Streamlining Attacks on CAPTCHAs with a Computer Game

Jeff Yan, **Su-Yang Yu**
School of Computing Science
Newcastle University, England
{jeff.yan, s.y.yu}@ncl.ac.uk

Abstract

CAPTCHA has been widely deployed by commercial web sites as a security technology for purposes such as anti-spam. A common approach to evaluating the robustness of CAPTCHA is the use of machine learning techniques. Critical to this approach is the acquisition of an adequate set of labeled samples, on which the learning techniques are trained. However, such a sample labeling task is difficult for computers, since the strength of CAPTCHAs stems exactly from the difficulty computers have in recognizing either distorted texts or image contents. Therefore, until now, researchers have to manually label their samples, which is tedious and expensive. In this paper, we present *Magic Bullet*, a computer game that for the first time turns such sample labeling into a fun experience, and that achieves a labeling accuracy of as high as 98% for free. The game leverages human computation to address a task that cannot be easily automated, and it effectively streamlines the evaluation of CAPTCHAs. The game can also be used for other constructive purposes such as 1) developing better machine learning algorithms for handwriting recognition, and 2) training people's typing skills.

1 Introduction

A CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a program that generates and grades tests that are solvable by humans, but intended to be beyond the capabilities of current computer programs [von Ahn et al., 2004]. This technology is now almost a standard security mechanism for defending against undesirable or malicious Internet bot programs, such as those spreading junk emails and those grabbing thousands of free email accounts instantly. It has found widespread application on numerous commercial web sites. The most widely used CAPTCHAs are *text-based* schemes, which typically require users to solve a text recognition task. Google, Microsoft and Yahoo have all deployed their own text CAPTCHAs for years to defend against email spam.

Spammers could achieve substantial financial gain by using computer programs that can automatically bypass a heavily used CAPTCHA such as those deployed by Google, Microsoft and Yahoo. Therefore, it is both intellectually interesting and practically relevant for researchers to understand and improve the robustness of CAPTCHAs, in order to try to stay (at least) a step ahead of spammers.

The typical method for evaluating whether or not a CAPTCHA scheme is robust is to try to attack or break the scheme, in the sense of writing a computer program that can automatically answer the tests generated by the scheme. A well-established method for attacking text CAPTCHAs [Chellapilla and Simard, 2004] is based on machine learning and typically includes the following steps.

Segmentation: locating where the characters are in each CAPTCHA challenge. Figure 1(1) shows an original challenge image, and (2) shows the image after segmentation, where eight valid characters were successfully located in the right order with each displayed in a different color (and most non-character lines removed).

(1) (2)

Figure 1. Segmentation. (1) A sample CAPTCHA challenge, (2) after segmentation, eight characters were identified. (Taken from [Yan and El Ahmad, 2008])

Segment	Label	Segment	Label
	X		5
	T		Y
	N		R
	M		E

Figure 2. Sample labeling: labeling each segment with the right character.

Recognition: using machine learning to recognize each segmented character. Key steps include 1) *labeling* each segment with the right character (see Figure 2), and 2) *training* a recognition engine (e.g. a convolutional neural network [Simard et al., 2003]) using a large number of the la-

beled segments (or "samples" in generic terms). In simple terms, the engine needs to be told what images represent an 'a', what images represent a 'b' and so on. Then, the trained engine can be used to recognize unlabeled character segments.

To fully evaluate the robustness of a CAPTCHA scheme, typically at least ten thousand segments have to be labeled [Chellapilla and Simard, 2004]. Since the strength of such CAPTCHA relies on the difficulty of computers to understand and decode distorted texts, sample labeling cannot be automated. Instead, each sample has to be **manually** labeled, which is tedious and expensive.

In this paper, we show that Magic Bullet (MB), a computer game, can turn the labeling of distorted characters, which would otherwise be tedious for humans and difficult for computers, into a fun experience. MB is a dual-purposed game, serving two purposes simultaneously: 1) like in any games, people play the game just for fun, and 2) although players might not realize it at all, their game play contributes to solving a real problem that has practical utilities but to which the current state of the art does not have a solution.

The structure of this paper is as follows. Section 2 describes the design of the game. Section 3 discusses its implementation and other details. Section 4 evaluates the fun level of the game, the accuracy and throughput of data produced by the game. Section 5 shows additional applications of the game. Section 6 compares the game and related work. Section 7 concludes the paper.

2 General Description of the Game

Magic Bullet is an online multi-player shooting game. In a typical setting, it is a four-player game in which two teams compete against each other, with two players in each team. The players are chosen and assigned randomly; they are not told who the other participants in their games are, nor are they provided with any means of communicating with each other.

All of the players will share the view of the same gaming area, a screen with two targets - one is for their team and the other for their opponents' team (see Figure 3). Each game session consists of an arbitrary number of rounds. During each round, a randomly chosen character image (which is typically a segmented CAPTCHA character, e.g. "M" in Fig 1) is shown to all four players. This image will become a bullet, starting off in a stationary position an equal distance away from the two targets. The team who shoots the bullet to hit their target wins the current round of the game.

The key game rules are as follows.

- To trigger the bullet's initial movement: the first player to hit any key on his/her keyboard will get the character image to become a bullet, moving slowly towards their target.
- To get the bullet to hit their target:
  - Both players in the team must type the same key for that bullet, i.e. they have to reach an agreement about the image. Note: the players are not required to type the key at the same time, but each must type the same key at some point while the image is on the screen. For example, for a character image resembling 'q', it is a valid match if one player has typed a sequence of 'g', 'o' and 'q' and his or her partner types a 'q'.
  - If both teams have reached an agreement, the team that managed to reach their agreement first wins - the bullet would zoom across the gaming area to hit their target, and score points for this team.

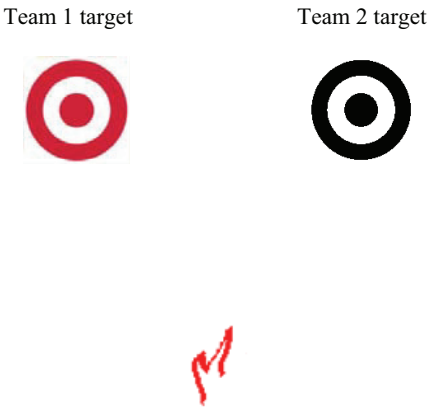

Figure 3. At the beginning of each game round

The movement of the bullet can drastically change as if by magic – for example, the bullet starts to move towards the target of the team that hit a key first. But if the other team reaches an agreement first, the bullet will change its direction to hit the winning team's target - hence the name "Magic Bullet".

If after 10 seconds since the starting of the round neither team were able to reach an agreement, the bullet would explode and neither team gets any points.

When a "bullet" has either hit a target or exploded, it signifies the end of one round. The session would then generate a new round with a different randomly chosen "bullet", and this cycle will continue until the 2 minutes time limit of each session. By then the final scores of both teams will be compared and the players will be informed if they have won or lost that session.

Security techniques are applied to make sure that players in the same game are geographically apart so that the likelihood of cheating will be dramatically reduced. The only honest way of wining the game is for each player to decide which character they think that "bullet" resembles the most, and press the corresponding key on their keyboards.

To have more likelihood of winning, the players will need to be both fast and accurate when they type. First, a team will get points only if they could agree before their opponents do. Therefore, the players must type quickly and

accurately. Sending in multiple randomly chosen keys is not an optimal way of playing. In fact, the fastest scenario for a team to reach an agreement is when the first keys they both send are a match. Moreover the faster the players can complete each round, the more rounds they will be able to play in that session, and thus the potential score they could get is higher.

As such, while people play the game of collaborative shooting, they also label each image with the correct character. The labeled images are useful output produced by the game.

3 Implementation and Other Details

MB is designed as a web-based game so that people can conveniently have access to it online. It is implemented with the Google Web Toolkit (GWT). GWT allows developers to write their application in Java, but then compiles it into optimized JavaScript for deployment. Since all major browsers support JavaScript by default, end-users do not have to install anything on their computers before they can play the game. On the contrary, if the game is implemented as a Java applet, users might have to install an appropriate Java Virtual Machine before they can play the game.

3.1 Cheating

Since the game relies on the team members collaborating in order to produce any useful data, cheating becomes a significant concern. This is because the players could also be collaborating for malicious purposes.

One example of cheating is when both members on the same team have previously agreed upon the use of a single key for all their answers. As a result, no matter which segments show up in their games, they will always be labeling it with the same character. This form of cheating will be very effective, because, since the cheaters have pre-empted the answers, they will no longer need to spend any additional time used in recognition. Therefore, their response times will generally be much quicker than the legitimate players of similar levels, and thus allowing them to win more often and attain higher scores.

To prevent this type of cheating, no communication should be allowed between the partners. We adopt the following cheating-prevention methods - some are taken from [Yan, 2003; von Ahn and Dabbish, 2004].

- *Player queuing and random pairing*: players who log on at the same or similar time will not necessarily be paired together.
- *IP address checks*: to make sure that players are not paired with themselves or with people who have a similar address.
- *Trap*. Some images are manually labeled and will be used as trap at a random interval. If players keep getting the trap images wrong then they will get flagged as cheaters and black listed, all data from them will be discarded.
- *Keyboard hamming detection*: This involves counting the number of unique inputs by a player. If it is over a threshold (e.g. 5) then the player is most likely randomly pressing keys.

3.2 Case sensitivity

Most CAPTCHA schemes known to us use only a single case of letters, for example, the Microsoft scheme [Yan and El Ahmad, 2008] used capital letters, and the Gimpy-r and Google CAPTCHAs used lower-case letters. For these schemes, case sensitivity of the labels we collect is not an issue, since we can simply convert the collected labels into the correct cases before we use them.

A very limited number of CAPTCHAs (including two recent schemes deployed by Yahoo) used both cases of letters, although this is not necessarily a good design choice. For these schemes, it appears that a player's input should be case sensitive in the game, in order to obtain accurate labels.

However, for both of the above scenarios, we prefer to have case insensitive input and so encourage players to use lower-case letters, for the following reasons: 1) it is more convenient for players to input lower-case letters than upper-case ones, and therefore more fun for them to play the game; 2) it is difficult, and sometimes even impossible, to tell whether a letter as distorted in CAPTCHA is in upper or lower case; examples include letters such as Cc, Kk, Oo, Pp, Ss, Uu, Vv, Ww, Xx, and Zz; 3) because of reason 2, the schemes using both cases of letters are in fact typically case insensitive, that is, answers to a CAPTCHA challenge are case insensitive. Therefore, labeling a letter with its opposite case does not really matter much in terms of the impact on the accuracy of the trained engine – this is true in particular for letters such as Cc, Kk, Oo, Pp, Ss, Uu, Vv, Ww, Xx, and Zz. For other letters such as *Aa*, *Bb* and *Dd*, even if accurate cases are required for their labels, it is a simple task. For example, when the letter is known, a simple geometrical analysis will tell whether it is in lower or upper case.

3.3 Bot

When there are not enough human players, our system will automatically enable bots to play with waiting people. There are two types of bots in our design: 1) a bot that simply replays data from old games (we call this a Data Replay Bot or DRB for short), and 2) a bot that performs actions at response times tailored to an opponent team's performance (we call this a Tailored Response Bot or TRB for short).

DRB is used to simulate a player's partner; it would simply fetch an existing label of a segment and set the label as the answer. All the player has to do is to input the same key as that label to reach an agreement.

A single TRB participates in a game as a team by itself, and it never has to decide which letter to press. Whether or not the TRB team reaches an agreement on an image is entirely the result of flipping a biased coin. Typically, a TRB monitors response times of its opponent team in previous rounds of the current game session, and generates some response times for its own team around those of the opponent team (or uses predefined values at the beginning of a session). At these intervals, TRB will flip a biased coin, with the result being either an agreement being reached or not.

The bias of the coin depends on the scores of the current game session; it will be more in favor of the TRB if the opponent team is winning and less otherwise. The purpose is to keep the TRB's score around that of its opponent team, this way the stronger player(s) in the opponent team can be pushed to test their abilities and the weaker player(s) would not feel too overwhelmed.

With these two types of bots, we can create the following bot game types: i). **Human player & DRB vs. TRB**. This game type can be used to either verify the correctness of the labels, or detect cheaters. ii). **Human player & Human player vs. TRB**. This game type is used to collect new labels when there are only two human players.

Although it is also possible to make Player & Player vs. Player & DRB game types, it would serve us no useful purpose. For instance, if new segments are used then the DRB cannot supply an answer thus making the game unfair. But if only labeled segments are used, then we are not utilizing the available players who could be labeling new segments.

Due to the fact that the players were not foretold who their partner or opponents are, it will not be apparent that they are actually playing against a bot. As a result the players would not get "the computer is cheating" feeling if they do lose the game.

3.4 Extensions

We have created a high score ranking list. In future implementations, we will also support player skill levels, so if a player scores above a number of points they go up a level. Both of these will encourage players to keep aiming for higher scores, and give the game even better replay value.

The team playing element could be extended in two different directions, either by increasing the number of players per team, or by increasing the number of teams. Of course it is also possible to increase both at the same time. The game rules would stay more or less the same; all of the players on the same team must reach an agreement before all other opposing teams in order to score points. With the multi-team extension it is also possible to have a hierarchical points system, to award points by the order of the teams reaching their agreements (e.g. 200 for first place, 100 for second, 50 for third… etc).

The principle of the game could also be applied to create other forms of collaborative shooting games (e.g. tank shooting and darts). There is, clearly, great scope to use graphics and audio effects creatively to present the game in the most appealing way.

4 Evaluation

Our evaluation includes two parts. First, we show that the game is indeed enjoyable. Second, we show the accuracy and throughput of the data produced by the game.

Since our game has not been formally released to the public, we carried out an evaluation with a pilot study. We recruited 26 volunteers (20 male and 6 female; 20 with a technical background and 6 with a non-technical background). They were in the age range of 20-30, and an informed consent was obtained from them.

The character segments used in our game were from well-known CAPTCHAs such as those used by Microsoft, Yahoo and Google, and the Gimpy-r scheme. The segments were produced by automated programs written by us.

4.1 The level of fun

We used a questionnaire to survey the level of fun that the players experienced when playing the MB game. Table 1 shows the average rating (on a five point scale) to questions related to the enjoyability of the game.

Question	Total responses	Rating		
		mean	std dev	% at 4 or above
Did you find the game fun to play? [a]	25	3.60	0.71	56
Did you like playing with your partner? [b] *	25	3.40	0.87	48
Are you likely to play this game again? [c]	25	3.36	0.95	44

Table 1. How enjoyable is Magic Bullet? Average rating on the scale of 1 to 5, provided by 25 players who filled in the survey after playing the game. Higher scores are better.

a 1=No fun at all, 2=not much fun, 3=average, 4=some good fun, and 5=extremely fun

b 1=strongly disagree, 2= somewhat disagree, 3= neither agree or disagree, 4= somewhat agree, and 5= strongly agree

c 1= highly unlikely, 2= unlikely, 3=maybe, 4= very likely, and 5=definitely

*Bots were disabled in order to measure this fun element.

To give a further idea for how much the players enjoyed the game, we include below some quotes taken from comments submitted by players in the survey.

> "The competition aspect was very good as was the unknown 'enemy'. - Having leader boards is a nice touch to increase the competitive nature and repeat plays."

> "Identifying the characters and to get it correct as much as possible is really fun. Knowing you win motivates you to play again and again ;-)"

> "Being competitive is fun; you're trying to reach the letters before the other team. It creates a decent, active pace."

> "(The main fun elements include) 'trying to recognise what I think the letter/number is and hoping that my partner looks at it the same way I do and also trying to be fast enough recognise it and type it in. I made lots of mistakes but it was fun. "

4.2 Data quality and throughput

Label accuracy

In our study, 45 game sessions (including incomplete ones) were played, in which in total of 1852 sample images were

used. A manual inspection shows that 1798 images were correctly labeled, giving an accuracy rate of 97.1%.

We examined all the failure cases, and identified three types of labeling errors (see Table 2). Type 1 errors have occurred as the segments are simply unreadable for human eyes. Type 2 errors were due to confusing characters that the players misjudged. Type 3 errors seem strange, but were identified as the result of a bug in our game prototype: a player's late reaction to an image displayed for the current round was sometimes mistakenly taken as input to the next round. With the bug fixed, this type of errors was no longer observed.

Therefore, only type 2 errors represent labels that were incorrectly labeled by the game. Consequently, the accuracy rate achieved by the game should be calculated as: `(1798 + #bug)/(1852 - #impossible)` = 98.4%, where `#bug` is the number of type 3 errors (=10) and `#impossible` the number of type 1 errors (=14).

	Image	Actual text	Label by MB*
Type 1 error		1(I)s(5)?	1
		IJ (?)	N
Type 2 error		U	t
		Y	K
Type 3 error		C	q
		2	g

Table 2. Three types of labeling errors (*generated as in a single session)

Automatic detection of problematic samples
By using the same set of samples in multiple game sessions, characters that would cause type 1 or 2 error can be automatically identified by our game. The idea is simple: different people will enter different inputs for the same character – letting the character explode is equivalent to enter an empty label – therefore, the existence of different labels for a sample image implies such a sample is problematic. In this way, wrong labels will be prevented from being used to train the recognition engine.

Throughput
On average, a single game session produced 25 (std dev=9.2) correct labels per minute, giving 1,500 labels per human hour. This rate of labeling does not seem to be particularly fast, but it can be explained as follows. First, some players were not touch typists so they typed slowly. Second, the labeling process was delayed by the images that no one could decode. Third, some players were not familiar with the game so their pace of play was slow.

However, when players get familiar with the game or typing, the labeling rate will be improved. More importantly, the game supports a large number (denoted by *n*) of parallel sessions. The throughout of the game can be quickly scaled by a factor *n*, which is constrained only by the network bandwidth and the game server's CPU and memory.

5 Application

Building better algorithms for handwriting recognition
When an algorithm for handwriting recognition is developed, it is routine to segment cursive texts and then manually label character samples. MB can replace such a manually intensive and laborious labeling process, and thus speed up the test and development of the algorithm.

A tool for training typing skills
Fast keyboard typing is key to winning the MB game. Anecdotal evidence shows that some people were beaten in MB not because they could not recognize the displayed characters fast enough, but because it took them longer to type the right key – for example, some of them had to look for the right key on their keyboard, while their opponents who had better keyboard skills could hit the right key without paying any attention to their keyboard. As such, MB can be a training tool to improve people's keyboard skills. Another way of looking at MB is that it is a typing game, in which an element of CAPTCHA is combined into.

Binary labeling
For some applications, two types of objects are required to be labeled respectively. For example, to evaluate the robustness of a CAPTCHA that requires people to tell whether an image is cat or dog, 13,000 sample images had to be manually labeled according to the content of the image [Golle 2008]. This is an example of binary labeling, and it cannot be automated by the state of the art of image recognition, which works poorly in recognizing cat and dog in the images – otherwise the CAPTCHA would have no reason to exist at all. However, MB can be extended to provide an effective human computation solution to this problem. The details of such an extension are discussed in a forthcoming paper.

6 Related Work

Our work is inspired by the ESP game [von Ahn and Dabbish, 2004], which pairs two randomly selected online players to create labels for images on the Web. Designed to improve the quality of image search, the ESP game worked well to label some content-rich images. But it would perform poorly at solving the problem that Magic Bullet is designed to resolve, for the following simple reason: it would be boring for people to play the ESP game when the images to be labeled contain only a single character.

A major fun element of the ESP game in our view stems from the fact that you have to try and guess how other people think. Typically, agreeing on an appropriate name for an image in the game creates an enjoyable feeling of "extrasensory perception" about each other within a partnership: "hey, I read your mind!" This fun element is highly depend-

ent on the choice of images used. Not much is known about exactly what kind of images are the most suitable for the ESP game – the present paper is the first discussion on this issue. This is probably why an option is provided in the ESP game to allow players to skip images that they find hard to agree on with their partners.

In our view, to maintain a reasonable level of enjoyment, images used in the ESP game must satisfy the following condition: the search space for labels of each image should be neither too big nor too small for two players to reach an agreement. For example, if the image to be labeled is as large as a 17" computer screen and contains tens of thousands of objects, then it is highly likely that two players will never agree with each other although each of them might have already entered hundreds of words to describe the image. On the other hand, if an image implies only a single possible label, it will be trivial for two players to reach an agreement. In both cases, the enjoyment from the ESP effect of "reading each other's mind" will quickly diminish. That is, being either too easy or too difficult to agree on an image is not good. This can also be explained by a common principle of making a game fun to play: it is essential (although tricky) to make sure that the difficulty level is right.

Since the ESP game supports taboo words, theoretically we can increase the difficulty of single character images by adding the obvious descriptions to the taboo list. However, this may be a step jump in difficulty from too easy to too hard. On the other hand, only the "obvious descriptions" are the labels that we want.

On the contrary, people can have considerable fun in MB by labeling a single character. The fun aspect that is lost because of the weaker level of enjoyment from the ESP effect in the MB game is compensated by other fun elements. For example, a player commented in the survey that "the slightly more interactive nature of this game (vs. something like Google Image Labeler) made it slightly more entertaining". Our empirical data also appears to suggest that MB has got its difficulty level right and thus is fun to play.

Moreover, the ESP game is of limited attractiveness to people who are not fluent in English, since they find it difficult to figure out the right words to describe the images, or perhaps simply cannot play the game at all. Anecdotal evidence suggests that some non-native English speakers found that the words on the taboo list were all that they could figure out, thus finding it very frustrating to play the game. On the contrary, what is required in MB is just the capability of reading individual Roman characters, which is almost a universal skill. Therefore, for this category of players, MB is much more enjoyable than the ESP game.

Other differences between the ESP game and MB include the following.

1) ESP is a two-player game, while MB is designed for at least four players.
2) ESP is a cooperative game in which the players are partners working together to obtain points. There is a high score table in the ESP game, so you can compete, but not with your playing partner. On the other hand, MB is a competitive game in which two teams play against each other, and the points earned by one team are the points lost by the other.
3) ESP can be extended in the way the MB is played (, but not vice versa, as discussed earlier). For example, in such an extended ESP game, four players can be divided into two teams. Each team tries to label the same image and the team who first reaches consensus wins points. Such an extended version will be a team-based competitive game, which is likely of more fun than the original game due to the new competitive aspect.

7 Conclusions

In this paper, we have for the first time identified a number of labeling problems that the famous ESP game would fail to address. We have also designed our own game, Magic Bullet, as the first effective solution to the identified problems, which no known computer algorithm can yet solve.

When human computation is used to generate or process data, there is a risk of error. This is confounded in a game setting where players may not be motivated to be careful, or even cheat. However, our empirical evaluation shows that the data generated from our game can be highly accurate.

Acknowledgments

Ahmad El Ahmad deserves a big "thank you" from us for providing CAPTCHA segments and other support. We are grateful for helpful comments from Brian Randell, Sacha Brostoff, Yongdong Wu, Xuhong Xiao and anonymous reviewers. We also thank all participants of our study.

References

[Chellapilla and Simard, 2004] K Chellapilla and P Simard, "Using Machine Learning to Break Visual Human Interaction Proofs", NIPS, MIT Press, 2004.

[Golle 2008] Philippe Golle. "Machine learning attacks against the Asirra CAPTCHA", CCS 2008, ACM Press. pp535-542.

[Simard et al., 2003] P Simard, D Steinkraus, J Platt. "Best practice for convolutional neural networks applied to visual document analysis", ICDAR 2003, pp.958-962.

[von Ahn and Dabbish, 2004] Luis von Ahn and Lora Dabbish, "Labeling Images with a Computer Game", CHI 2004, ACM Press. pp319-326.

[von Ahn et al., 2004] L von Ahn, M Blum and J Langford. "Telling Humans and Computer Apart Automatically", CACM, V47, No2, 2004.

[Yan 2003] Jeff Yan. "Security Design in Online Games". ACSAC 2003, IEEE Computer Society, pp286-295.

[Yan and El Ahmad, 2008] Jeff Yan and Ahmad S El Ahmad. "A Low-cost Attack on a Microsoft CAPTCHA", CCS 2008, ACM Press. pp 543-554.