustering algorithm is used to quantize the features to find the visual features for the construction of vocabulary. Then a histogram is created which contains the occurrence of a feature in the image. This histogram representation is used to train the classifier model and then test the new images and label them.

4.4.2 Modified LeNet-5 Model:

In this method a modified version of LeNet-5 model is used for recognition. The modified LeNet-5 architecture is shown in Fig.14.

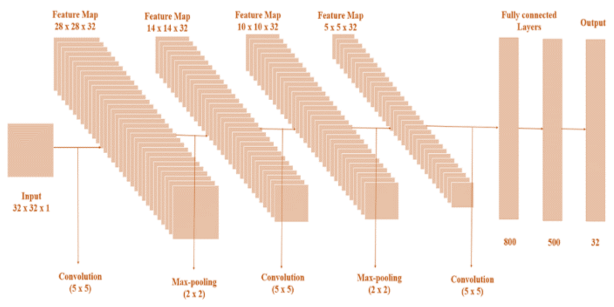


Fig.14. Modified LeNet-5 architecture

The original LeNet-5 architecture consists of a total of 5 layers which includes three sets of convolutional layers each followed by an average pooling layer. At last, it has two fully connected layers with a SoftMax classifier for classification. Here, we slightly modified the original LeNet-5 architecture for our convenience in classifying 32 classes of characters. This modified LeNet-5 architecture consists of 7 layers which includes 3 convolutional layers, 2 max-pooling layers and 2 fully connected layers. Network layers and their dimensions are shown in Table: 2. The input image layer is initially provided as a grayscale image with a size of 32 x 32. The first 2 convolution layers have the kernel size as 5 x 5 and a total of 32 filters to extract features from the input image. The output size of each convolution block is given in Eq.(3):

$$O_w = \frac{W - F_w + 2P}{S_W} + 1 \, ; O_h = \frac{H - F_h + 2P}{S_h} + 1 \tag{3}$$

where O_w, O_h - output width and output height, W, H - input width and input height, F_w, F_h - filter width and filter height, P - padding size, S_w, S_h - stride width and stride height.

Table.2. Network layers and dimensions for modified LeNet-5 model

Layers	Layer Dimensions	Output Dimensions
Input image	32 x 32 x 1	32 x 32
2D Convolution 1	5 x 5, stride = 1, filters = 32	28 x 28
Max-pooling 1	2 x 2, stride = 2	14 x 14
2D Convolution 2	5 x 5, stride = 1, filters = 32	10 x 10
Max-pooling 2	2 x 2, stride = 2	5 x 5
2D Convolution 3	5 x 5, stride = 1, filters = 32	1 x 800
Fully connected 1	500	1 x 500
Fully connected 2	32	1 x 32

The two average pooling layers in original architecture is replaced with two max-pooling layers of size 2 x 2 and stride 2. The output size of max-pooling block is given by the Eq.(4).

$$O = \frac{W - F}{S} + 1 \tag{4}$$

where, O is the output size, W is the input width, F is the filter size, S is the size of the stride.

The third convolution layer is replaced with eight hundred filters of size 5 x 5 which is followed by a fully connected layer having 500 neurons. Tanh activation in all the convolution layers is replaced with Rectified Linear Unit (ReLU) activation function. It is used in each convolution layer which outputs the input directly only for positive inputs. This function defined in the Eq.((5) provides excellent accuracy because it eliminates the vanishing gradient problem.

$$y(z)=\max(0,z) \tag{5}$$

For all classification problems, a classification layer computes cross-entropy loss and Adaptive moment estimation (Adam) optimizer is used to optimize the loss function. 40,433 images belonging to 32 classes (TD1) are used for training the modified LeNet-5 model with 500 steps per epoch with 32 batch size and the training process is for 10 epochs. In testing phase 9,830 images (DS1) and 303 images (DS2) are used which are segmented and then each character is predicted using the trained model. The same process is carried out for both original and modified LeNet-5 CNN. Also, different combinations of training and testing datasets are used (DS1 and DS2) and their accuracy rates are compared for all these combinations. The block diagram of segmentation free approach which is our proposed work-2 for recognizing CAPTCHAs using a customized CNN is shown in the Fig.15.

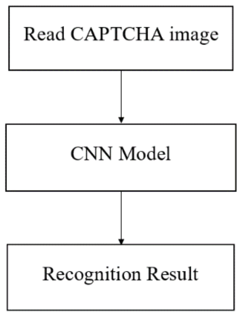


Fig.15. Segmentation-free recognition block diagram

In this approach, the original image along with its label is as such used without any pre-processing and segmentation techniques. A set of images from the dataset is used for training and validation and the remaining images are used for testing and the prediction accuracy is got finally. The network architecture for the customized CNN model is shown in Fig.16. This customized CNN architecture consists of three convolution layers, 3 max-pooling layers, three batch normalization layers, two fully connected layers and a dropout layer. It takes input of any size that is it accepts the size of DS1 and DS2 images. The first convolution layer consists of 16 kernels each of size 3 x 3 and zero padding is used. It is followed by a layer of batch normalization (BN) and a max-pooling layer of kernel size 2 x 2 and zero padding. The second convolution layer consists of 32 kernels of size 3 x 3 and zero padding is used. It is again followed by a batch normalization layer an a max-pooling layer of the same kernel size 2 x 2 and zero padding. The last convolution layer is same as the second convolution layer and is again followed by the same batch normalization and max-pooling layer configuration. The number of neurons in the first fully connected layer is 384. The activation function used in all these layers is ReLU activation. It is followed by a dropout layer with 40%. The final fully connected layer has 128 neurons. SoftMax activation is used at the final layer of classification.

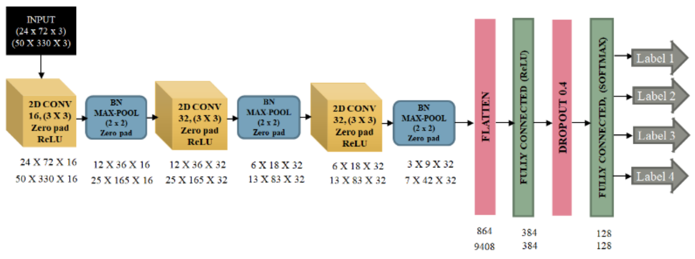


Fig.16. Customized CNN architecture for segmentation-free method

The CNN structure and parameters are shown in Table.3.

Table.3. Customized CNN layer dimensions

Layers	Layer Dimensions
Input image	Input size of DS1, DS2
2D Convolution 1	3 x 3, stride = 1, filters = 16, zero padding
Max-pooling 1	2 x 2, stride = 1, zero padding
2D Convolution 2	3 x 3, stride = 1, filters = 32, zero padding
Max-pooling 2	2 x 2, stride = 1, zero padding
2D Convolution 3	3 x 3, stride = 1, filters = 32, zero padding
Max-pooling 3	2 x 2, stride = 1, zero padding
Fully connected 1	1 x 384
Fully connected 2	1 x 128

4.4.3 *Multi-Task Joint Training for Customized CNN*

To improve the generalization ability by using domain specific information in the training signals hidden in the multiple related tasks multi-task joint learning is used [26]. It uses shared representation to parallel training multiple tasks. It reduces the number of models and improves learning efficiency. This involves dividing the image labels into multiple learning tasks with each task training one character and all tasks training together. The structure of the multi-task joint training network is shown in Fig.16.

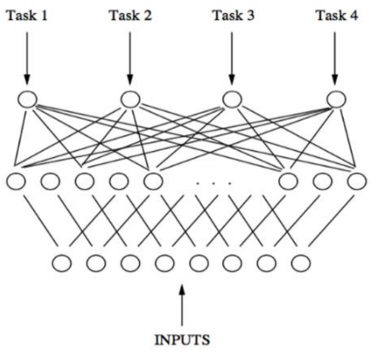


Fig.16. Multi-task joint training model

For DS1 and DS2 containing 4 letters, in the output layer every 32 neurons is used to predict a character since there are only 32 valid characters in these datasets. A bijection function $\theta(x)$ given in Eq.((6) is used to map a character x ∈{2' …9' , *A'* …*H'* , *J'* …,*P'*…*Z'*} to an integer *y* ∈{0' … 31'}.

$$\theta(x) = \begin{cases} 0 \sim 7 & x = 2' \sim 9'(excluding\,0\,and\,1) \\ 8 \sim 31 & x = A' \sim Z'(excluding\,I\,and\,O) \end{cases} \quad (6)$$

First set of 32 neurons is assigned to the first character of the sequence, the second set of 32 neurons is assigned for the second character and so on. Therefore, the output layer has 4 x 32 = 128 neurons.

DS1 has a total of 9,830 images out of which 7,372 is split for training, 1,474 for validation and 984 for testing. It is trained for 100 epochs with batch size 32. DS2 has a total of 362 images out of which 271 is split for training, 54 for validation and 37 for testing. It is also trained for 100 epochs with batch size 32.

In order to test the robustness of this model, some complex CAPTCHAs (DS3, DS4 and DS5) are recognized. For this recognition, small change in this model is made in the fully connected layers and bijection function used for mapping. Since, DS3 and DS4 has only 19 valid characters, the first fully connected layer is made to have 285 neurons and the second fully connected layer is made to have 95 neurons. The bijection function is modified to map a character *x* ∈{2' …8', *b*', *c*', *d*', *e*', *f*', *g*', *m*', *n*', *p*', *w*', *x*', *y*'} to an integer *y* ∈{0'…18'}. First 19 neurons are assigned to first character, the next set of 19 neurons is assigned to second character and so on. The output layer has 5 x 19 = 95 neurons. For DS5 having 34 valid characters, the first and second fully-connected layers are modified to have 510 neurons and 170 neurons. The bijection function is changed to map the character *x* ∈{1' …9' , *A*' … *N*' , *P*'… *Z*'} to an integer *y* ∈{0'…33'}.

5. RESULTS AND DISCUSSION

Here we assume that even if one of the characters in the CAPTCHA images is misclassified, it is considered as a wrong prediction. After segmentation using contours and bounding box method, and extracting features using SIFT and KAZE algorithms, SVM with RBF kernel is used for recognition.TD1, DS1 and DS2 are used for training and testing. The accuracy comparison is shown in Table.4.

Table.4. CAPTCHA recognition accuracy for SVM

Dataset		Accuracy (%)		
Train	Test	Pixel features	SIFT features	KAZE features
TD1	DS1	70.50	46.56	67.95
TD1	DS2	17.82	16.50	16.17
DS1	DS2	88.78	84.82	86.14
DS2	DS1	89.36	81.15	87.59

When DS2 is used for training and segmented DS1 is tested the highest accuracy of 89.36%. For DS2 highest accuracy of 88.78% is obtained for pixel features when trained with segmented DS1. DS1 has the least accuracy of 46.56% with SIFT features when trained with TD1 and DS2 has the least accuracy of 16.17% with KAZE features when trained with TD1.

Another comparison is done by splitting DS1 and DS2 in different ratios for training and testing and their recognition rates are observed for these three features. The comparison results are shown in Table.5.

Table.5. CAPTCHA recognition accuracy for SVM with different train-test split

Dataset		**Individual Accuracy (%)**		
Train	**Test**	**Pixel features**	**SIFT features**	**KAZE features**
75% DS1 25% DS2	25% DS1 75% DS2	96.74 70.70	83.54 66.51	90.99 72.80
50% DS1 50% DS2	50% DS1 50% DS2	95.50 81.12	78.09 75.52	89.31 83.45
25% DS1 75% DS2	75% DS1 25% DS2	95.25 92.36	79.20 85.19	87.21 91.36

For DS1, highest accuracy of 96.74% is achieved with pixel features when 75% DS1 is trained and tested with remaining 25% and the least accuracy of 78.09% is achieved with SIFT features when trained and tested with equal split of DS1. For DS2, highest accuracy of 92.36% is achieved with pixel features when 75% is trained and remaining is tested. The least accuracy of 66.51% for DS2 is achieved when trained with 25% DS2 and tested with remaining 75%. Not much difference is observed in the accuracy of DS1 when the train-test split is 25%-75% and 75%-25%. As in previous case for modified LeNet-5 model, TD1, DS1 and DS2 are used for training and testing in different combinations and the recognition rates are compared in Table.6.

Table.6. Modified LeNet-5 recognition accuracy for CAPTCHAs

Dataset		**Accuracy (%)**	
		Original LeNet-5	**Modified LeNet-5**
Train	**Test**		
TD1	DS1	90.76	95.5
TD1	DS2	32.81	35.64
DS1	DS2	85.96	88.23
DS2	DS1	89.12	92.3

Both DS1 and DS2 showed better accuracy rates in modified LeNet-5 than in the original model. 95.5% is the highest rate for DS1 when trained with TD1 and 88.23% is the highest for DS2 when trained with segmented DS1.

Same comparison as before by splitting DS1 and DS2 in different combination ratios for training and testing is carried out. The features are extracted by the convolution and max-pooling layers. The accuracy rates are compared for both original and modified LeNet-5 models. It is shown in Table.7.

Table.7. Modified LeNet-5 recognition accuracy for different train-test split

Dataset		**Individual Accuracy (%)**	
Train	**Test**	**Original LeNet-5**	**Modified LeNet-5**
50%DS1	50%DS1	93.21	97.6
50%DS2	50%DS2	87.62	88.12
75%DS1 25%DS2	25%DS1 75%DS2	90.42 70.88	94.9 72.9
25%DS1 75%DS2	75%DS1 25%DS2	87.22 89.63	90.03 91.33

Here also modified LeNet-5 showed better results than original model achieving the highest accuracy of 97.6% for DS1 in 50%-50% train-test split and 91.33% for DS2 in 75%-25% train-test split.

5.1 SEGMENTATION-FREE APPROACH RECOGNITION ACCURACY

DS1 and DS2 are used as whole CAPTCHA images without performing any segmentation and recognized with the customized CNN design for 100 epochs. Their accuracy rates are compared in this section, and it is shown in Table.8.

Table.8. Customized CNN recognition accuracy for DS1 & DS2

Dataset	**Total**	**Training**	**Validation**	**Testing**	**Accuracy (%)**
DS1	9,830	7,372	1,474	984	99.60
DS2	362	271	54	37	25.31

For DS1, 7,372 images are trained, 1,474 images are validated along with their labels in the same way mentioned in the multi-task joint training method and 984 images are used for testing. In the same way, for DS2, 271 images and 54 images are used for training and validation. 37 images are used as test dataset. For DS1 our model achieved 99.60% accuracy rate and for DS2 accuracy rate achieved is only 25.31%. Some of the correctly and wrongly predicted CAPTCHAs are tabulated in Table.9(a) and Table.9(b).

Table.9(a). Sample prediction result for DS1

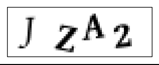
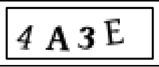
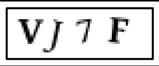
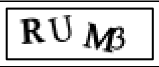
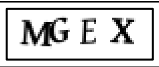
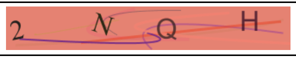
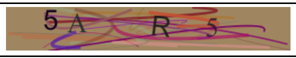
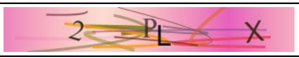
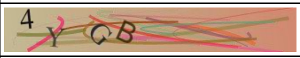
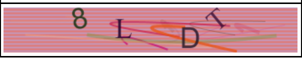
Original image	**Predicted Value**	**Result**
	JZA2	Correct
	4A3E	Correct
	VJ7F	Correct
	RUM8	Wrong
	M6EX	Wrong

Table.9(b). Sample prediction result for DS2

Original image	**Predicted Value**	**Result**
	2NQH	Correct
	5AR5	Correct
	2PLX	Correct
	4RBE	Wrong
	BLD7	Wrong

5.2 COMPLEX CAPTCHA DATASET RECOGNITION ACCURACY FOR CUSTOMIZED CNN (SEGMENTATION-FREE)

DS3, DS4 and DS5 are 5-letter CAPTCHAs which are somewhat complex than DS1 and DS2. They are literally difficult to segment into individual characters. So, they were directly used in our customized CNN model for recognizing without performing any segmentation. Their split for training, validation and testing with 500 epochs and batch size of and the obtained recognition accuracy are discussed in Table: 10. In DS3 out of all the 74,831 images present, 56,123 images are trained, 11,224 images are used for validation. They were trained for 50 epochs with batch size as 32. For testing 7,484 images are used and achieved 82.09% as the success recognition accuracy.

Table.10. Customized CNN recognition accuracy for DS3, DS4, DS5

Dataset	Total	Training	Validation	Testing	Accuracy (%)
DS3	74,831	56,123	11,224	7,484	82.09
DS4	1,070	802	160	108	62.96
DS5	2,000	1,700	150	150	61.33

In DS4 out of all the 1,070 images present, 802 images are trained, 160 images are used for validation. They were trained for 500 epochs with batch size as 32. 100 images are used as testing data and achieved 62.96% as the success recognition accuracy. In DS5 out of all the 2,000 images present, 1,700 images are trained, 150 images are used for validation. They were trained for 500 epochs with batch size as 32. 150 images are used as testing data and achieved 61.33% as the success recognition accuracy. Some of the correctly predicted and wrongly predicted CAPTCHAs of DS3, DS4 and DS5 are listed in Table.11(a), Table.11(b) and Table.11(c)

Table.11(a). Sample prediction result for DS3

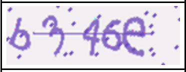
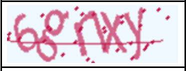
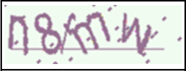
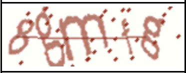
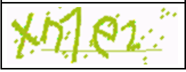
Original image	Predicted Value	Result
	b346e	Correct
	6gnxy	Correct
	n8fmw	Correct
	8gmxg	Wrong
	xm7e2	Wrong

Table.11(b). Sample prediction result for DS4

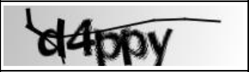
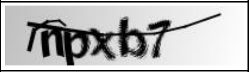
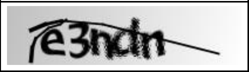
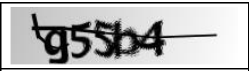
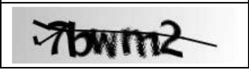
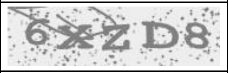
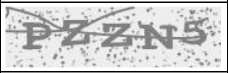
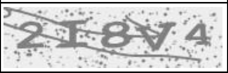
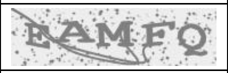
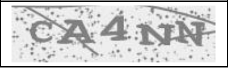
Original image	Predicted Value	Result
	d4ppy	Correct
	npxb7	Correct
	e3ndn	Correct
	g55m4	Wrong
	7bmm2	Wrong

Table.11(c). Sample prediction result for DS5

Original image	Predicted Value	Result
	6XZD8	Correct
	PZZN5	Correct
	2I8V4	Correct
	BAMFQ	Wrong
	6A4NN	Wrong

6. CONCLUSION

In this paper, a CAPTCHA recognition technology based on segmentation-based and segmentation-free approaches are implemented. Segmentation-free recognition with multi-task joint training of customized CNN showed excellent recognition accuracy rate of 99.6% for Really Simple CAPTCHAs dataset. For Vulnerable CAPTCHAs dataset, better accuracy rate of 96.74% is achieved in segmentation-based approach using SVM classifier with RBF kernel and pixels as features. With the customized CNN used in non-segmentation approach some complex CAPTCHA datasets (DS3, DS4 and DS5) showed better recognition accuracy rates of 82.09%, 62.96% and 61.33% respectively. Overall, it is seen that CAPTCHAs with no background interference can be efficiently recognized using machine learning techniques while it is not so efficient for CAPTCHAs with some noise, lines and interference in its background. Such complex CAPTCHAs can successfully be recognized with better accuracy rates in deep learning-based techniques. Thus, it can be concluded that these type of text CAPTCHAs are more vulnerable to machine attacks which can be a threat to network security since they are recognized with better accuracy rates using machine learning and deep learning techniques.

REFERENCES

[1] L. Von Ahn and J. Langford, "*CAPTCHA: Using Hard AI Problems for Security*", Springer, 2003.

[2] Y. Zi, H. Gao, Z. Cheng and Y. Liu, "An End-to-End Attack on Text CAPTCHAs", *IEEE Transactions on Information Forensics and Security*, Vol. 15, pp. 753-766, 2020.

[3] A. Thobhani and A. Abdussalam, "CAPTCHA Recognition using Deep Learning with Attached Binary Images", *Electronics*, Vol. 9, pp. 1522-1532, 2020.

[4] A. Christos and Christopher Town, "Character Segmentation for Automatic CAPTCHA Solving", *Open Computer Science Journal*, Vol. 13, No. 1, pp. 1-14, 2014.

[5] Antreas Dionysiou and Elias Athanasopoulos, "SoK: Machine vs. Machine A Systematic Classification of Automated Machine Learning-Based CAPTCHA Solvers", *Computers and Security*, Vol. 34, No. 2, pp. 101947-101957, 2020.

[6] Kumar Chellapilla and Patrice Y. Simard, "Using Machine Learning to Break Visual Human Interaction Proofs", *Proceedings of International Conference on Neural Information Processing Systems*, pp. 1-7, 2004.

[7] M.W. Hindle and R.C. Holt, "Reverse Engineering CAPTCHAs", *Proceedings of International Conference on Reverse Engineering*, pp. 59-68, 2008.

[8] A. Fiot, Jean Baptiste and Remi Paucher, "The Captchacker Project", *Proceedings of International Conference on Computer and Security*, pp. 79-88, 2009.

[9] J. Zhang and X. Wang, "Breaking Internet Banking CAPTCHA Based on Instance Learning", *Proceedings of International Symposium on Computational Intelligence and Design*, pp. 39-43, 2010.

[10] Elie Bursztein, Matthieu Martin and John Mitchell, "Text-Based CAPTCHA Strengths and Weaknesses", *Proceedings of International Conference on Computer and Communications Security*, pp. 125-138, 2011.

[11] C. Cruz Perez and L. Reyes Cabrera, "Breaking reCAPTCHAs with Unpredictable Collapse: Heuristic Character Segmentation and Recognition", *Proceedings of International Conference on Computer and Security*, pp. 110-118, 2012.

[12] J.F. Martínez-Trinidad, "*Pattern Recognition*", Springer, 2021.

[13] E. Bursztein and J.C. Mitchell, "The End is Nigh: Generic Solving of Text-based {CAPTCHAs}", *Proceedings of International Workshop on Offensive Technologies*, pp. 1-5, 2014.

[14] O. Starostenko, F. Uceda-Ponga and V. Alarcon-Aquino, "Breaking Text-Based CAPTCHAs with Variable Word and Character Orientation", *Pattern Recognition*, Vol. 48, No. 4, pp. 1101-1112, 2015.

[15] Z. Li, S. Shujun, M. Amier Haider Shah and Roland Schmitz, "Breaking E-Banking CAPTCHAs", *Proceedings of Annual Conference on Computer Security Applications*, pp. 171-180, 2010.

[16] Haichang Gao, Wei Wang, Jiao Qi, Xuqin Wang, Xiyang Liu and Jeff Yan, "The Robustness of Hollow CAPTCHAs", *Proceedings of ACM Conference on Computer and Communications Security*, pp. 1075-1086, 2013.

[17] C. Rui, Y. Jing, H. Rong-Gui and H. Shu-Guang, "A Novel LSTM-RNN Decoding Algorithm in CAPTCHA Recognition", *Proceedings of International Conference on Instrumentation, Measurement, Computer, Communication and Control*, pp. 766-771, 2013.

[18] F. Stark and Daniel Cremers, "Captcha Recognition with Active Deep Learning", *Proceedings of Workshop on New Challenges in Neural Computation*, pp. 94-99, 2015.

[19] G. Dileep, H. Wang, A. Lavin and D.S. Phoenix, "A Generative Vision Model that Trains with High Data Efficiency and Breaks Text-Based Captchas", *Science*, Vol. 358, pp. 6368-6387, 2017.

[20] Guixin Ye, Zhanyong Tang, Dingyi Fang, Zhanxing Zhu, Yansong Feng, Pengfei Xu, Xiaojiang Chen and Zheng Wang, "Yet Another Text Captcha Solver: A Generative Adversarial Network Based Approach", *Proceedings of ACM Conference on Computer and Communications Security*, pp. 332-348, 2018.

[21] J. Chen and D. Gong, "An Attack on Hollow Captcha using Accurate Filling and Nonredundant Merging", *IETE Technical Review*, Vol. 35, pp. 106-118, 2018.

[22] Y. Zi, Z. Cheng and Y. Liu, "An End-to-End Attack on Text Captchas", *IEEE Transactions on Information Forensics and Security*, Vol. 15, pp. 753-766, 2020.

[23] N. Sarvjeet and Naveen Dhillon, "SVM Classifier based Handwritten Character Recognition", *International Research Journal of Engineering and Technology*, Vol. 8, No. 2, pp. 1-14, 2021.

[24] T.E. De Campos, B.R. Babu and M. Varma, "Character Recognition in Natural Images", *Proceedings of International Conference on Computer Vision Theory and Applications*, pp. 1-10, 2009.

[25] David G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Master Thesis, Department of Computer Science, University of British Columbia, pp. 1-98, 2004.

[26] Pablo F. Alcantarilla, Adrien Bartoli and Andrew J. Davison, "KAZE Features", Master Thesis, Department of Computer Science, Universite d'Auvergne, pp. 1-112, 2012.

[27] Y. Hu, L. Chen and J. Cheng, "A CAPTCHA Recognition Technology based on Deep Learning", *Proceedings of IEEE Conference on Industrial Electronics and Applications*, pp. 617-620, 2018.

[28] Captcha Letter, Available at https://www.kaggle.com/genesis16/captcha-4-letter, Accessed in 2021.

[29] Vulnerable Data, Available at https://www.kaggle.com/code/electrototo/starter-vulnerable-captchas/data, Accessed in 2023.

[30] Srivat 97, Available at https://www.kaggle.com/datasets/srivat97/ece-1373-dataset, Accessed in 2022.

[31] Captcha Dataset, Available at https://www.researchgate.net/publication/248380891_captcha_dataset, Accessed in 2021.

[32] Captcha Greysky Dataset, Available at https://www.kaggle.com/datasets/greysky/captcha-dataset, Accessed in 2023.

[33] Arun Madakannu and A. Selvaraj, "A Unified Feature Descriptor for Generic Character Recognition based on Zoning and Histogram of Gradients", *Neural Computing and Applications*, Vol. 34, pp. 1-12, 2022.

[34] R. Wason, "Deep Learning: Evolution and Expansion", *Cognitive Systems Research*, Vol. 50, pp. 1-17, 2018.

[35] C. Boufenar and M. Batouche, "Investigation on Deep Learning for Off-Line Handwritten Arabic Character Recognition", *Cognitive Systems Research*, Vol. 50, pp. 180-195, 2018.