Breaking noCAPTCHA reCAPTCHA

Aaryen Mehta 190495
Mohd Muzzammil 190503
Varenya Srivastava 190943

January 2023

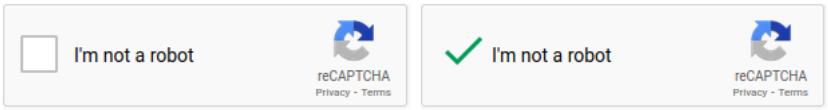
CAPTCHA stands for Completely Automated Public Turing test to tell Computers and Humans Apart. It is a type of challenge-response test used to determine whether or not the user is human. CAPTCHAs are often used to prevent automated software (bots) from engaging in abusive activities on a website, such as registering multiple accounts, posting spam, or attempting to gain unauthorized access. Common types of CAPTCHAs include image recognition, where the user is asked to identify objects in a distorted image; audio recognition, where the user is asked to transcribe spoken words; and logical puzzles, where the user is asked to solve a simple math problem or follow a set of instructions. The effectiveness of CAPTCHAs can be reduced by advances in artificial intelligence, but new forms are continuously developed to stay ahead of bots.

1 What is noCAPTCHA?

noCAPTCHA [1] is a type of CAPTCHA designed to be more user-friendly than traditional CAPTCHAs. Instead of asking users to solve a puzzle or enter a set of characters, noCAPTCHA asks users to check a box or click a button to confirm that they are human. The system then uses advanced algorithms to determine whether the user is a human or a bot based on their behavior. noCAPTCHA is considered to be less intrusive and more accessible for users with disabilities.

1.1 Widget

When the user visits a webpage protected by reCAPTCHA, the following widget is displayed.



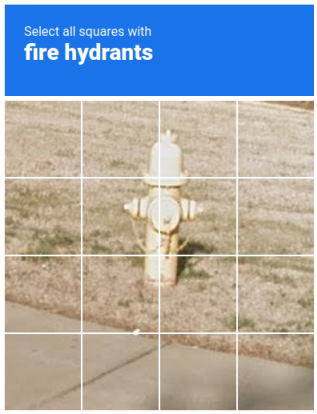
The JavaScript code used for this widget is jumbled to prevent third-party analysis. This widget performs multiple hidden checks before the user clicks the checkbox. After which, all the information is sent back to the server, and a verification code is returned.

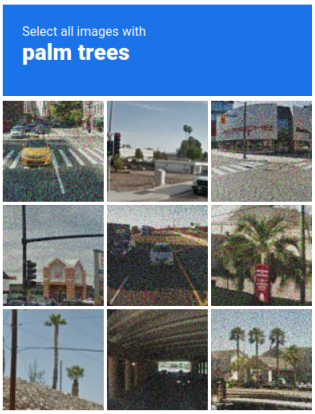
1.2 Process

Once the checkbox is clicked, a request is sent to Google servers which contains the information about cookies from the user's browser and the information regarding the user's various browser checks. This is then analyzed by an advanced risk analysis system, which decides which type of CAPTCHA challenge will be given to the user.

1.3 Challenges

- **reCAPTCHA v3** This is a hidden challenge integrated into the webpage. This tracks user behavior and cookies and returns a probability of the user being human. If it is greater than 0.5, the user is verified on form submission without any pop-up or dedicated widget.
- **noCAPTCHA** This challenge involves clicking on a checkbox as discussed before
- **reCAPTCHA v2** In this challenge, the user has to click the images with regards to the question asked, and then the user is distinguished from the bot. This challenge has two subtypes
  - One of the challenge has 16 images, and the user is supposed to check all the boxes containing the particular object
  - The other one has 9 images, and the user is supposed to check all the boxes having the given object in it





2 Solving noCAPTCHA

Once the challenge is presented to the user, it has to be solved within 55 seconds. Otherwise, the user is required to click on the checkbox again to receive a new challenge. The approach implemented by us is broken down into two major steps.

2.1 noCAPTCHA Checkbox Clicker

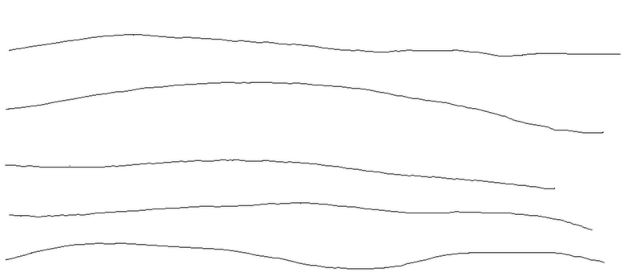
As discussed, the noCAPTCHA is hidden and has a lot of jargon code, which makes it extremely difficult to analyze the JavaScript code and find out where the checkbox is. Hence, similar to a human, computer vision techniques have been used to find out the position of the checkbox on the webpage.

2.1.1 Locating Checkbox

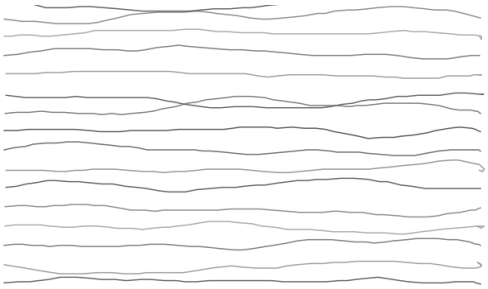
This method requires that the webpage containing the noCAPTCHA is open. Once the code for the clicker is run, it takes a screenshot of the open website and reads the image using the OpenCV library in Python [2]. Standard grayscaling and masking techniques are used to figure out the location of the checkbox. Bounding rectangle coordinates are acquired; thus, the center of the noCAPTCHA checkbox is obtained strictly using computer vision.

2.1.2 Simulating Human-like Mouse Movement

An algorithm called WindMouse [3] is used to simulate these movements. This algorithm is a physics-based movement of a particle in the wind. The cursor is modeled as having some mass and is acted by gravitational and wind forces. Gravity makes up for a constantly acting force, whereas wind makes up for a variable and randomly acting force. The wind force reduces the closer the cursor moves to the target. So the cursor converges to the final destination.



Actual Mouse Movement



WindMouse Algorithm

Here, the gravitational force is the intention of the user to reach the checkbox. Hence it is constantly acting in a constant direction. Whereas wind is the micromovements, our hand does while moving to the target. Due to this, the model is able to deliver human-like movements.

2.1.3 Clicking the Checkbox

WindMouse algorithm returns the list of coordinates on the screen through which the cursor should move, essentially giving the path that should be traversed to simulate human-like movement and reach the target. These points are visited by using Python [4]. Within a realistic timeframe that imitates humans. Finally, when the noCAPTCHA checkbox's center is reached, the mouse is clicked automatically to complete the challenge.

If an existing session of a real user is employed for this challenge, it is an immediate success, according to our testing. But if the incognito mode is used or cookies are cleared before solving this challenge, then almost always reCAPTCHA v2 challenge is popped up. The next step details the solution to this challenge.

2.2 reCAPTCHA Breaker

Ideally, this would require a huge dataset to train the model for multiple classes. And an even bigger one if we wish to solve both cases that can pop up for higher accuracy. Some workarounds to avoid this was thought of as follows:

1. We can train the model for some subclasses and apply classification for this. If case (a) in reCAPTCHA pops up, then we simply skip the captcha until we get a classification problem with nine images.

2. Extending upon the previous approach, we can train the model to identify only one specific object, say cars. And obtain a huge dataset to train the model to identify cars and cars only. Skip the rest of the challenges. This would also increase the accuracy of solving correctly.

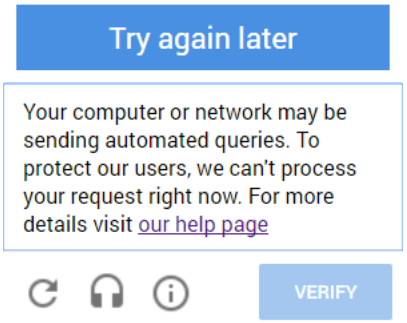
3. Instead of solving the image reCAPTCHA, we can opt for the audio option and use Google Assistant to identify the phrase spoken in the audio captcha and use that as a possible solution for the reCAPTCHA

The first approach was applied to this project. The dataset is a narrowed-down version of the Oxford dataset for "Very Deep Convolutional Networks for Large-Scale Image Recognition" [6], containing five classes and around 1000 images for each class. The model was trained using the torch library in Python. Transfer learning, along with a pre-trained VGG16 model, was able to perform with 80% accuracy for the classification of a single image into the five classes. The same dataset was used for training, testing, and validation by partitioning. [7]

3 Insights

Here are some insights regarding the mathematical feasibility of the approach, along with various other approaches and their drawbacks found in manual testing.

- 80% accuracy is low if we use it to classify nine images. This would come out to be 13%, which is extremely low in practice.
- For approach 2, the most distinguishable object that appears with decent frequency was found to be fire hydrants. But no good free dataset for fire hydrants could be obtained for training purposes.
- Another drawback in the previous approach is too much refreshing of reCAPTCHA. Google is able to identify this behavior and flags it as a bot. This was observed during manual testing.
- In manual testing, the audio option was not efficient. The audio is distorted smartly such that Google Assistant is not able to identify the phrase at all. Hence deeming it infeasible.
- At least 97% accuracy is required if we wish to identify all nine images correctly with 80% success probability. But the paper by Oxford University[6] can only get an accuracy of 92.7% even after training on 14 million images.



4 Conclusion

If we are using an existing session of a real user, then noCAPTCHA can be automatically solved almost always with success. But if we want to deploy the bot on a fresh Google account or in an incognito tab, then we are required to solve reCAPTCHA, which for the single image has an accuracy of 80%. Google reverse image search with some large enough labeled dataset is required for getting even a decent enough accuracy for the solution of reCAPTCHA. In addition to this, it required high computation power making it impossible for this project.

References

[1] About Google noCAPTCHA: https://www.google.com/recaptcha/about/

[2] OpenCV library in Python: https://pypi.org/project/opencv-python/

[3] Details of the mathematics used in the WindMouse algorithm: https://ben.land/post/2021/04/25/windmouse-human-mouse-movement/

[4] Simulating mouse and keyboard movements using Python: https://pypi.org/project/PyAutoGUI/

[5] Literature on a similar problem statement: Sivakorn, Suphannee. "I'm Not a Human: Breaking the Google reCAPTCHA." (2016).

[6] Oxford University paper: https://doi.org/10.48550/arXiv.1409.1556

[7] Transfer learning for image classification: https://github.com/RyanSh3n/reCAPTCHASolver

[8] Our Git repository containing codes for both the mouse clicker and reCAPTCHA breaker can be found here: https://github.com/AaryenMehta/CGS402