Text-based Captcha Recognition

Second Assignment of the Curricular Unit TAA.

João Pedro Saraiva Borges
DETI
Universidade de Aveiro
Nº Mecanográfico 98155

Daniela Filipa Pinto Dias
DETI
Universidade de Aveiro
Nº Mecanográfico 98039

***Abstract*—A Captcha is a type of challenge-response test used in computing to determine whether the user is human [1]. The challenge consists of obscuring a message from computer interpretation, for example, by twisting the letters and adding a slight background color gradient. It was designed to be easily recognized by humans and difficult to identify by machines or robots.**

Captcha verification was born from a need to prevent harmful Internet attacks and increase the security of websites, taking multiple shapes, such as text-based, picture-based, and sound-based. However, there are multiple approaches to solving a Captcha, being one of them by implementing an automated solver with machine learning. We'll take this approach in this paper.

The focus of this paper is to compare several common machine learning classification algorithms for Character Recognition of Captcha codes and, finally, to propose our solutions.

***Index Terms*—machine learning; Captcha; text-based Captcha; neural network, convolutional neural network; deep learning;**

I. Introduction

After the many technological breakthroughs we've experienced in the last decades, computer vision has been one of the major branches of cybernetics that keeps growing and increasing persistently with new use cases and applications. There are many diverse applications for image processing and image retrieval methods (inherent to computer vision), in particular within the Internet security branches. One of the most known, which we will study in this paper, is Captcha recognition, specifically text-based Captcha recognition.

With the increase of automated bots systems and software that misuse the public web services, the user is often required to solve a challenge before they gain access to the website. This challenge is called CAPTCHA (short for "Completely Automated Public Turing test to tell Computers and Humans Apart"). It is a Turing test designed to determine whether a user is a computer or a human.

Different types of Captchas - including text-based, image-based, and audio-based Captchas -, have proven to be effective against machine attacks, keeping websites safe (i.e. providing security against bots). Text-based Captcha is the most widely used type of Captcha, due to its simplicity, extensive usability, easy implementation, and low cost. It's nothing more than an image that contains alphanumeric characters with some distortion and degradation, along with a slight background color gradient.

Its implementation is usually based on OCR (Optical Character Recognition), a technology that identifies characters from printed books, handwritten papers, or images, enabling the transfer of documents into digital systems [2]. In other words, the image is prepared in a way that uses known OCR issues against the computers, making it difficult to recognize the text.

However, with the significant advancements in deep learning, it has become much easier to build systems that can efficiently recognize text-based Captchas without any kind of human interaction. Research on recognizing Captcha images has expanded in order to identify weak points and loopholes in the generated Captchas, consequently leading to the avoidance of these loopholes in newly designed Captcha-generating systems.

Some techniques and mechanisms were applied to improve the security of Captcha, such as background noise, anti-segmentation techniques (waviness and horizontal stroke crossing the letters), merging of characters, and variable keyword length, which makes it more difficult to guess the position of individual letters. We'll be dealing with a dataset made of Captcha images with these characteristics.

However, with the dramatic advancements in deep learning, Captcha recognition systems have become more competent than before in recognizing text-based Captchas, despite all the newly added defense mechanisms. As a result, sophisticated security mechanisms need to be developed to make text-based Captchas more robust against malicious attacks.

Hence, in this study, we compare different approaches for Captcha image recognition and introduce other models capable of solving text-based Captchas.

II. Related Work

Before deciding on an approach to address the issue of Captcha Recognition, we reviewed other references (papers, reports, and blogs) handling the same or similar problems. While our dataset consists of text-based Captcha, we also found it useful to study techniques present in handling picture-based Captcha.

A. Captcha Recognition Using Deep Learning with Attached Binary Images

In the article "Captcha Recognition Using Deep Learning with Attached Binary Images" [3], the authors first go

into detail about the different available text-based Captcha recognition algorithms and their respective categories. Here, we learn we can take two different routes for our solutions: segmentation-based and segmentation-free.

Segmentation-based algorithms typically involve two main steps: segmentation (the Captcha image is segmented into individual characters that are fed to the character recognition module) and character recognition. The segmentation step must be done correctly because it can considerably affect the overall accuracy and efficiency of the entire system. However, most segmentation algorithms do not perform well and suffer from low efficiency and efficacy.

Segmentation-free algorithms (typically deep-learning-based) can directly recognize and classify the characters of Captcha images without the need for segmentation. While this type of model demonstrates high accuracy and efficiency, it needs large datasets to extract features efficiently. With our kind of dataset, it's not possible to implement this algorithm without sacrificing its accuracy, since we do not have enough samples.

With these concepts in mind, the authors propose a new, simple and efficient deep-learning-based Captcha breaking system with low complexity. They avoid the segmentation step by using the proposed attached binary images algorithm, which consists of:

- Making several copies of the Captcha image equal to the number of characters on Captcha.
- Creating several external distinct binary images equal to the number of characters on Captcha.
- Attaching the first distinct binary image to the first Captcha copy, the second binary image to the second copy, and so on.
- Adding a label that represents the character class to each Captcha copy.
- Using Captcha copies with their labels to train a convolutional neural network (CNN) model to classify Captcha characters.

Here, the characters of the Captcha input images can be directly and smoothly recognized and classified without segmentation by using the proposed attached binary images algorithm.

The authors used two different datasets: Weibo Captcha Scheme and Gregwar Captcha Scheme. On the resultant Weibo dataset, the model reached a testing total character recognition accuracy of 97.89%, such that 39,156 characters out of the 40,000 resultant testing set characters were recognized correctly. It also reached an overall Captcha testing accuracy of 92.68%, with 9268 Captchas out of the 10,000 original testing set recognized correctly.

On the resultant Gregwar dataset, the model achieved a testing total character recognition accuracy of 85.28% with 34,111 characters out of the 40,000 resultant testing set characters recognized correctly. It also obtained an overall Captcha testing accuracy of 54.20%, with 5420 Captchas out of the 10,000 original testing set recognized correctly.

B. Recognition of Captcha Characters by Supervised Machine Learning Algorithms

In the article "Recognition of Captcha Characters by Supervised Machine Learning Algorithms" [4], the author goes in-depth on how a Captcha is based on an OCR (Optical Character Recognition) problem, and that current OCR algorithms have many weaknesses, but still are very robust, with great performance when recognizing Captcha images.

The author also described how a Captcha is created, stating that one of the key elements to a successful text-based Captcha scheme is the use of multiple fonts. The author took this idea and improved it by developing a generator that would dynamically generate a font for every character of the Captcha.

The bubble Captcha concept was implemented to create this Captcha generator. The bubble Captcha concept is a two-dimension array representing a binary grid of an elementary bi-color Captcha scheme with randomly positioned circles/bubbles forming the font. Here, every character used is one of the key elements entering the generator algorithm to create the image. The generator also changes slightly the position of the bubbles and their size when generating the Captcha.

The first approach taken by the author was the K-Nearest Neighbors, which was the fastest method to learn; however, in contradiction, it was the slowest method during the classification phase because of the exhaustive distance computing, having a precision of 98.99%. Decision tree and SVN methods were also used, with almost perfect precision.

The Pattern Recognition Artificial Neural Networks was also implemented, which is a feed-forward neural network that can be used for pattern recognition and classification to target classes. It uses 1 input layer, two hidden layers, and an output layer: the two hidden layers utilize the Nguyen-Widrow initialization method, and both have a transfer function of hyperbolic tangent sigmoid. These two layers also use the summation of weight and biases as an input function. This method had 100% of precision.

Although all the methods implemented by the author had a precision of around 99%, the main difference lies in the computational cost; the overall best algorithm would be the pattern recognition neural network.

C. A deep learning-based attack on text Captchas by using object detection techniques

In the article "A deep learning-based attack on text Captchas by using object detection techniques" [5], the authors use an end-to-end method for breaking text Captcha by resorting to an object detection technique that combines a residual network (Residual neural network) and a feature pyramid network (FPN), for feature extraction, and a regional proposal network (RPN), for character localization. The entire framework developed in this report consists of three main parts:

- A feature extracting module: cleans the image and makes it more perceptible for the machine learning algorithm to classify.

- A character location and recognition module: it applies a coordinate matching module to perform character recognition and sorting.
- A coordinate matching module: Finds the coordinates of each character and sorts it so that the sequence is correct.

This report also has a real-time execution of the method created in real sites, where sites like google and amazon that have captcha verification were tested to see if the machine learning process could automatize this process. For that, the coordinate matching module was needed so that some Captcha that have their characters in different heights in the image would be sorted correctly.

The approach taken for the recognition module was the mask region-based convolutional neural network. This includes a mask branch, in which items that have the same label are marked with different masks, and these masks are removed at the end so that the output doesn't come as a different character. The R-CNN is a CNN that takes the input image and produces a set of bounding boxes as output, where each bounding box contains an object and also the category of the object.

A sample size of 1000 is small among deep learning techniques; however, with 1000 samples, the model developed by the authors achieved a success rate of over 80% on more than half the test schemes.

D. Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment

In the article "Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment" [6], the authors take a similar approach to the one being used in this report. The authors apply pre-processing steps to the Cpatcha image, where they reduce the image size, set the image to a gray scale coloring, and perform a noise reduction algorithm. This algorithm contain a conventional median-filter, which removes the median value of the surrounding pixels values instead of the pixel itself. Overall, the CNN developed by the authors obtained great results, with an accuracy of around 99.3%. It is said that while tuning the CNN, the use of a series of paralleled softmax layers played a huge role in the success and accuracy improvement of the CNN.

III. Dataset

A. The Dataset

The Data Set used for analysis in this project was obtained from the website Kaggle, with the name CAPTCHA Images. Our dataset contains Captcha images with exactly 5 alphanumeric characters with some distortion, noise, and blur applied to them. The images also have a single line crossing the characters, one of the defense mechanisms previously mentioned to interfere with the recognition process. We have, in total, 1070 samples in different image formats. Each image contains 50 x 200 pixels. However, we'll only consider png images, in order to simplify the data preprocessing stage, making it 1040 samples (and 5200 characters).

B. The Composition of the Dataset

The dataset composition can quickly be summarized to:

- The dataset contains 1040 files.
- Each file is an image representing a CAPTCHA image.
- The image is either in png format (1040 png files) or in jpg format (30 jpg files).
- The file name is composed of the 5 characters contained in the CAPTCHA image followed by the image format (e.g. 34pcn.png).
- Each image has 50x200 pixels.
- The shape of the array differs between jpg and png images indicating that jpg images are RGB and png are RGBA.

We've decided to divide our dataset into 3 different subsets: training set, cross validation set and test set. The training set will be used to fit and train our models. The cross validation will be used to provide an unbiased evaluation of the models fit on the training dataset while tuning model hyperparameters. Finally, the test set will be used to provide an unbiased evaluation of the final models fit on the training dataset.

Since our dataset is relatively small (under 10000 samples), we've decided to set the dataset split ratio as such:

- 60% - Training set
- 20% - Cross Validation set
- 20% - Test set

In other words, we'll have: X_train with 624 images, y_train with 624 labels, X_val with 208 images, y_val with 208 labels, X_tests with 208 images and y_test with 208 labels.

C. Data Preprocessing Summary

In order to improve the performance of our models, we have taken multiple preprocessing steps to construct the final input for the machine learning algorithms. We normalized the data, removed all the noise applied to the images (by smoothing out the images and removing the stroke crossing the characters) and separated out each one of the 5 letters in the image (i.e. segmentation, a technique we mentioned previously). The steps taken for the image smoothing were the following ones:

- Adaptive Thresholding - An algorithm that defines clearly the edges of an image, in this study case, the varying illumination and blured zones in the captcha are cleared.

Fig. 1. Tresholding application in a Captcha

- Closing - It is useful in closing small holes inside the foreground objects, or small black points on the object

Fig. 2. Closing application in a Captcha

- Dilation - It increases the margin of the letters drawn in the Captcha, the algorithm puts a pixel as black if the pixel under has the same value.

Fig. 3. Dilation application in a Captcha

- Image Smoothing - It smooths the image by passing a filter that removes high frequency components ie. edges and noises from the image.

Fig. 4. Image Smoothing filter application in a captcha

After all these steps, the characters are ready to be fed to the models individually.

D. Statistical Analysis

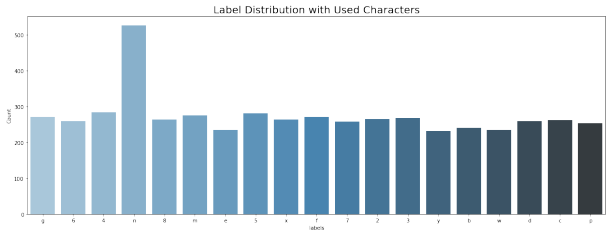

Fig. 5. Label Distribution with Used Characters

It's worth noting that the distribution of occurrences of each digit and letter isn't uniform - not all characters in the Latin alphabet and not all digits are used in the Captcha texts. However, for the used characters and digits, the frequency of each character is roughly the same with the exception of the character n.

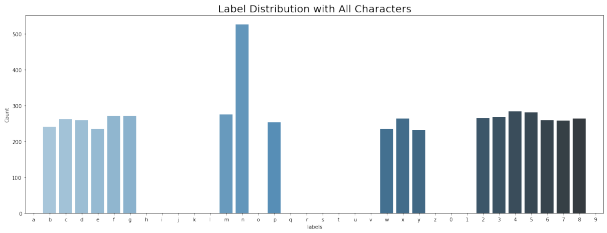

Fig. 6. Label Distribution with All Characters

As we can see, the following characters are not present: a, h, i, j, k, l, o, q, r, s, t, u, v, z, 0, 1, 9. Only 19 characters are used: 2, 3, 4, 5, 6, 7, 8 and b, c, d, e, f, g, m, n, p, w, x, y.

IV. Methodologies

In this paper, we will implement two different algorithms - one based on machine learning and another based on deep learning - to solve the text-based Captcha images.

We've chosen to implement a linear Support Vector Machine, SVM, which is a supervised learning method used for classification, regression, and outliers detection. It works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.

We've also chosen to implement a Convolutional Neural Network method, CNN, which is a deep learning algorithm that obtains the best results for this study case. There is a significant number of people who have tried to solve the Captcha recognition problem with this method, hence, we plan on comparing our results in order to recognize the flaws in our algorithms and what went better than other related works.

A. Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. They work by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.

SVC implements the "one-versus-one" approach for multi-class classification. In total, n_classes * (n_classes - 1) / 2 classifiers are constructed and each one trains data from two classes.

We tried different regularization parameters, ranging from C = 1 to C = 5. The strength of the regularization is inversely proportional to C. The penalty is a squared l2 penalty.

Using the data previously processed, the best accuracy results that we could obtain were around 81.6%, with C = 1, which is quite satisfactory but could be improved. The method shows less accuracy when predicting the letter "n", the letter "m" and some other characters.

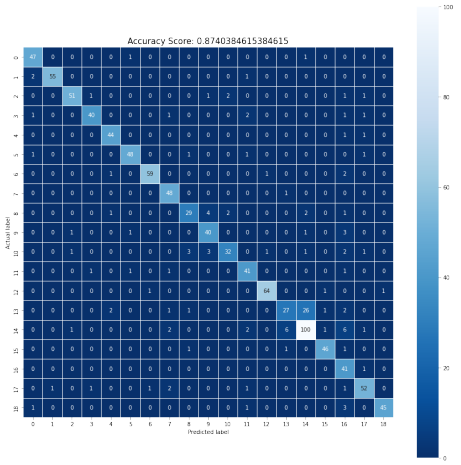

Fig. 7. Accuracy matrix for each character using the Support Vector machine

B. Convolutional Neural Network

A convolution neural network is similar to a multi-layer perception network.The major differences are what the network learns, how they are structured and what purpose they are mostly used for.

Convolutional neural networks were also inspired from biological processes, their structure has a semblance of the visual cortex present in an animal. CNNs are largely applied in the domain of computer vision and has been highly successful in achieving state of the art performance on various test cases.

The hidden layers in a CNN are generally convolution and pooling (downsampling) layers. In each convolution layer, we take a filter of a small size and move that filter across the image and perform convolution operations. Convolution operations are nothing but element-wise matrix multiplication between the filter values and the pixels in the image and the resultant values are summed.

1) Implementation: A CNN architecture is developed by a stack of different layers that convert the input volume into an output volume through differentiable functions. A few different types of layers are commonly used, such as:

- Input layer
- Convolutional layers
- Pooling layer
- Fully connected layer

In the convolutional layers, we use *conv2d* function from *keras*, which is a 2D convolution layer. After that we normalize the batch and use the function *dropout* to regularize the neural network by ignoring randomly selected neurons, preventing overfitting. Finally, we include a pooling layer using the function *MaxPooling2D*. We use three convolutional layers, where the only difference lies in the change of amount of filters used. Then we apply the function *flatten* to flatten our pooled feature map into a column so that we can insert the data into an artifical neural network. Next we apply two dense layers, which is a layer that is deeply connected with its preceding layer, by connecting with every neuron of the preceding layer; like before, the two dense layers only change in the amount of filters used, the dense layer also applies batch normalization and the *dropout* function. Finally, we use the softmax layer to convert the vector of values into a vector of probabilities, which the sum of them must add up to 1.0.

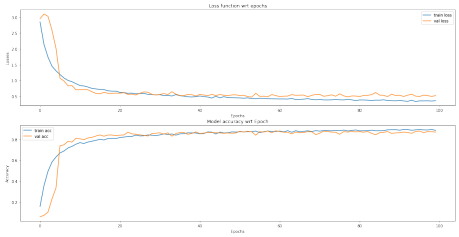

Fig. 8. Loss function evolution and Model accuracy through epochs

The CNN model starts to stabilize and reach the best values for loss and accuracy at 20 epochs, as we can see in the figure 8.

2) Results: The accuracy obtained is around 87%, which is slightly lower than what we expected. We looked for the source of the problem and came to the conclusion that the neural network often fails to recognize some specific characters. For example, the numbers ”8” and ”9” are guessed correctly only 83% of the times, while the accuracy is even lower for the characters ”m” (label 13) and ”n” (label 14), guessing correctly only 77% and 76% of the times, respectively. We can see this in the following images.

	precision	recall	f1-score	support
0	0.90	0.96	0.93	49
1	0.98	0.95	0.96	58
2	0.94	0.89	0.92	57
3	0.93	0.87	0.90	46
4	0.92	0.96	0.94	46
5	0.94	0.92	0.93	52
6	0.97	0.94	0.95	63
7	0.87	0.98	0.92	49
8	0.83	0.74	0.78	39
9	0.83	0.87	0.85	46
10	0.89	0.73	0.80	44
11	0.84	0.91	0.87	45
12	0.97	0.96	0.96	67
13	0.77	0.45	0.57	60
14	0.76	0.84	0.80	119
15	0.94	0.94	0.94	49
16	0.62	0.98	0.76	42
17	0.88	0.88	0.88	59
18	0.98	0.90	0.94	50
accuracy			0.87	1040
macro avg	0.88	0.88	0.87	1040
weighted avg	0.88	0.87	0.87	1040

Fig. 9. Accuracy table

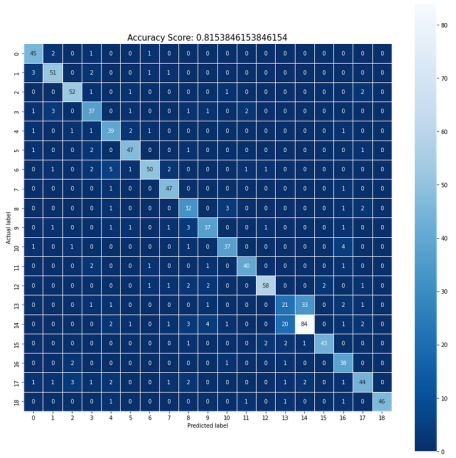

Fig. 10. Accuracy matrix for each character using the Convolutional neural network

Research was made to see if we could better understand this problem, and we discovered that most papers that implement Captcha recognition with deep learning faced the same problem, in the same characters [6]. So, this clearly presents a weakness to this CNN structure, which makes it vulnerable to errors when predicting some characters (like the character "n").

V. Conclusions

Comparing to the results obtained by the other papers analysed in the related work section, we can conclude that our results are less accurate than other implementations.

In the "Recognition of Captcha characters by supervised machine learning algorithms" [4], the implementation of the pattern recognition artificial network obtained an accuracy of 100%, while our CNN had an accuracy of 87%. From the same report, we can also compare the SVM, which had an accuracy of 99.80%, while ours had an accuracy of 81.6%. This was due to a more complex implementation of these models by the authors. The "Captcha Recognition Using Deep Learning with Attached Binary Images"[3] also implemented the CNN algorithm, which also obtained slightly better results than us, with 96.03% of accuracy on Weibo Captcha.

We should also take note that the CNN model took much more time and computational power to be fit than the SVM model.

Yet, we also learned that certain strategies could be applied to improve our algorithms, such as data augmentation (with the *ImageDataGenerator* function from *keras* by applying a rotation range of 5 and a width shift range of 4, for example). Also, it should be taken into account that, depending on the amount of data in the dataset, the CNN algorithm can be compromised because of the amount of time taken to process everything. The CNN could also be compromised by not using more paralleled softmax layers as proposed in the "Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment" report [6]. Also, other people that developed recognition algorithms with this dataset had similar problems when trying to recognize certain characters like the "n" and "m". This could be a general problem that algorithms face when trying to recognize this problem.

We can conclude that, although our algorithms lack some optimization, they still demonstrate that it's possible for a computer to recognize Captcha images. We can also affirm that Convolutional Neural Networks are extremely suitable for this type of problem, although sometimes its computational demands can be an obstacle. Taking this into account, there are always other algorithms like the Support Vector Machine method, which demand lesser computational power.

References

[1] Wikipedia, Captcha. Available: https://en.wikipedia.org/wiki/Captcha
[2] Recognition of Captcha Characters by Supervised Machine Learning Algorithms. Availble: https://www.sciencedirect.com/science/article/pii/S2405896318309017
[3] Captcha Recognition Using Deep Learning with Attached Binary Images. Available: https://www.mdpi.com/2079-9292/9/9/1522/htm
[4] Recognition of Captcha Characters by Supervised Machine Learning Algorithms. Available: https://www.sciencedirect.com/science/article/pii/S2405896318309017
[5] A deep learning-based attack on text Captchas by using object detection techniques. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/ise2.12047
[6] Deep-CAPTCHA: a deep learning based CAPTCHA solver for vulnerability assessment. Available: https://www.researchgate.net/publication/342197884_Deep-CAPTCHA_a_deep_learning_based_CAPTCHA_solver_for_vulnerability_assessment