Breaking Text-based CAPTCHAs with Convolutional Neural Networks

Diogo Daniel Ferreira, Luís Leira
Department of Electronic, Telecommunications and Informatics
University of Aveiro

***Abstract*—CAPTCHAs are essential for service providers to separate the human users from bots. In 2014 text-based CAPTCHAs were considered insecure and other approaches were explored, mainly relying on object recognition. However, many websites still use text-based CAPTCHAs as the main way to distinguish humans from machines.**

In this report, it will be shown how to break a text-based CAPTCHA by automatically reading the characters in the image. It is performed image transformation to remove the noise in the image, character segmentation to extract the characters and different approaches are taken to the construction of a model capable of recognizing each character. This report focuses particularly on the construction of a Convolutional Neural Network with different techniques to improve its accuracy.

After the network being trained, it is also visualized the learning done by the convolutional layers with three particular techniques: Calculation of the activations of the convolutional layers given an input image, visualization of the patterns that the convolutional layers are detecting and creation of heat maps to detect what part of the character is most important for the classification made.

In the dataset used, our model achieves an accuracy of 93.91% in character recognition, proving that with an accurate character segmentation and robust noise removal techniques applied to the images, the text-based CAPTCHAs can be broken by a machine.

I. Introduction

CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) is an automated test designed to check if the user is human. That test is made using a challenge-response approach, where the challenge is easy for a human to solve, but hard for a machine. If the response is correct, the machine assumes that the user is human.

CAPTCHAs are used by most service providers, such as email or online shopping, to prevent bots from abusing their online services. For example, to prevent a botnet to create hundreds of new email accounts per second, a CAPTCHA can be used to assure that the users creating the email accounts are humans.

The construction of CAPTCHAs is hard because it is uneasy to create challenges hard for machines but easily solvable by humans. Over the years, the most used CAPTCHAs are based on visual-perception tasks. Usually, are presented distorted characters that must be typed correctly by the user. Most of the times, it is also added background and foreground noise to the created CAPTCHAs, making it almost impossible for a computer to automatically recognize the characters. However, for humans, the characters are usually recognizable, due to our brain's capacity for recognizing patterns.

There are three characteristics for a modern text-based CAPTCHA to be resilient [1]:

- The large variation in the shape of letters. While there is an infinite variety of versions for the same character that the human brain can recognize, the same is not true for a computer. If all the versions of a character are different, it is hard for a computer to recognize any version not previously seen.
- Due to the large variation in the shape of characters, it can also be hard to perform segmentation for each character, mainly when the characters have no space in between.
- In specific CAPTCHAs, the context may be the key to answer correctly to the task. When the word is taken into context, it is easier for a human to answer what are the characters in the challenge, even if some of them are dubious.

The conjugation of these three characteristics makes a CAPTCHA hard to solve by a machine. In 2014, the authors of [2] concluded that text distortion-based CAPTCHAs schemes should be considered insecure due to technological advances. Since then, alternative CAPTCHA schemas based on object recognition were proposed ([3], [4]), making it harder for machines to solve them, but remaining easily solvable by humans.

However, there are still many implementations of insecure text-based CAPTCHAs on the web. This report explains a possible attack that can be made to a text-based CAPTCHA. It focuses on the pre-processing and optical character recognition (OCR) of characters in a text-based CAPTCHA, exploiting different techniques to achieve higher accuracy on the OCR task.

II. Related Work

The majority of the CAPTCHAs used in service providers is text-based. Accordingly, there is a lot of research showing the flaws of the text-based CAPTCHAs.

In [5], the authors propose two algorithms for text-based CAPTCHA recognition, based on pattern matching and hierarchical algorithms. Both algorithms try to find the shape of the objects by defining key points in their structure using the Canny Edge Detector and then comparing it to the structure of each character in a local database. One algorithm is bottom-up, tries to find words in images starting with visual cues, and incorporates lexical information later (the CAPTCHAs text is words from the dictionary). The second algorithm tries to find entire words at once. The second algorithm gave better results, with an accuracy of 92% on the EZ-Gimpy dataset and 33% accuracy on the Gimpy dataset. This study showed that, in most cases, algorithms that deal with the whole CAPTCHA at once usually achieve higher accuracy than algorithms that deal with each character separately. The authors also said that while this technique can be effective for text-based CAPTCHAs, it is no longer effective for object recognition, due to the various shapes of objects in the world, and suggested object-based CAPTCHAs for higher complexity.

In [6], the authors break the Microsoft CAPTCHA, used for systems such as MSN or Hotmail, with image segmentation and pattern matching. The image segmentation task achieves 92% accuracy, and the segmentation and recognition combined have an accuracy of 60%. Other approaches, such as [7] or [8], also use image segmentation and pattern matching to break specific CAPTCHAs dataset.

In [9], the authors propose a two-step approach to recognize text-based CAPTCHAs. The dataset used includes CAPTCHAs from the most visited websites, like MSN, Yahoo!, Google/Gmail or TicketMaster, and a language model was not used to help with the recognition. First, segmentation is used to separate the characters on the CAPTCHAs. The next step is recognizing each character, using a Convolutional Neural Network. It was shown that the recognition had high accuracy when the segmentation was done right, and most of the errors derive from a bad segmentation.

In [10], a Convolutional Neural Network (CNN) is used for CAPTCHA recognition. The network is composed by three convolution layers, three pooling layers and two fully-connected layers. The network recognizes the sequence without pre-segmentation, and with a fixed size of six characters. The problem of CNN requiring a very large training set is solved with an Active Learning mechanism. To prevent from feeding the neural network millions of CAPTCHAs, each CAPTCHA is recognized with a certain measured uncertainty. Only the most uncertain CAPTCHAs on the test set are used for retraining the model. The algorithm reaches an accuracy of almost 90%.

In [11], a novel approach is taken to break text-based CAPTCHAs. It is introduced the Recursive Cortical Network (RCN), a hierarchical probabilistic generative model with an outstanding capacity of generalization based on the human brain, designed to be trained with few examples. This model achieved an accuracy of 94.3% on character recognition on the reCAPTCHA algorithm, created and currently used by Google.

In [2], the authors present a general framework for solving text-based CAPTCHAs, with a multi-step algorithm based on reinforcement learning with joint phases of segmentation and recognition, for better results in both steps.

III. Dataset

The dataset used in this work is from Kaggle Website[1]. It contains 1070 text-based CAPTCHAs. These CAPTCHAs contain five of the following 19 characters: 2, 3, 4, 5, 6, 7, 8, b, c, d, e, f, g, m, n, p, w, x, y.

[1]https://www.kaggle.com/fournierp/captcha-version-2-images

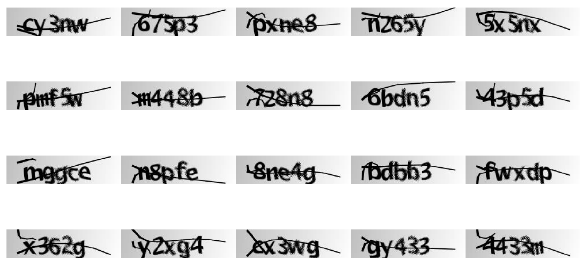

Fig. 1. 20 samples from the CAPTCHA dataset.

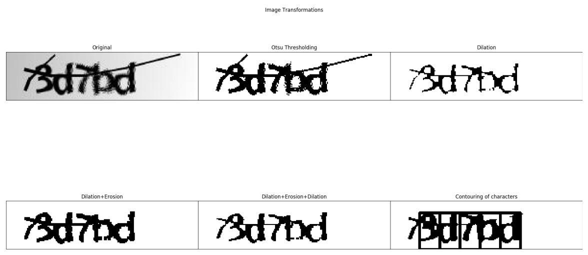

Fig. 2. Transformations applied to CAPTCHAs to remove the noise. On the top, from left to right, it can be seen the original image, the image after the Otsu thresholding and the image after one dilation. On the bottom, from left to right, it can be seen the image after the erosion, the image after the second dilation and finally the contouring of the characters.

The dataset has several important characteristics to be mentioned. Each CAPTCHA has exactly five random characters, with no relation among them (the characters do not form a word). There is noise in the CAPTCHAs, mainly in the form of black lines over the black characters. It can also be seen that the characters are approximately in the same position for every image, which eases the process of segmentation. The characters have the same font across all CAPTCHAs, but they are rotated in different angles. Twenty samples from the dataset can be seen in figure 1.

IV. Image pre-processing

To ease the classification method, the CAPTCHAs will be divided into characters, turning the problem into a classic OCR problem. Before the segmentation, a few steps are taken to remove or alleviate the background noise. First, it is applied the Otsu method to perform clustering-based image thresholding to assure that the CAPTCHA is a binary image. Then, a sequence of morphological transformations is applied to the image. A dilation and an erosion are applied sequentially, using a kernel of 3x3. Finally, another dilation is applied with a kernel of 3x1, in an attempt to eliminate the horizontal lines that create the noise in the image. After all image operations, the characters are extracted from the CAPTCHA using predetermined pixel coordinates. After human analysis, it was determined that the extracted characters had a size of 21w x 38h. The transformations applied to the CAPTCHAs are depicted in figure 2.

The characters have the frequency shown in table I and a sample of each class of characters can be seen in figure 3.

TABLE I
CHARACTER FREQUENCY IN THE DATASET.

Character	Frequency
2	270
3	271
4	289
5	288
6	267
7	262
8	272
b	247
c	276
d	268
e	245
f	277
g	281
m	282
n	541
p	259
w	244
x	271
y	240

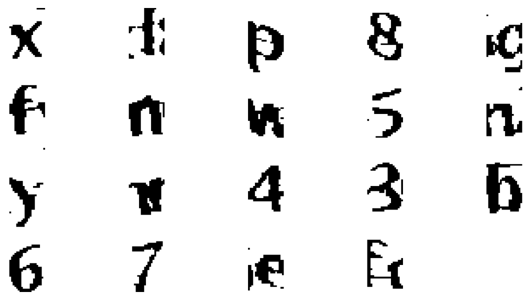
Fig. 3. One sample for each one of the 19 classes of characters extracted.

As can be seen, almost all characters have a similar frequency, with the exception of the character ’n’, that has approximately double the frequency of the other characters. For each character, the images were divided following rule: 60% of the dataset of each character was for training, 20% for cross-validation and 20% for test.

V. EXPERIMENTS AND RESULTS

A. Training with original CAPTCHAs dataset

The first approach to achieve a model to classify the CAPTCHA characters was trained with the training set of the CAPTCHA dataset and validated with the cross-validation set. Various Convolutional Neural Network models were tested with different configurations.

Initially, the Adam optimizer was used but gave poor results, as it can be seen in figure 4, with a maximum accuracy on the cross-validation set of 10.16%. The optimizer used was stochastic gradient descent with a learning rate of 0.0001. The chosen loss function was categorical cross-entropy. All layers have the ReLu (Rectified Linear Unit) as activation function, except the last layer, that has the softmax function as activation function. The network configuration is described in table II.

With the stochastic gradient descent, it achieves better results than with the Adam optimizer accordingly to figures 5 and 5. With this network, it was achieved maximum accuracy of 88.52% on the cross-validation set.

To prevent over-fitting and improve the general accuracy, they were added three dropout layers to the network, with the value of 25%. The new network configuration is described in table III and the results can be seen in figure 6. The accuracy on the cross-validation set improved to 92.00%.

In the next model, the network configuration is the same as in the table III, but the dropout value is higher (50%), as can be seen in the table IV. The results can be seen in the image 7. The maximum accuracy on the cross-validation set did not

TABLE II
FIRST NETWORK CONFIGURATION.

Layer Type	**Neurons**	**Kernel Size**
Convolutional Layer	256	3x3
Convolutional Layer	256	3x3
Pooling Layer		2x2
Convolutional Layer	512	3x3
Convolutional Layer	512	3x3
Pooling Layer		2x2
Fully-connected Layer	512	
Fully-connected Layer	512	
Fully-connected Layer	512	
Output Layer	19	

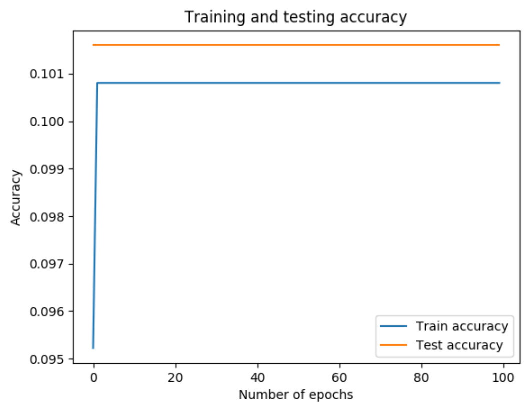

Fig. 4. Training and cross-validation accuracy with Adam as optimizer.

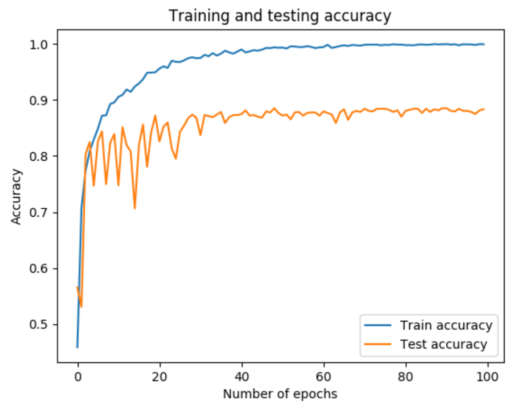

Fig. 5. Training and cross-validation accuracy with the Stochastic Gradient Descent as optimizer.

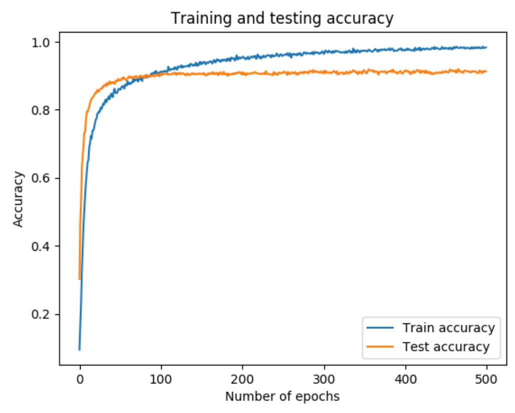

Fig. 6. Training and cross-validation accuracy with a dropout of 25%.

TABLE III
SECOND NETWORK CONFIGURATION. DROPOUT WAS ADDED TO IMPROVE REGULARIZATION.

Layer Type	Neurons	Kernel Size
Convolutional Layer	256	3x3
Convolutional Layer	256	3x3
Pooling Layer		2x2
Dropout (0.25)		
Convolutional Layer	512	3x3
Convolutional Layer	512	3x3
Pooling Layer		2x2
Dropout (0.25)		
Fully-connected Layer	512	
Fully-connected Layer	512	
Dropout (0.25)		
Fully-connected Layer	512	
Output Layer	19	

change from the previous experiment (92.00%).

Smaller and bigger networks were also tested (as it can be seen in tables V and VI), but the results were worse than with the previous network configurations (as it can be seen in the figures 8 and 9, with the maximum accuracy of 12.70% and 90.87%).

TABLE IV
THIRD NETWORK CONFIGURATION. DROPOUT VALUE WAS DOUBLED OVER THE PREVIOUS.

Layer Type	Neurons	Kernel Size
Convolutional Layer	256	3x3
Convolutional Layer	256	3x3
Pooling Layer		2x2
Dropout (0.50)		
Convolutional Layer	512	3x3
Convolutional Layer	512	3x3
Pooling Layer		2x2
Dropout (0.50)		
Fully-connected Layer	512	
Fully-connected Layer	512	
Dropout (0.5o)		
Fully-connected Layer	512	
Output Layer	19	

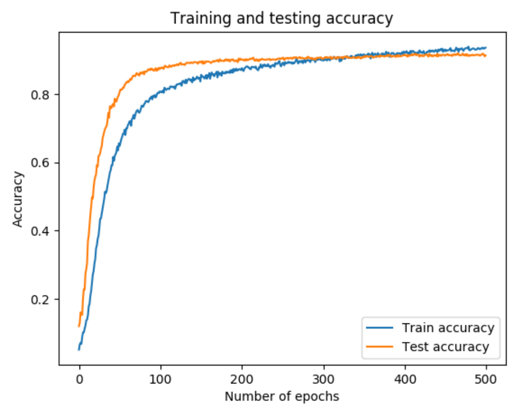

Fig. 7. Training and cross-validation accuracy with a dropout of 50%.

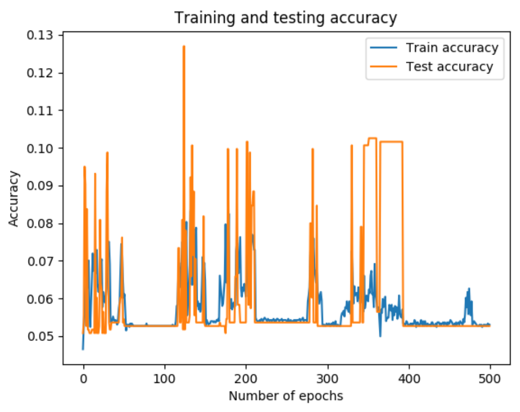

Fig. 8. Training and cross-validation accuracy for a smaller network configuration.

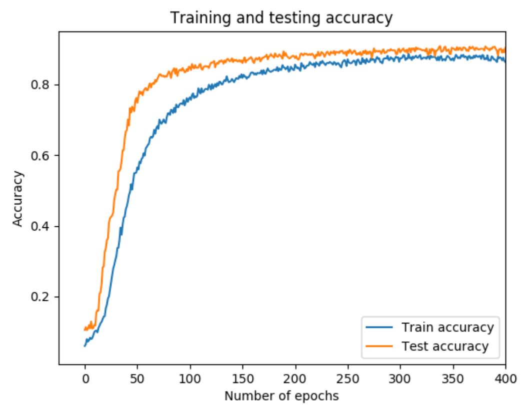

Fig. 9. Training and cross-validation accuracy for a larger network configuration.

B. *Training using EMNIST dataset*

A possible way to increase the general accuracy of the model is to increase the number of training examples. That can be done using another dataset with more training examples and use the cross-validation set of the original CAPTCHA dataset to get the cross-validation accuracy. It is needed a dataset with

TABLE V
SMALLER NETWORK CONFIGURATION.

Layer Type	Neurons	Kernel Size
Convolutional Layer	256	3x3
Pooling Layer		2x2
Dropout (0.50)		
Convolutional Layer	512	3x3
Pooling Layer		2x2
Dropout (0.50)		
Fully-connected Layer	512	
Dropout (0.50)		
Fully-connected Layer	512	
Output Layer	19	

TABLE VI
BIGGER NETWORK CONFIGURATION.

Layer Type	Neurons	Kernel Size
Convolutional Layer	256	3x3
Convolutional Layer	256	3x3
Convolutional Layer	256	3x3
Pooling Layer		2x2
Dropout (0.50)		
Convolutional Layer	512	3x3
Convolutional Layer	512	3x3
Convolutional Layer	512	3x3
Pooling Layer		2x2
Dropout (0.50)		
Fully-connected Layer	512	
Fully-connected Layer	512	
Dropout (0.50)		
Fully-connected Layer	512	
Fully-connected Layer	512	
Output Layer	19	

all the 19 characters in the CAPTCHAs dataset. The EMNIST [12] is an extension to the original MNIST dataset, with digits and letters. Only the 19 characters in the CAPTCHAs dataset are used from the EMNIST dataset. The frequency of each character in the EMNIST dataset is 2400. All characters have the same frequency.

The network configuration is described in table IV with 50% of dropout and results are shown in the image 10. The highest accuracy achieved was 39.51% for the cross-validation set. There are some possible explanations for why the cross-validation accuracy is much lower than the training accuracy (the model is underfitted). Due to the different dataset used, it can be possible that the difference between the characters in the EMNIST and in the CAPTCHAs dataset is too big for the model to be able to differentiate clearly the different characters. Besides that, the characters in the CAPTCHAs dataset all use the same font, which eases the recognition when trained with the same dataset, while the EMNIST dataset uses handwritten characters. It can be concluded that if it is desired to improve the accuracy in the CAPTCHAs dataset by increasing the training set, the best way to do that is to use the same dataset in the training and cross-validation set, but make modifications in the training set to increase the number of examples.

C. Training with generated characters

Due to the conclusions from the last subsection, the next step to improve the cross-validation accuracy is to generate

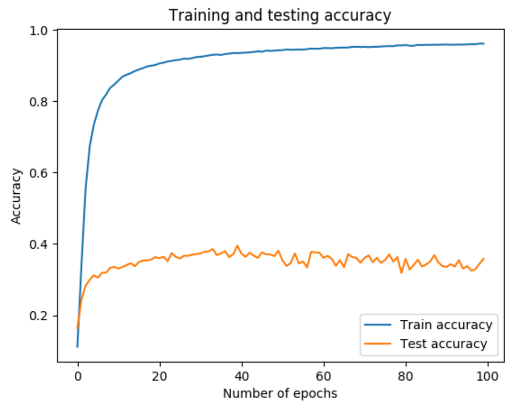

Fig. 10. Training and cross-validation accuracy for training with the EMNIST dataset and testing with the CAPTCHAs dataset.

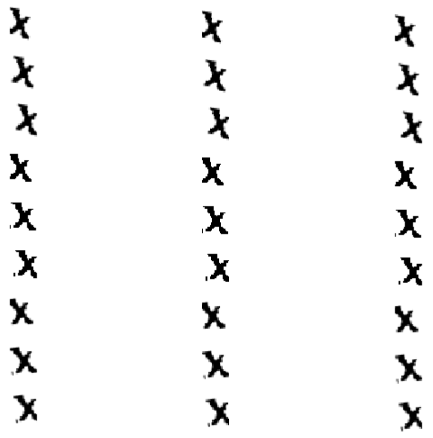

Fig. 11. 27 generated images based on an image for the character ’x’.

more training examples based on the ones already existent. For this, two operations were applied to the images in the original training set: rotation and shifting.

To each example in the training set it was applied all combinations of rotations (-10º, 0º, 10º) and vertical (-3px, 0px, 3px) and horizontal (-3px, 0px, 3px) shifts. For each original example, they generated 27 images with the combinations of the image operations. A sample of the 27 images created for a character can be seen in figure 11. The training set grew to 87048 examples among all characters (initially the training set had 3224 examples).

As it can be seen in the figure 12, the cross-validation accuracy is close to the training accuracy, reaching a maximum accuracy of 93.60%.

VI. FINAL RESULTS

To get the final results, the model with the best cross-validation accuracy was chosen. In this case, the approach chosen was to generate images, based on the training and cross-validation set, for training, as described in the last section, and test with the test set. The results can be seen in figure 13, with a maximum accuracy of 93.91%.

To obtain the accuracy of recognizing an entire CAPTCHA, a rough approximation can be made, following some assump-

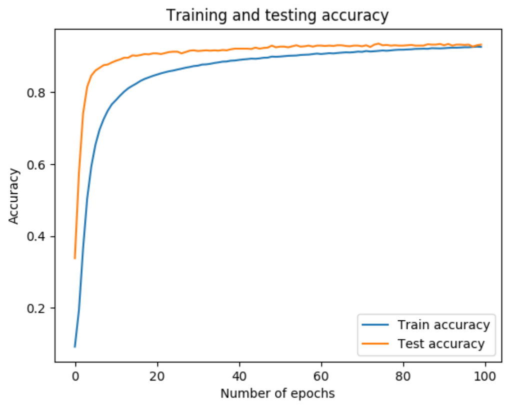

Fig. 12. Training and cross-validation accuracy with generated images derived from the original training images for training.

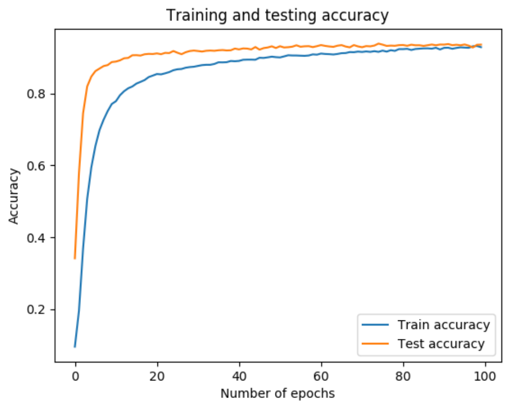

Fig. 13. Final training and test accuracy for the model with the best cross-validation results.

tions. If it is assumed that the probability of a recognizing a character is independent of its position in the CAPTCHA and independent of the other characters in the CAPTCHA, the probability of correctly recognizing the text in a CAPTCHA can be calculated by

$$p(CAPTCHA) = p(c_1) * p(c_2) * p(c_3) * p(c_4) * p(c_5)$$

being $p(c_x)$ the probability of recognizing the character in the position x.

If it is assumed that all characters have the same probability of being recognized, the probability of correctly recognizing the text in a CAPTCHA can be calculated by

$$p(CAPTCHA) = p(c)^5$$

being $p(c)$ the probability of recognizing any character.

In this case, $p(c)$ is the obtained accuracy. So, the probability of correctly recognizing the text in a CAPTCHA is 73.04%.

VII. Visualizing the Learning in the Convolutional Layers

Since 2013, a wide array of techniques have been developed for visualizing and interpreting the internals of a Convolutional Neural Network [13]. These techniques are useful to understand what is the network learning, to improve the network in future iterations.

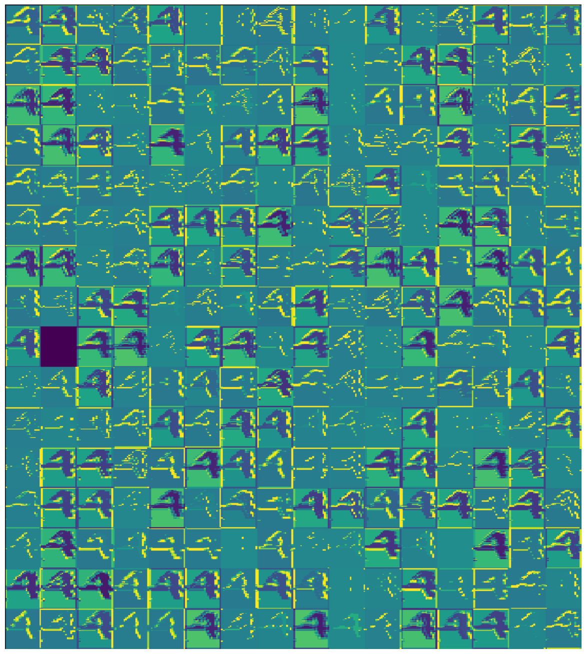

Fig. 14. Activations of the first convolutional layer of the network, given an image with the character ’4’.

Firstly, it will be shown the intermediate activations of the network, displaying the outputs of the four convolutional layers in the network given a certain input. The activation image is an image with the character ’4’. The four convolutional layers described in the architecture of the network (table IV) are depicted in the figures 14, 15, 16 and 17.

As expected, the first layer contains a collection of edge detectors, and the activations retain a big part of the information in the figure. As it goes deeper into the network, the activations become less interpretable and more abstract. Higher-level concepts are being created by the network. It also can be seen that more filters are ”blank”, which means that they are not activated by the input image. This happens because deeper layers are more specialized in specific patterns than the initial ones.

It is also useful to visualize the convolutional filters, to understand exactly what pattern each filter is detecting. To do that, it can be used the technique of *gradient ascent in input space* [14]: applying gradient descent to the input image so as to maximize the response of a filter, starting from a random image. The first 64 filter patterns of each of the four convolutional layers in the network can be seen in the figures 18, 19, 20 and 21.

As can be seen, the filters of the first layer detect vertical, horizontal and diagonal patterns, which was expected due to the activations of the first convolutional layer (figure 14). Deeper layers encode more abstract textures, some of them with resemblance to parts of characters in the dataset.

Finally, it was applied a technique called Class Activation Map (CAM) visualization, that produces heat maps of class

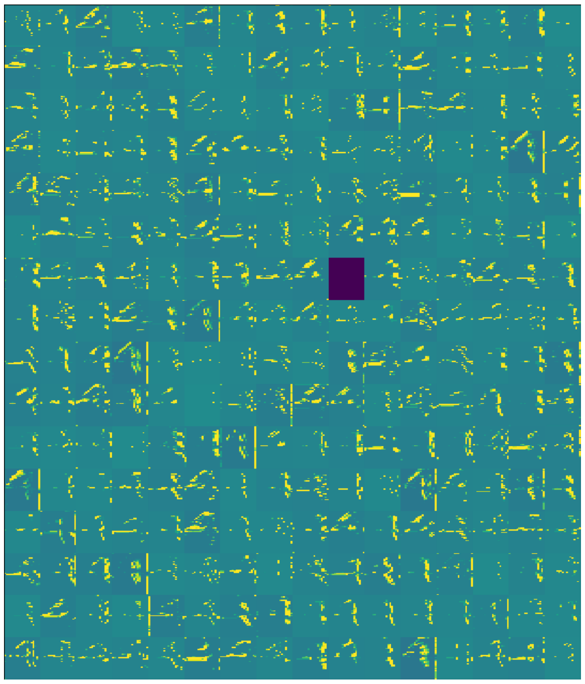
Fig. 15. Activations of the second convolutional layer of the network, given an image with the character ’4’.

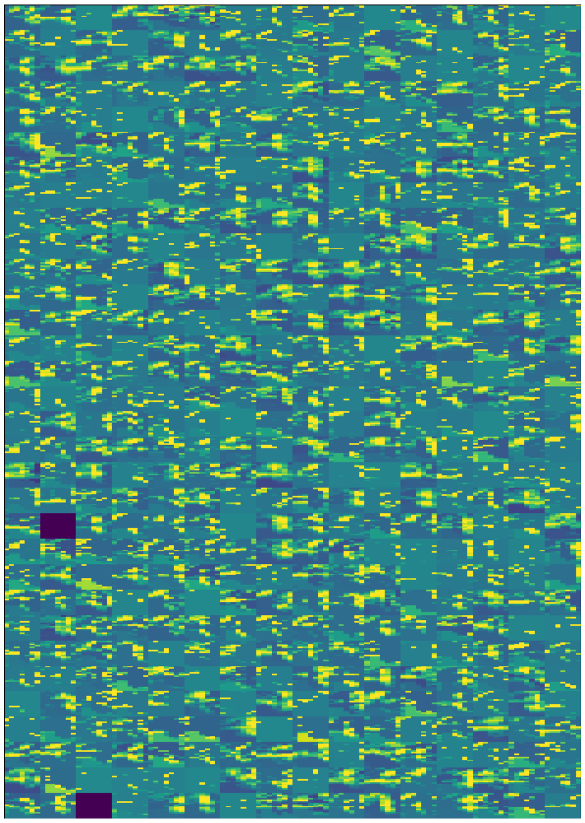
Fig. 16. Activations of the third convolutional layer of the network, given an image with the character ’4’.

activation over images. The implementation was explained in [15] and the output image represents how important each location is with respect to the class under consideration.

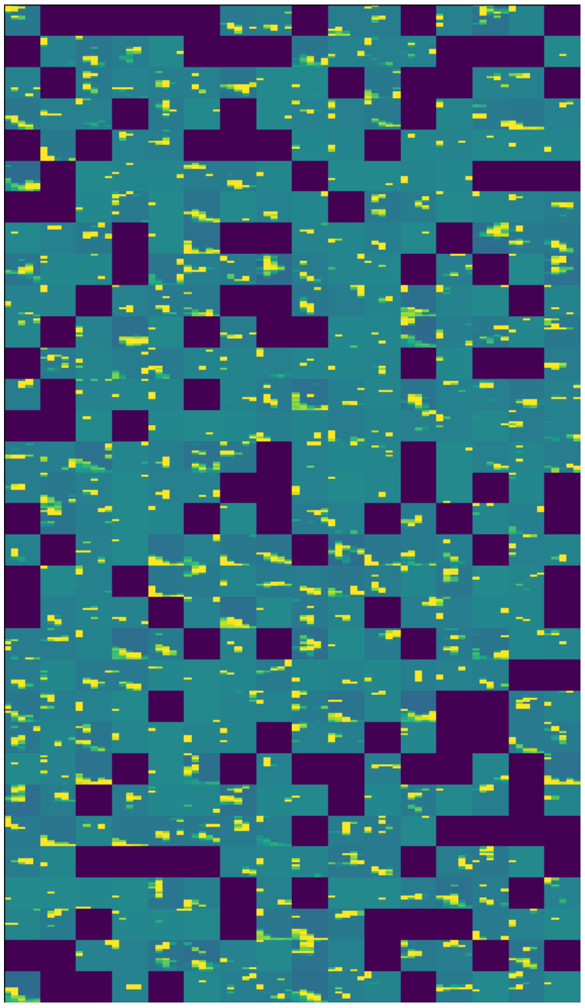
Fig. 17. Activations of the last convolutional layer of the network, given an image with the character ’4’.

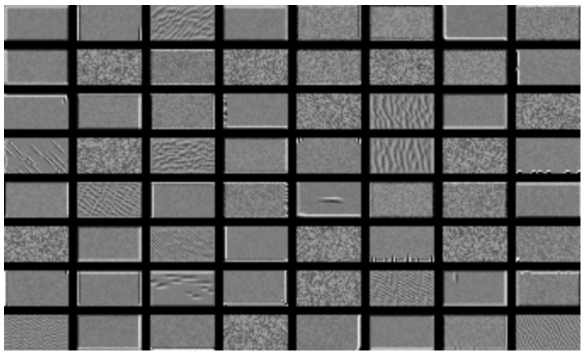
Fig. 18. Filter patterns for the first convolutional layer of the network.

According to the output of the network, it will be created a heat map that highlights the most important location that leads to the network deciding on that output. The figure 22 shows the heat map for a member of each class in the dataset.

It is interesting how, for some characters, the network seems to give more importance to the white space around the character than the character itself, as it can be seen in the heat map for the characters ’4’, ’e’, ’5’ or ’6’. However, for most

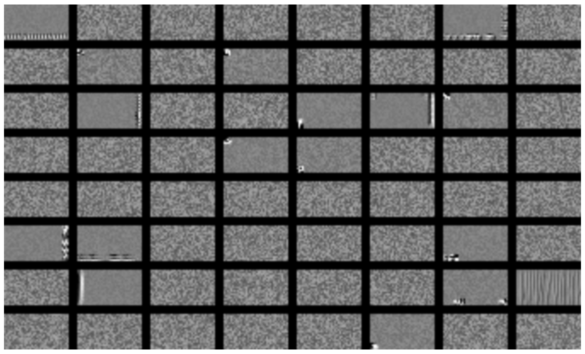

Fig. 19. Filter patterns for the second convolutional layer of the network.

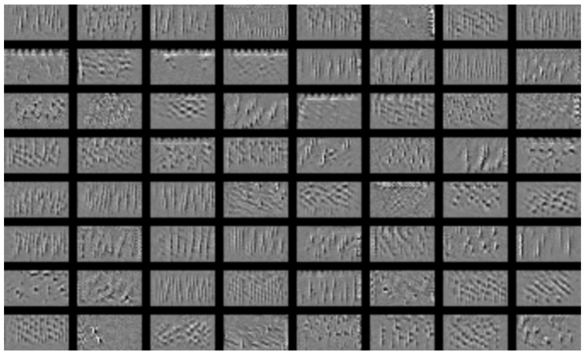

Fig. 20. Filter patterns for the third convolutional layer of the network.

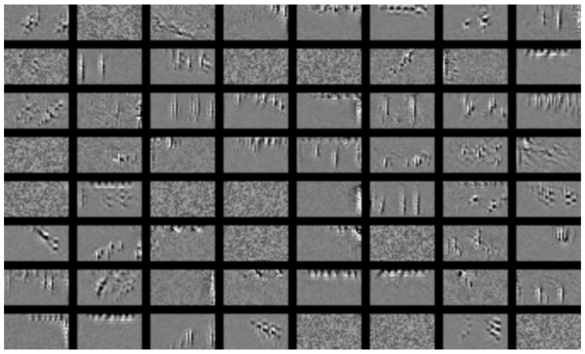

Fig. 21. Filter patterns for the last convolutional layer of the network.

characters, the heat map is understandable and the network focuses on the particular strokes of each character, as it can be seen for characters ’2’, ’3’, ’m’, ’x’ or ’y’.

VIII. Conclusion & Future Work

On this report, it was shown that the CAPTCHA recognition problem can be easily solved when the character segmentation is done effectively with no errors. It achieved a final accuracy of 93.91% of character recognition on the used dataset.

Another possible technique to improve the current accuracy of the model to character recognition is to explore the recent advances in transfer learning with a bigger dataset. Although another dataset was used for training in one of the approaches, other approaches can be made to achieve better accuracy with transfer learning, such as training just the initial layers with the bigger dataset and retraining only the last layers with the

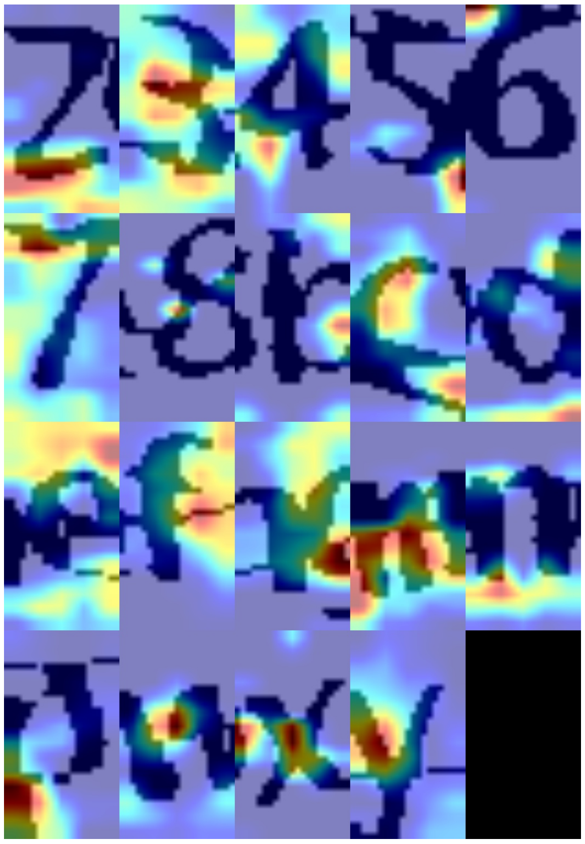

Fig. 22. Heat map detailing the most important locations of an image for the classification given by the network.

dataset from the problem. Such approach was not taken due to the specificity of the task. Because most pre-trained datasets are suitable for object recognition, with RGB colour, for bigger images, it was opted not to go for that approach, believing that data augmentation (image generation) would prove itself to achieve higher accuracy (as it did). However, it is an approach that is worth exploring in the future.

It was also shown that the generation of new images to include in the training set with slight modifications, such as rotations or translations, improves the accuracy of the model, especially when the training set is small.

References

[1] Wikipedia contributors, “Captcha — Wikipedia, the free encyclopedia,” 2018, [Online; accessed 27-December-2018]. [Online]. Available: https://en.wikipedia.org/wiki/CAPTCHA#Characteristics

[2] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, “The end is nigh: Generic solving of text-based captchas,” in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*. San Diego, CA: USENIX Association, 2014. [Online]. Available: https://www.usenix.org/conference/woot14/workshop-program/presentation/bursztein

[3] S. Sivakorn, I. Polakis, and A. D. Keromytis, “I am robot: (deep) learning to break semantic image CAPTCHAs,” in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, mar 2016. [Online]. Available: https://doi.org/10.1109/eurosp.2016.37

[4] B. Zhao, H. Weng, S. Ji, J. Chen, T. Wang, Q. He, and R. Beyah, “Towards evaluating the security of real-world deployed image captchas,” in *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*, ser. AISec ’18. New York, NY, USA: ACM, 2018, pp. 85–96. [Online]. Available: http://doi.acm.org/10.1145/3270101.3270104

[5] G. Mori and J. Malik, “Recognizing objects in adversarial clutter: breaking a visual CAPTCHA,” in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* IEEE Comput. Soc. [Online]. Available: https://doi.org/10.1109/cvpr.2003.1211347

[6] J. Yan and A. S. El Ahmad, “A low-cost attack on a microsoft captcha,” in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ser. CCS ’08. New York, NY, USA: ACM, 2008, pp. 543–554. [Online]. Available: http://doi.acm.org/10.1145/1455770.1455839

[7] J. Yan and A. S. E. Ahmad, “Breaking visual CAPTCHAs with naive pattern recognition algorithms,” in *Twenty-Third Annual Computer Security Applications Conference (ACSAC 2007).* IEEE, dec 2007. [Online]. Available: https://doi.org/10.1109/acsac.2007.47

[8] S. Li, S. A. H. Shah, M. A. U. Khan, S. A. Khayam, A.-R. Sadeghi, and R. Schmitz, “Breaking e-banking captchas,” in *Proceedings of the 26th Annual Computer Security Applications Conference*, ser. ACSAC ’10. New York, NY, USA: ACM, 2010, pp. 171–180. [Online]. Available: http://doi.acm.org/10.1145/1920261.1920288

[9] K. Chellapilla and P. Y. Simard, “Using machine learning to break visual human interaction proofs (hips),” in *Advances in Neural Information Processing Systems 17*, L. K. Saul, Y. Weiss, and L. Bottou, Eds. MIT Press, 2005, pp. 265–272. [Online]. Available: http://papers.nips.cc/paper/2571-using-machine-learning-to-break-visual-human-interaction-proofs-hips.pdf

[10] F. Stark, R. Triebel, and D. Cremers, “Captcha recognition with active deep learning.”

[11] D. George, W. Lehrach, K. Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, 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, no. 6368, p. eaag2612, oct 2017. [Online]. Available: https://doi.org/10.1126/science.aag2612

[12] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, “EMNIST: an extension of MNIST to handwritten letters,” *CoRR*, vol. abs/1702.05373, 2017. [Online]. Available: http://arxiv.org/abs/1702.05373

[13] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 818–833.

[14] J. Yosinski, J. Clune, A. M. Nguyen, T. J. Fuchs, and H. Lipson, “Understanding neural networks through deep visualization,” *CoRR*, vol. abs/1506.06579, 2015.

[15] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” in *2017 IEEE International Conference on Computer Vision (ICCV).* IEEE, oct 2017. [Online]. Available: https://doi.org/10.1109/iccv.2017.74