UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Text-based CAPTCHA Vulnerability Assessment using Deep Learning-based Solver

Daniel Alejandro Aguilar Fauta

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 10 de mayo de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Text-based CAPTCHA Vulnerability Assessment using Deep Learning-based Solver

Daniel Alejandro Aguilar Fauta

Nombre del profesor, Título académico	**Daniel Riofrio, PhD.**

Quito, 10 de mayo de 2021

© DERECHOS DE AUTOR

Nombres y apellidos:	Daniel Alejandro Aguilar Fauta
Código:	00139519
Cédula de identidad:	1726290230
Lugar y fecha:	Quito, 10 de mayo de 2021

RESUMEN

El objetivo de este trabajo es probar la seguridad que ofrecen los CAPTCHA basados en texto. Presentamos diferentes tipos de CAPTCHAs y un preprocesamiento y segmentación para limpiar la distorsión de los CAPTCHAs y recortar sus dígitos o caracteres en imágenes individuales. Presentamos una arquitectura de una red neuronal convolucional que se entrenó bajo varios hiperparámetros, lo que permite la comparación de múltiples modelos con diferentes combinaciones de tamaños de lote, épocas y optimizador. Confirmamos que el uso de CAPTCHAs basados en texto ya no es un mecanismo seguro de protección porque con técnicas simples de visión por computadora y los actuales algoritmos de aprendizaje automatizado se pueden resolver fácilmente. Alcanzamos un 90.49% de precisión con nuestro modelo entrenado con una combinación de cuatro conjuntos de datos, y hasta un 97.10% con un conjunto de datos, lo suficiente para considerar estos esquemas inseguros en la práctica.

Palabras clave: CAPTCHAs basados en texto, Red Neuronal Convolucional, Aprendizaje profundo, LeNet, Visión por computadora.

ABSTRACT

The focus of this work is to test the security offered by Text-based CAPTCHAs. We present different types of CAPTCHAs and a preprocessing and segmentation process to clean noise in CAPTCHA Images and crop digits or character in single images. We present a convolutional neural network architecture which is trained under several hyperparameter; thus, allowing for comparison of multiple models with different combinations of batch sizes, epochs, and optimizer. We confirmed that using Text-based CAPTCHAs is no longer a secure mechanism for protection because with simple computer vision techniques and current machine learning algorithms can be broken. We achieved a 90.49% of accuracy with our model trained with a mix of four datasets, and up to 97.10% with one dataset, which is enough to consider these schemes insecure in practice.

Keywords: Text-Based CAPTCHAs, Convolutional Neural Network, Deep Learning, LeNet, Computer Vision.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

A CAPTCHA (Stark et al. 2015) (Completely Automated Public Turing test to tell Computers and Humans Apart) is a measure to deal with cyber-attacks, security threats and spam. They were introduced to stop automated attacks because they are based on hard problems for an artificial intelligence but easy to resolve for a human.

An important task for CAPTCHAs is to prevent websites from being accessed by an automatic program and bots that waste network resources, maintain polls accuracy, prevent bots from spamming false reviews, false comments and contact forms. CAPTCHAs have been successfully applied to most of the most popular websites (Bursztein et al. 2011) such as Google, Wikipedia, CNN, and others.

In order to make CAPTCHA breaking a hard problem for computers many CAPTCHAs inject noise to the images, rotates the letters or digits, varying their sizes, and adding structures like lines, grids, and bubbles following the guidelines given by Wilkins (2010). In addition, Wilkins recommends that CAPTCHA generators to also avoid characters which cannot be recognized easily for humans like the letter “O” and “Q” and the number “0”, letter “S” and number “5” that can be similar depending on the CAPTCHA font to prevent user confusion.

Therefore, loads of CAPTCHA breaking algorithms and recognition methods based in Deep Learning (Wang et al. 2019) appeared to verify the security provided by them, and improve the creation of more robust and secure CAPTCHAs. There exist various CAPTCHA recognition methods for text-based CAPTCHAs according to their scheme, usually artificial neural networks with different associated Deep Learning models.

Zhao, Liu and Jiang (2017) demonstrated that the efficiency of using Convolutional Neural Networks for learning and breaking CAPTCHAs in different scenarios for single letter

CAPTCHAs schemes achieving 99% of accuracy and a Multi-CNN (4 letters) achieving 76% of accuracy compared with a Clustering scheme achieving a 30% of accuracy and a Support Vector Machine using the one-vs-the-rest strategy for Multi-class classification achieving a 69% of accuracy.

Nouri and Rezai (2020) performed a deep neural network architecture named Deep Captcha, their model consisted of a Convolutional layer, a Max-Pooling, two Convolutional-MaxPooling, followed by a Dense layer and finally a Softmax layer. Their model uses five-digits CAPTCHAs generated using Python ImageCaptcha Library reaching an accuracy rate of 99.33% on the training set and 98.94% on the test set.

To increase the security of text-based CAPTCHAs from this automated model, websites and CAPTCHA generators have defined that a must have feature is to overlap CAPTCHA characters. Therefore, it appeared the “segment and recognize” approach where CAPTCHAs must go through a preprocessing process where the CAPTCHA will be segmented in individual character images and the accuracy of the model to break CAPTCHA will be limited by the segmentation process. Bursztein et al. (2014) created an algorithm that combines the segmentation and recognition step to solve CAPTCHAs and tested it with real-world CAPTCHA schemes achieving a 51.39% recognition rate on the CNN CAPTCHA dataset and a 55.22% on Baidu CAPTCHAs dataset enough rate to consider the CAPTCHA insecure but not reaching the accuracy of Neural Network models.

Tang et al. (2018) modified LeNet-5 Convolutional Neural Network model, they added an extra convolutional layer to the original model of three convolutional layers, two subsampling layers and two fully connected layers. They proposed a model to classify the CAPTCHAs according to their characters and collected random CAPTCHA Images from the 50 most popular websites with CAPTCHAs. Their success rate comes for each CAPTCHA

Scheme and goes from 10.1% to 90.0%. Bostick and Klecka (2018) developed a feedforward neural network, that can be used for pattern recognition and classification to target classes. This neural network consists of an input layer, two hidden layers, and output layer reaching a precision of 98.79% using Google ReCaptcha text Scheme.

We present an effective approach to automatically solve Text-Based CAPTCHAs. We developed two preprocessing schemes, one for alphanumerical CAPTCHAs and other for numerical CAPTCHAS. We developed a Convolutional Neural Network and performed a hyperparameter grid search to find the best combination for the model for each dataset of different CAPTCHAs schemes and a combined dataset of the retrieved Captchas, following the recommendation for future work by Nouri and Rezai to solve CAPTCHAs with variable length and alphanumerical CAPTCHAS.

MATHERIAL AND METHODS

Experimental Databases.

We acquired four text-based alphanumerical and numerical CAPTCHA datasets from Kaggle with different features. Table 1 shows a summary of each dataset features.

- The first dataset consists of alphanumerical CAPTCHA images with 4 characters generated using Python ImageCaptcha Library (see figure 1(a)). This python library gives the freedom to generate CAPTCHAs by setting the desired font style, background noise and distortion. We have developed a CAPTCHA generator using this library to create CAPTCHA images setting their character length and characters available.
- The second dataset consist of numerical CAPTCHAs that consist of 6 digits jailed CAPTCHAs with a red cut and some characters blurred (see figure 1(b)).
- The third dataset consist of 5-character alphanumerical CAPTCHAs with gray background strikethrough text (see figure 1(c)).
- The fourth dataset retrieved consist of 5 characters Alphanumerical CAPTCHAs with noisy background and strikethrough text some of the characters are slightly blurred, this CAPTCHA scheme is used in CNN (see figure 1(d)).

Database	**Resolution**	**Characters**	**Alphanumerical**	**Quantity**
Dataset 1	160-by-60 pixels	4	✓	24000
Dataset 2	182-by-50 pixels	6	✕	5000
Dataset 3	200-by-50 pixels	5	✓	1998
Dataset 4	180-by-50 pixels	5	✓	1066

Table 1. Datasets Characteristics

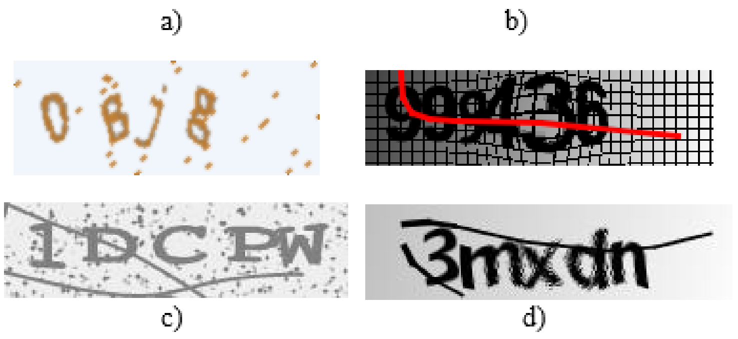

Figure 1. Sample images of datasets: a) Dataset 1, b) Dataset 2, c) Dataset 3, d) Dataset 4.

Artificial Neural Networks.

In order to break the CAPTCHAs acquired, we developed a variation of the LeNet-5 CNN model (Zhang, 2018). We added an extra Convolutional Layer varying the number of filters, the number of pools for the MaxPooling Layers and activation functions. The model consisted of two Convolutional layers (the activation function defined for this layer is Relu) followed by a MaxPooling Layer, followed by two Convolutional layers with Relu activation function, followed by a MaxPooling Layer, a Dropout Layer to prevent over-fitting, a Flatten layer followed by a fully connected layer, a Dropout Layer and finished with a fully connected layer with the number of classes detected as defined output.

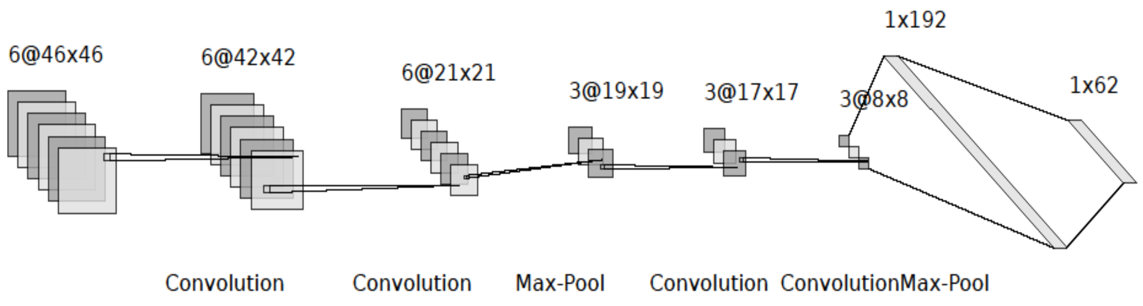

Figure 2. The Architecture of the proposed Convolutional Neural Network (scale 1:10).

Database preprocessing.

CAPTCHA image preprocessing is a series of steps to prepare the images used to train and evaluate the Convolutional Neural Network. The preprocessing method was divided in two different methods for the Alphanumeric datasets and the Numeric dataset. For the Alphanumeric CAPTCHA images, we converted the image color space from RGB to gray scale. Afterwards, we use a threshold segmentation method to remove the background noise. We are binarizing the image using the OTSU's algorithm (Gonzalez & Woods, 2002) that automatically calculates a threshold value for the image to extract the foreground of the image. This optimal global threshold value is found given by equation 1.

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t) \tag{1}$$

where $\omega_{0,1}(\mathrm{t})$ is calculated from the histogram bins with:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(\mathrm{t}) = \sum_{\mathrm{i=t}}^{\mathrm{L}-1} \mathrm{p(i)}$$

maximizing their inter-class variance:

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2$$

$$= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2$$

which is expressed in class probabilities ω and class means μ, where the class means $\mu_0(\mathrm{t})$, $\mu_1(\mathrm{t})$ and μ_T are calculated by:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

Then we applied an opening morphological transformation (Fisher et al. 1996) to the image which is defined as erosion followed by dilatation. Erosion removes the pixels at the boundaries and decreases the size of the foreground, in this case the letters. This operation we erase background noise. Dilatation is the opposite operation of erosion and increments the foreground object size and it is used to join broken parts of the foreground due to erosion operation. This process was applied to Dataset 1, Dataset 3, and Dataset 4 as shown in figure 3(a). In the case of Dataset 2 (Numerical Captchas) we developed other preprocessing. We had to deal with the red cut line of the CAPTCHA. Hence, we started with converting the red color to black. We use an adaptive thresholding using the mean of neighborhood area to decide the value of conversion of each pixel, blurring the image and opening morphological transformation as shown in figure 3(b).

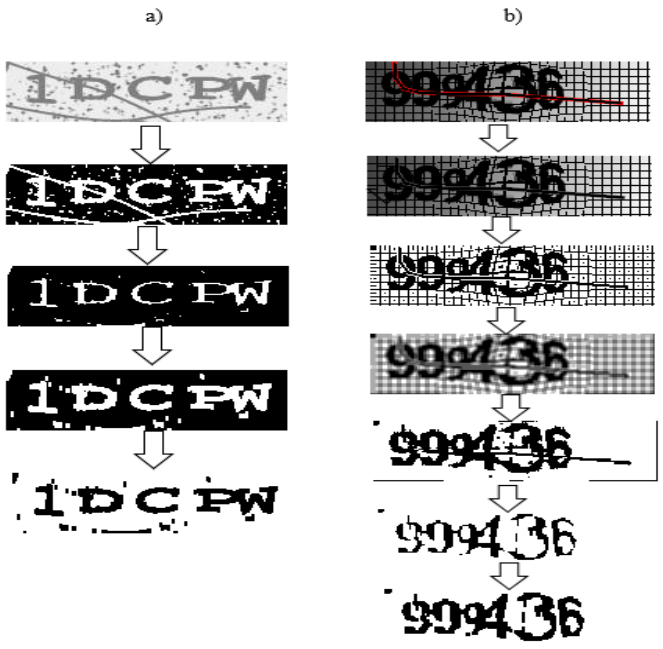

Figure 3. Preprocessing CAPTCHAs process for a) Alphanumerical and b) Numerical CAPTCHAs

After preprocessing the image, the CAPTCHA Characters will be cropped in individual characters. We use the connected components method for segmentation and a Watershed algorithm (Gonzalez & Woods, 2002). This algorithm works treating the image as a topological surface where white areas represent peaks and black areas valleys. To determine segment boundaries, the algorithm floods basins from the markers until basins attributed to different markers meet on watershed lines.

These watershed lines should match the CAPTCHA character length for each dataset, if not, we implemented an arbitrary pixel count for each mask to check if two characters are in the same mask if pixel value is greater than 2000 pixels. If this is the case, we split the mask in the middle. Using the mask, we segmented the original image in order to get individual character images. After preprocessing the CAPTCHA Images, we segmented every CAPTCHA character as a single image with black background and white letters as shown in Figure 4. We performed this process for every dataset showing the results in Table 2. We created a large dataset consisting of the combination of the four datasets.

Database	**Resolution**	**Correct Segmented CAPTCHAs**	**Bad Segmented CAPTCHAs**	**Single Character Images**
Dataset 1	50-by-50 pixels	22890	1110	91560
Dataset 2	50-by-50 pixels	4214	786	21070
Dataset 3	50-by-50 pixels	1659	339	8295
Dataset 4	50-by-50 pixels	629	437	3145
Mixed	50-by-50 pixels	29392	0	124070

Table 2. Preprocessing Results.

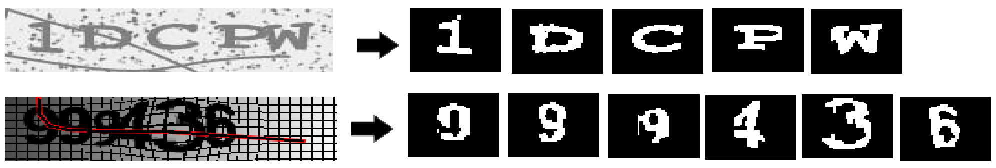

Figure 4. CAPTCHA Segmented.

Training and testing.

We created train and test sets of images and a validation set of images. 30% of the total images are used for testing, 30% of the testing images are used for validation and 70% are used for training the network. We are dealing with categorical labels therefore we transformed our labels using One Hot Encoding (Casari & Zheng, 2018) into vector of zeros and ones with a length equal to our number of classes, in this case the number of characters found in the datasets. Every element in this vector will be zero except for the element that correspond to that category. This category will be unique and will help the neural network to do a better job while predicting.

Since we want to find the best hyperparameters for our Deep Learning Model, we use a GridSearch strategy for hyperparameter tuning. This method is based in build a model for each possible combination of the parameter we want to vary and find the best combination of them. We selected the following parameters: batch size, number of epochs, and optimizer function. We performed this task to the model for every dataset we retrieved and for the mixed dataset. Convolutional Neural Networks are sensitive to the batch size; therefore, we vary the batch size using 100, 150, 200, 220 and 240. The number of epochs changed from 100 to 200 in steps of 20 and finally we tuned the Optimization Algorithm using Adagrad, Adam, Adamax and Nadam. These configurations created 100 different models and we used 3 Cross fold Validation to find the best one. For the mixed dataset we vary the batch size using 100, 125, 150, 175, 200, 220 and 240. The number of epochs changed from 80 to 200 in steps of 20 and finally we tuned the Optimization Algorithm using Adagrad, Adam, Adamax and Nadam. Creating 196 different models to find the best combination.

Selection criteria.

After our GridSearch process we selected the best model reported for each dataset based in the test accuracy achieved.

Results and Discussion.

The CAPTCHA preprocessing process was successful for the majority of the CAPTCHA Images. We kept the count of wrong preprocessed CAPTCHAs giving us a total of 2672 failed CAPTCHA Images preprocessed of the total of CAPTCHA Images retrieved of our dataset. We retrieved the best hyperparameters for every dataset, accuracy during training, test accuracy and Area Under the Curve Receiver Operating Characteristics (AUC-ROC) results showed in Table 3. We plotted the accuracy per epoch for the validation and training set for the mixed dataset (see figure 5). The gap between the training and validation curve suggests little overfitting.

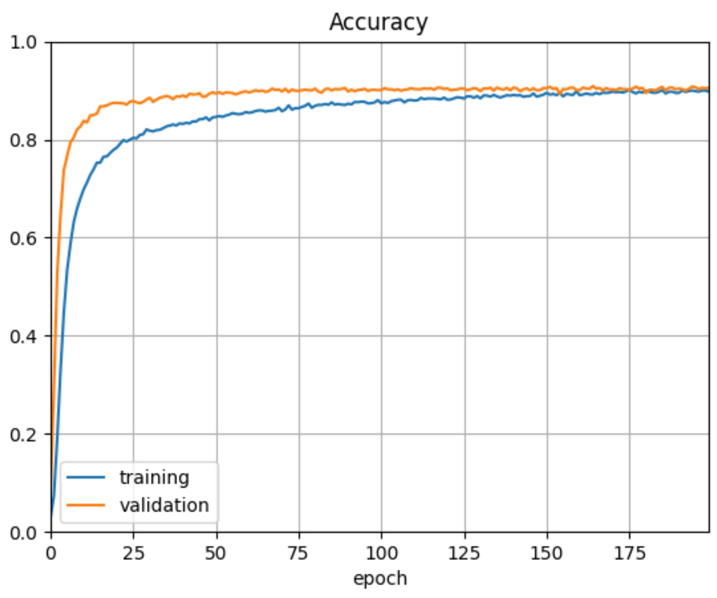

Figure 5. Accuracy vs Epoch plot Mixed dataset.

Database	Batch size	Epochs	Optimizer	Training Accuracy	Test Accuracy	One-vs-Rest ROC AUC score
Mixed	125	200	Nadam	91.23%	90.49%	0.9974
Dataset 1	100	200	Nadam	88.68%	87.80%	0.9927
Dataset 2	150	200	Adam	94.12%	93.90%	0.9938
Dataset 3	220	160	Nadam	98.72%	97.10%	0.9981
Dataset 4	220	160	Adamax	92.24%	93.48%	0.9916

Table 3. Comparison Results

With our model we obtained a multi-class classification problem we used AUC-ROC curve. We use this metric to check the performance of the model, by checking how much each model is capable of distinguish between classes. The AUC-ROC curve is used only in binary classification problems. We will use an extension of this metric because of our multi-class classification, using a One vs Rest technique. We will retrieve this value with the macro average that average the performance of the individual true positives, true negatives, false positives, and false negatives of each class for every dataset. We graph the curve for the dataset 4 as shown in figure 6.

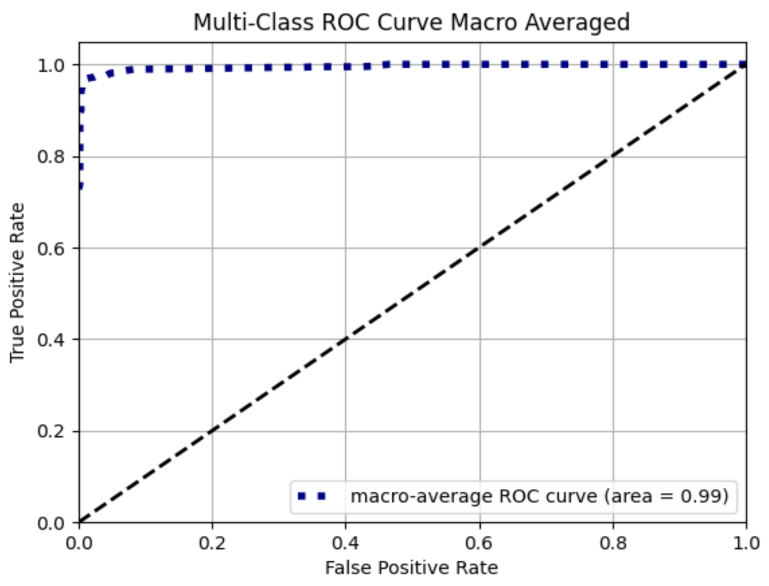

Figure 6. Multi-Class ROC Curve Macro Averaged for dataset 4.

Our best model achieved a 97.10% of accuracy in the third dataset of Five Character CAPTCHA with gray background. This dataset consisted of characters which are not rotated and not overlapped. Hence, our segmentation algorithm worked better, and the network obtained better results. Furthermore, we developed a pipeline of prediction to test the mixed model with new CAPTCHAs that the model never saw. We generated one hundred 6-character CAPTCHAs. We tested this CAPTCHAs with the model trained with the mixed dataset achieving an accuracy of 75%. In most of the failed CAPTCHA prediction the model predicted wrong only one character. The model confused the character “B” with the number “8”, the character “S” and “s” with the number “5” and finally the character “l” with “I”. These cases were common mistakes in most of the models proposed.

We trained a model with a dataset of 12,576 taking 3,144 single character images from every dataset to have a more even mixed dataset. We compared the validation accuracy from the model trained with the mixed dataset and the even dataset (see Table 4).

Dataset used for training	**Validation Accuracy**
Mixed	85.15%
Mixed-even	93.52%

Table 4. Validation Accuracy of mixed models.

CONCLUSIONS AND FUTURE WORK

A CAPTCHA is considered broken if it can be automatically solved at a rate above 1% (George et al. 2017). We designed our CNN to measure the strength and weakness of our CAPTCHAs retrieved. We achieved up to 97.10% accuracy for the dataset 3. For the dataset 4 of CAPTCHAs that follow the CNN scheme for security, we achieved an accuracy of 93.48%, which represent an improvement to the 51.39% achieved by the algorithm proposed by Bursztein et al. (2011). For the dataset 1 the model achieved an accuracy of 87.80% and for the dataset 2 achieved an accuracy of 93.90%.

Overall, the accuracy achieved by our approach goes from 87.80% to 97.10% which is enough to consider these CAPTCHA datasets broken. Our results also suggest that the performance of our network could be improved with the correct segmentation process, our preprocessing was very accurate, but it showed weaknesses with the dataset 4 which scheme has their characters more overlapped that the rest of datasets.

There is an AUC of 0.99, this suggest us that there is an 99% chance of a correct prediction when giving the model a CAPTCHA Character image. Our model achieved an accuracy of 90.49% with the mixed dataset with the different type of CAPTCHAs. This suggest that adding the frequency of characters with different font types and sizes improve the efficiency of the model. Therefore, improving the segmentation process for overlapping characters of the CAPTCHA and feed the network with more images for training will increase the accuracy of prediction.

As a recommendation for future work, we suggest the use of neural networks for the segmentation process in order to increase the number of correct segmented CAPTCHAs and the accuracy of the models. Furthermore, we plan to explore the use of recurrent neural networks instead of convolutional neural networks for the character recognition process.

REFERENCES

Bostik, O., & Klecka, J. (2018). Recognition of CAPTCHA characters by supervised machine learning algorithms. *IFAC-PapersOnLine, 51*(6), 208-213.

Bursztein, E., Aigrain, J., Moscicki, A., & Mitchell, J. C. (2014). The end is nigh: Generic solving of text-based captchas. In *8th {USENIX} Workshop on Offensive Technologies ({WOOT} 14)*.

Bursztein, E., Martin, M., & Mitchell, J. (2011, October). Text-based CAPTCHA strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 125-138).

Casari, A., Zheng, A. (2018). *Feature engineering for machine learning: principles and techniques for data scientists*. " O'Reilly Media, Inc.".

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (1996). Hypermedia image processing reference. *England: John Wiley & Sons Ltd*, 118-130.

George, D., Lehrach, W., Kansky, K., Lázaro-Gredilla, M., Laan, C., Marthi, B., ... & Phoenix, D. S. (2017). A generative vision model that trains with high data efficiency and breaks text-based CAPTCHAs. *Science*, *358*(6368).

Gonzalez, R. C., & Woods, R. E. (2002). Digital image processing.

Nouri, Z., & Rezaei, M. (2020). Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment. *Available at SSRN 3633354*.

Stark, F., Hazırbas, C., Triebel, R., & Cremers, D. (2015, October). Captcha recognition with active deep learning. In *Workshop new challenges in neural computation* (Vol. 2015, p. 94).

Tang, M., Gao, H., Zhang, Y., Liu, Y., Zhang, P., & Wang, P. (2018). Research on deep learning techniques in breaking text-based captchas and designing image-based captcha. *IEEE Transactions on Information Forensics and Security*, *13*(10), 2522-2537.

Wang, J., Qin, J., Xiang, X., Tan, Y., & Pan, N. (2019). CAPTCHA recognition based on deep convolutional neural network. *Math. Biosci. Eng*, *16*(5), 5851-5861.

Wilkins, J. (2010). Strong captcha guidelines.

Zhang, Q. (2018, January). Convolutional neural networks. In *Proceedings of the 3rd International Conference on Electromechanical Control Technology and Transportation* (pp. 434-439).

Zhao, N., Liu, Y., & Jiang, Y. (2017). CAPTCHA Breaking with Deep Learning.