An investigation into CAPTCHA improvement

Asu Tutku Gökçek, Barış Söke, Ertan Can Güner, Vincent Garver Stewart Pedersen

Summary

The study of privacy and security and how it can be applied has become essential with the need of protecting personal data in the tech world. One way that websites provide security is through CAPTCHAs which is the short version of “Completely Automated Public Turing test to tell Computers and Humans Apart”. CAPTCHA is a security feature which is used by websites to determine whether a user is a real human being or an automated bot. This project's approach is to test CAPTCHA effectiveness by implementing a CAPTCHA solver which can read the format of CAPTCHA which consists of letters and numbers in texts. This CAPTCHA solver is implemented by firstly creating the CAPTCHA images with the background and letters/numbers combinations and then reading this dataset of CAPTCHAs created and solving them. The solver is tested against a variety of CAPTCHAs to determine its efficiency. Afterwards this project implements a novel type of CAPTCHA, a directional CAPTCHA and tests it against a solver specifically designed for directional CAPTCHAs to get an estimate of its efficiency. Then the efficiencies of the two CAPTCHAs are compared to each other to get an insight into the successfulness of each CAPTCHA type. Various open source libraries of the Python programming language which have the necessary tools to achieve implementation, training, testing and evaluating the results have been used. After performing the tests, it has been observed that the directional CAPTCHA with the added noise turned out to be more successful in the Turing testing than the text based CAPTCHA. The results are explained in the upcoming sections.

Progress

CAPTCHA Solver

Generating CAPTCHAs

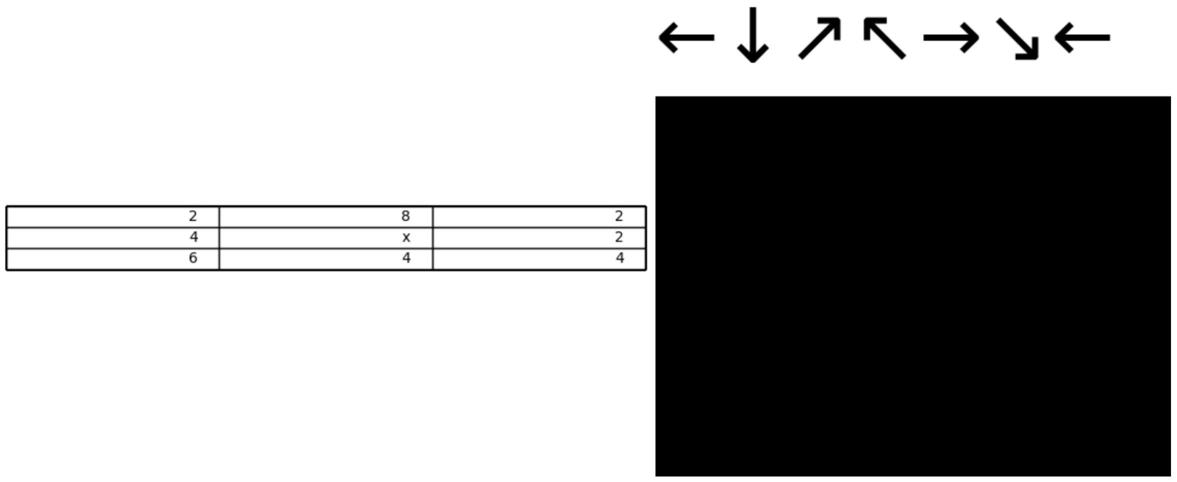
Captcha_gen.py is responsible for generating CAPTCHA images. Total of 900 images are generated with a random color for the background and 4 main colors (orange, green, blue, indigo) for the background. First, a random background color and a random string consisting of letters and numbers with a length of 8 is generated for the CAPTCHA. Then, the image is drawn using PIL (Python Imaging Library) library and random dots are added as noise to the image. To test the robustness of the CAPTCHA solver, another set of CAPTCHA images are generated with random colored backgrounds and random colored letters, with a length of 8.

Decoding CAPTCHAs

Captcha_decoder.py is responsible for solving the CAPTCHAs that were generated. First, all of the images are read and stored in the list “res”. Then two methods are used to recognize the text in the images; first one is using PIL to convert the image to grayscale and the second one is using OpenCV library to smooth the image by applying gaussian blur on to it. Finally, the Tesseract Engine is used to recognize characters in the images.

Directional CAPTCHA

A more novel approach to CAPTCHAs are the directional CAPTCHAs first proposed in Ranjan and Kumar’s paper (Ranjan & Kumar, 2015) which work by having a matrix of numbers with an x in the middle, displayed next to a row of arrows all pointing in different directions.



← ↓ ↗ ↖ → ↘ ←

2	8	2
4	x	2
6	4	4

Figure 1: Directional CAPTCHA

The idea is then that it is solved by reading the arrows left to right and inserting the number in the cell that the arrow points to if the arrows were to be coming out from the center (‘x’). Thus the answer to the above CAPTCHA is “4422244.” After a quick introduction, this type of CAPTCHA is pretty intuitive for humans to solve but should in theory be more complicated for computers to solve as they revolve around the idea of recognizing objects and where they are pointing and afterwards applying that to a matrix.

Generating Directional CAPTCHAs

DirectionalCaptcha_gen.py is responsible for generating directional CAPTCHAs. First, a random matrix of numbers and a random sequence (in 8 directions) of arrows is generated. Afterwards, the generated arrows are read and traversed in the matrix to find the solution to the CAPTCHA.

Additionally, some CAPTCHAs are created with noise added to the arrow image. This noise comes from a normal distribution. All the images are then saved with the name as the solution. They are divided into two different folders, directionalCAPTCHA which includes the type of directional CAPTCHA shown above which doesn't have any added noise, directionalCAPTCHANoise, which includes directional CAPTCHAs but with noise added to the arrow image.

Solving Directional CAPTCHAs

DirectionalCaptchaDecoder.py is responsible for decoding directional CAPTCHAs. First, the generated directional CAPTCHA image is processed to convert the image to a more workable representation. It is converted into a grayscale version then it applies gaussian blur to remove any noise. Finally, it detects the edges of the image and applies filling and removes any unwanted overflows on the edges. Then to determine the arrow directions, the algorithm finds the contours of the given image. Contours then used to approximate the shape of the arrow and convex hull are drawn to get a more simple shape. After finding the convex hull, top and bottom of the arrow needs to be determined. The top and bottom of the arrow indicate the direction of the arrow by checking the position of top and bottom points relative to each other. Finally, the matrix part of the picture is read by utilizing the API from extracttable.com, which works by giving it an image and returning the table/matrix on said image. Initially the plan was to use tesseract for this part of the solution, but it was quickly apparent that tesseract struggled heavily with reading the tables despite various attempts at changing the configurations. Afterwards, the directions of the arrows are applied to the extracted table, just as in the generating file, to determine what the solution would be.

Github link to the project: https://github.com/bsoke/CAPTCHA

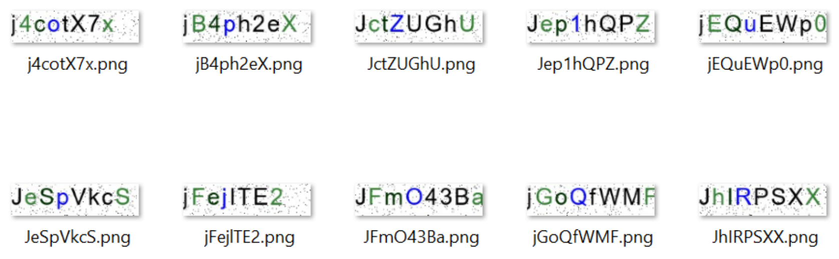
Experimental Evaluation

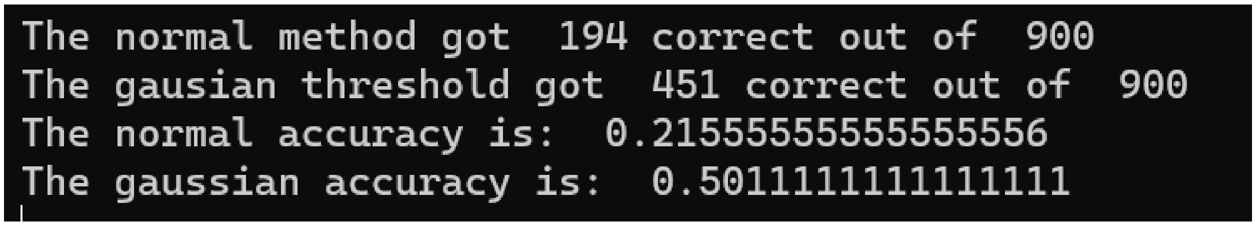
Text-Based CAPTCHAs

In order to test the overall accuracy of the CAPTCHA decoders, a test with two different datasets on basic text-based CAPTCHA images has been conducted. Since there are two slightly different sets of CAPTCHA images present, to test the relative accuracy of the solver two different tests are conducted on text-based CAPTCHAs.

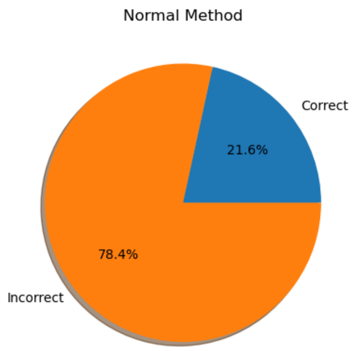
Text-Based CAPTCHAs with White Background

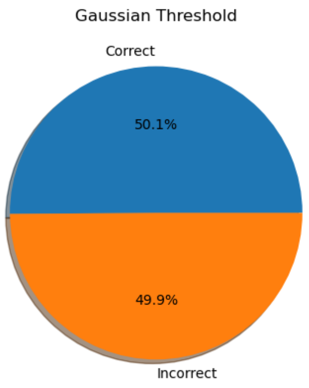
The objective of this test is to evaluate the accuracy of the text-based CAPTCHA solver on white background and colored letters CAPTCHA images. Overall the first set of 900 CAPTCHA images were used for testing. The CAPTCHA solver was run on each image using two different threshold and accuracy methods which are normal thresholding and Gaussian thresholding. In this test, the output of the CAPTCHA solver was compared with the actual solution of the CAPTCHA, which is the exact name of the *.png.* The results show that the application of Gaussian thresholding improved the accuracy of the CAPTCHA solver by 28.55% compared to the normal thresholding. This reveals the fact that the images used for testing contained a significant amount of noise, which was directly affecting the accuracy of the CAPTCHA solver. Gaussian thresholding effectively reduced the noise in the images and allowed the CAPTCHA solver to perform better. In practice, it is recommended to use Gaussian thresholding on noisy CAPTCHA images to increase the accuracy. One other fact that affected the results is the effectiveness of the CAPTCHA solver is the OCR that was used in the tests. As could be seen on the figures below, the noise has affected the results by getting recognised as dots(.) or dash(-) characters, along with the letters “S” being recognized as “5” and vice-versa.

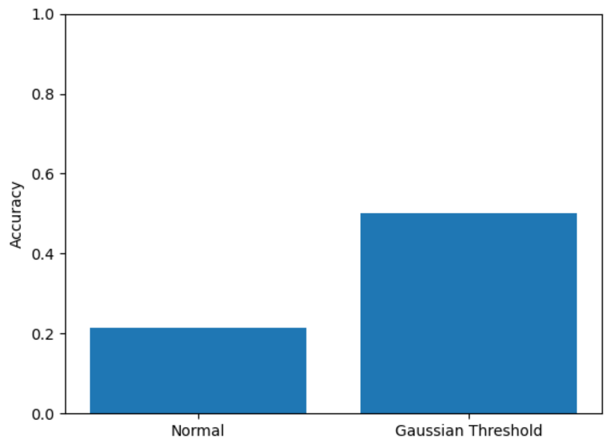


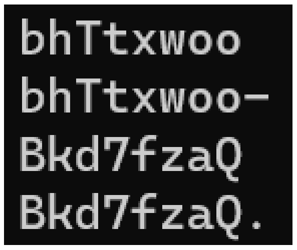


```
The normal method got  194 correct out of  900
The gausian threshold got  451 correct out of  900
The normal accuracy is:  0.21555555555555556
The gaussian accuracy is:  0.5011111111111111
```

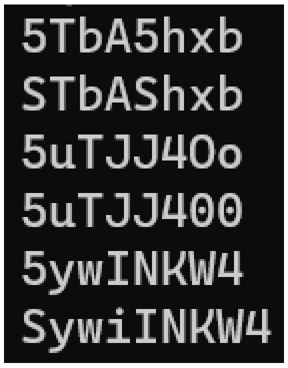








```
bhTtxwoo
bhTtxwoo-
Bkd7fzaQ
Bkd7fzaQ.
```

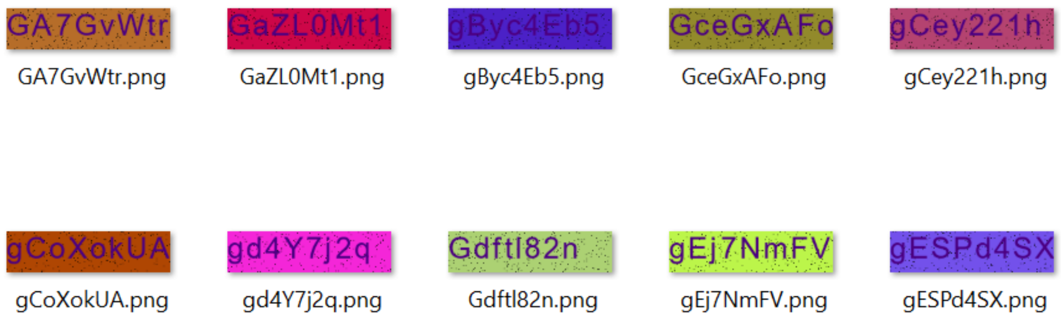


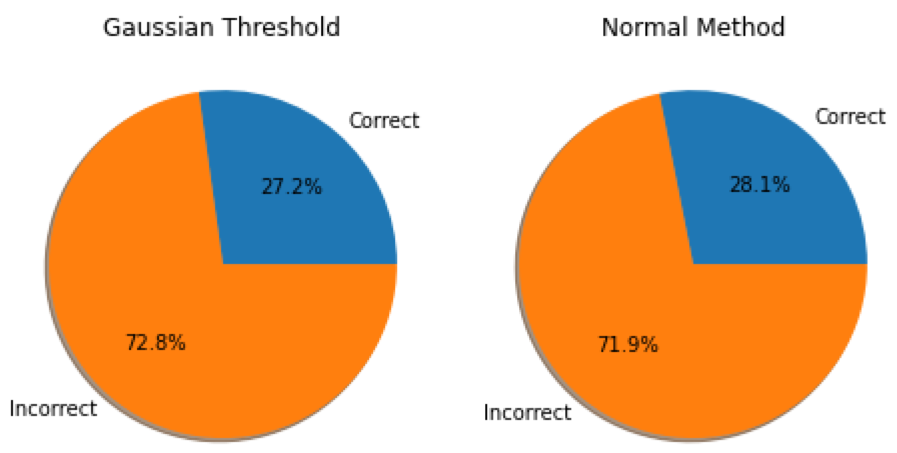
```
5TbA5hxb
STbAShxb
5uTJJ4Oo
5uTJJ400
5ywINKW4
SywiINKW4
```

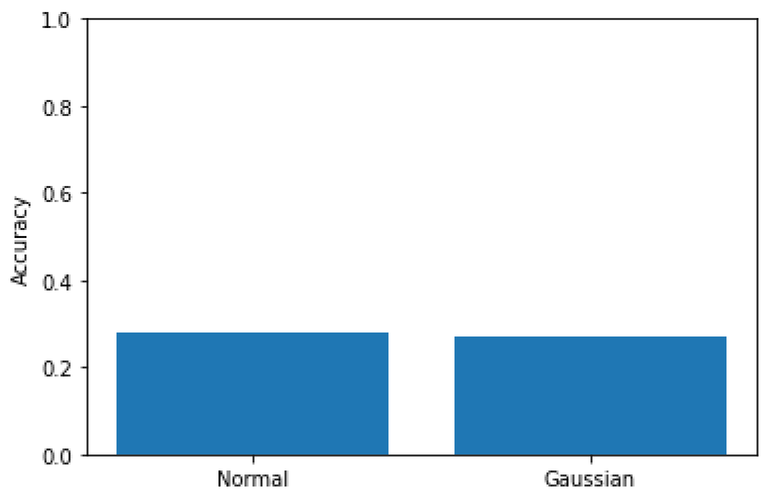
Text-Based CAPTCHAs with Colored Background and Colored Letters

The objective of this test is to evaluate the accuracy of a text-based CAPTCHA solver on a set of CAPTCHA that has colored background and colored letters with added noise. 1010 CAPTCHA images were used for testing. The captcha solver was run on each image using the two different thresholding methods which are normal and Gaussian. The results show that the application of

Gaussian thresholding did not have a significant effect on the accuracy of the CAPTCHA solver compared to the normal thresholding. Both methods performed similarly with an accuracy around 28%. This suggests that the images used for testing did not contain a significant amount of noise or the noise present in the image is not affecting the performance of the CAPTCHA solver. Since the set which was used to conduct this test does not have any significantly different noise from the set that was used for the previous test. It could be said that the colors are affecting the robustness of the text-based CAPTCHA solver dramatically, even if grayscaling and noise reduction is applied. As mentioned on the white-background & colored letters test, the overall accuracy is also affected by the character recognition engine that was used in the tests.





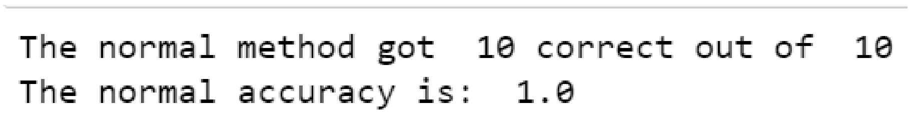


Directional CAPTCHAs

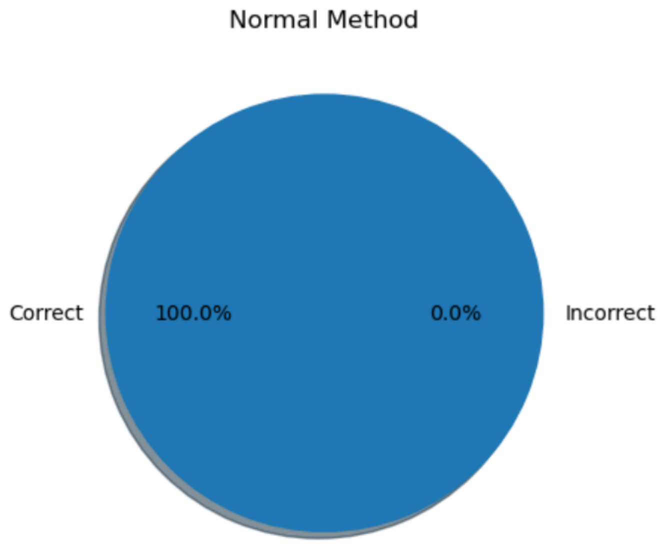
Directional CAPTCHAs work by generating a 3x3 matrix of eight numbers with the middle being empty and eighth arrows pointing at random directions (cardinal and ordinal directions). The arrows indicate the order of numbers that has to be pressed.

Directional CAPTCHAs with White Background:

The objective of this test is to evaluate the accuracy of a directional CAPTCHA solver on a set of directional CAPTCHAs that has a white background on the directional arrows. 10 directional CAPTCHA images were used for testing. The CAPTCHA solver was run on each image and the clean images gave %100 accuracy results.

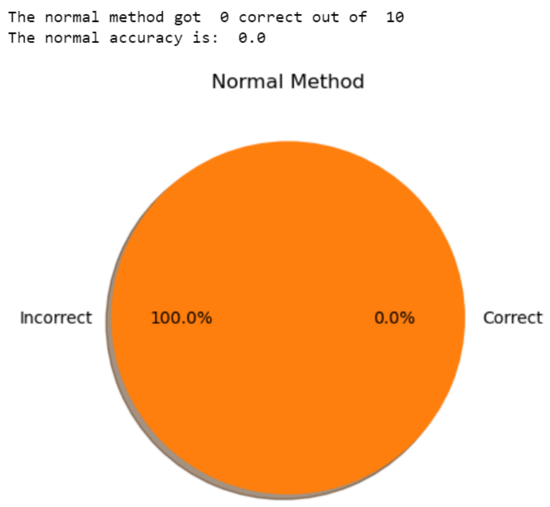


```
The normal method got  10 correct out of  10
The normal accuracy is:  1.0
```



Directional CAPTCHAs with Added Noise:

The objective of this test is to evaluate the accuracy of a directional CAPTCHA solver on a set of directional CAPTCHAs that has a white background but with an addition of random poisson noise. 10 directional CAPTCHA images were used for testing. The CAPTCHA solver was run on each noisy image and noise addition made the solver's accuracy drop to %0.



```
The normal method got  0 correct out of  10
The normal accuracy is:  0.0
```

Analysis

Through experiments on the two types of CAPTCHAs, we found that the solver on text-based CAPTCHAs managed to get between 30%-50% correct depending on the type of text-based CAPTCHA. The experiment on the directional CAPTCHAs was a bit more interesting as it showed that the solver doesn't fail to solve the directional CAPTCHAs made with no noise, as it easily reads both the arrow image and the matrix image. However with added noise the solver struggles tremendously and can't solve any CAPTCHAs. The bad results in this case is due to the fact that the algorithm might find one or two arrows, but without knowing them all and the order of them, it can't even try to solve it. Despite the solver not being able to solve any directional CAPTCHAs with noise, they are still very readable to humans, as the noise to human eyes is treated as part of the background. This makes for an interesting comparison between the two types of CAPTCHAs, as even though the added noise also decreases the solvability of traditional text-based CAPTCHAs, it still doesn't have zero percent solvability like the directional CAPTCHA with noise. This would in theory be possible to achieve, however you lose a lot of the human readability of the CAPTCHAs, which is one of the most integral parts of the CAPTCHA features.

Conclusion

The results of the tests which have been conducted show that the accuracy of the text based CAPTCHA solver (with white background) has been improved more by Gaussian thresholding compared to normal thresholding because Gaussian thresholding reduced the noise therefore increased the performance. The same thresholds that were applied to the text based CAPTCHA solver (with colored background) and the results of normal thresholding and Gaussian thresholding were close values. This shows that the noise was not affecting the performance of the CAPTCHA solver. It is concluded that colored backgrounds improve the strength of the CAPTCHA. The tests on directional CAPTCHA show that the addition of the noise dropped the accuracy to zero.

Directional CAPTCHAs with added noise decreases the solvability of the CAPTCHAs without damaging the human readability. Thus they can be considered a better alternative to traditional text-based CAPTCHAs.

However, although these CAPTCHA types provide security until some point, they can be easily solved by AI. Therefore, even though coming up with a new type of text based CAPTCHA such as directional CAPTCHA will not be enough to provide full privacy and security because an adversary can easily train an ML model or implement an algorithm to bypass this CAPTCHA with a targeted attack. Image based CAPTCHAs can be given as a better example for providing the security which the CAPTCHAs proposed failed to provide in this project. Even with today's access to complex machine learning models and computational power, it's hard to get accurate results with a variety of labeled images. This is why today a lot of companies don’t use text based or directional CAPTCHAs, instead they use image based CAPTCHAs.

References

1) Ranjan, A. K., & Kumar, B. (2015). Directional captcha: A novel approach to text based CAPTCHA.
2) How to detect different types of arrows in image? (1968, March 01). Retrieved from https://stackoverflow.com/questions/66718462/how-to-detect-different-types-of-arrows-in-image
3) Pillow. (n.d.). Retrieved from https://pillow.readthedocs.io/en/stable/
4) API to convert image to Excel, extract tables from PDF Online. (n.d.). Retrieved from https://www.extracttable.com/