The End is Nigh: Generic Solving of Text-based CAPTCHAs

By Elie Bursztein, Jonathan Aigrain, Angelika Moscicki, John C. Mitchell

Published at: WoOT (Usenix Workshop on Offensive Technology)
Year: 2014
URL: https://www.elie.net/publication/The end-is-nigh-generic-solving-of-text-based-captchas
Slides: http://www.slideshare.net/elie-bursztein/the-endisnighgenericsolvingoftextbasedcaptchasslides
Related research: https://www.elie.net/tag/captcha

The End is Nigh: Generic Solving of Text-based CAPTCHAs

Elie Bursztein
Google
elieb@google.com

Jonathan Aigrain
Stanford University
jonathan.aigrain@gmail.com

Angelika Moscicki
Google
moscicki@google.com

John C. Mitchell
Stanford University
jcm@cs.stanford.edu

Abstract

Over the last decade, it has become well-established that a captcha's ability to withstand automated solving lies in the difficulty of segmenting the image into individual characters. The standard approach to solving captchas automatically has been a sequential process wherein a segmentation algorithm splits the image into segments that contain individual characters, followed by a character recognition step that uses machine learning. While this approach has been effective against particular captcha schemes, its generality is limited by the segmentation step, which is hand-crafted to defeat the distortion at hand. No general algorithm is known for the character collapsing anti-segmentation technique used by most prominent real world captcha schemes.

This paper introduces a novel approach to solving captchas in a single step that uses machine learning to attack the segmentation and the recognition problems simultaneously. Performing both operations jointly allows our algorithm to exploit information and context that is not available when they are done sequentially. At the same time, it removes the need for any hand-crafted component, making our approach generalize to new captcha schemes where the previous approach can not. We were able to solve all the real world captcha schemes we evaluated accurately enough to consider the scheme insecure in practice, including Yahoo (5.33%) and ReCaptcha (33.34%), without any adjustments to the algorithm or its parameters. Our success against the Baidu (38.68%) and CNN (51.09%) schemes that use occluding lines as well as character collapsing leads us to believe that our approach is able to defeat occluding lines in an equally general manner. The effectiveness and universality of our results suggests that combining segmentation and recognition is the next evolution of catpcha solving, and that it supersedes the sequential approach used in earlier works. More generally, our approach raises questions about how to develop sufficiently secure captchas in the future.

1 Introduction

Many websites use CAPTCHAs [39], or *Completely Automated Public Turing tests to tell Computers and Humans Apart,* to block automated interaction with their sites. For example, GMail uses captchas [1] to block access by automated spammers, eBay uses captchas to improve its marketplace by blocking bots from flooding the site with scams, and Facebook uses captchas to limit creation of fraudulent profiles used to spam honest users or cheat at games. The most widely used captcha schemes use combinations of distorted characters and obfuscation techniques that humans can recognize but that may be difficult for automated scripts. Captchas are sometimes called *reverse Turing tests*, because they are intended to allow a computer to determine whether a remote client is human or machine.

Due to the proficiency of machine learning algorithms at recognizing single letters, it has become well-established that a captcha's ability to withstand automated solving lies in the difficulty of segmenting the image into individual characters [12, 10]. The standard approach to solving captchas automatically has been a sequential process wherein a segmentation algorithm splits the image into segments that contain individual characters, followed by a character recognition step that uses machine learning [13]. This is known as the *segment then recognize* approach. While this approach has been effective against particular captcha schemes [15, 4], its generality is limited by the segmentation step, which is hand-crafted to defeat the distortion at hand. No general algorithm is known for the character collapsing anti-segmentation technique used by most prominent real world captcha schemes. This technique is called *negative kerning*. It is a variant of the object occlusion problem, which is a difficult problem in computer vision.

[1] For readability purposes, we will write the acronym in lowercase.

This paper introduces a novel approach to solving captchas in a single step that uses machine learning to attack the segmentation and the recognition problems simultaneously. Performing both operations jointly allows our algorithm to exploit information and context that is not available when they are done sequentially. At the same time, we remove the need for any hand-crafted component, making our approach generalize to new captcha schemes where the previous approach can not.

We were able to solve all the real-world captcha schemes that we evaluated (section 7) with accuracy considerably above the 1% threshold necessary [10] to consider a captcha scheme insecure. Our algorithm is able to achieve a *38.68%* recognition rate on Baidu 2011, *55.22%* on Baidu 2013, *51.09%* on CNN, *51.39%* on eBay, *22.67%* on ReCaptcha 2011, *22.34%* on ReCaptcha 2013, *28.29%* on Wikipedia, and *5.33%* on Yahoo without any tuning. Our algorithm also recognizes occluding lines in a generic manner, and its success against the Baidu (38.68%) and CNN (51.09%) schemes indicates that it is in fact well-suited to deal with them.

The effectiveness and universality of our results suggests that combining segmentation and recognition is the next evolution of automated catpcha solving, and that it supersedes the sequential approach used in earlier works. After comparing the accuracy of our algorithm with the accuracy of humans in section 7, we suggest that purely text based captchas may be nearing their end, and provide early steps toward rethinking how reverse Turing tests can be performed securely (section 9).

The reminder of the paper is organized as follows: We start by discussing research ethics in section 2. Section 3 summarizes the previous approach and its limitations. Then in section 4, we describe how we assembled our dataset of real-world captchas. The core of our algorithm is presented in section 5. We discuss optimizations in section 6. We report the performance against real-world schemes in section 7. Ways to improve the algorithm are discussed in section 8. We speculate on what the future holds for captchas in 9. Further related work is discuss in 10, with our contributions summarized in section 11.

2 Ethics

This work is a joint collaboration between Google and Stanford University. The results were validated by Stanford University. While we disclose an attack in this paper for which we don't have an easy solution, we believe that attracting attention to the issue outweighs the risk of malefactors taking advantage of this work.

In particular, our algorithm is complex and more costly to reproduce than employing cheap manual labor to solve captchas [38]. We believe that providing details about how such a result can be achieved will enable the industry and academia to advance the state of the art in reverse Turing tests. Specially now that concurrent work with similar results were announced but without any of the technical detailed to reproduce their results [26]. We also note that we gave early notice of this technique to the companies whose captchas were evaluated in this publication.

3 Background

In this section we discuss the previous approach used to break captchas and its limitations.

Negative kerning, also known as character collapsing, uses negative space between characters to resist segmentation by ensuring that each character is occluded by its neighbors. Figure 1 shows a typical captcha that uses negative kerning. This technique is usually supplemented with noise, distortions, and randomization to prevent side channel attacks, for example to prevent an attacker from making educated guesses where to cut if a captcha always contains the same number of letters. [10].

Figure 1: Example of a Yahoo captcha that uses the negative kerning

As of 2013, negative kerning is considered the most secure method for preventing segmentation because it has successfully withstood years of attacks. Almost all of the most prominently used captcha schemes rely on it, as discussed in section 4. The other method of choice to prevent segmentation, which seems to have fallen out of fashion after a successful wave of attacks [47], is to use occluding lines.

The standard approach to solving captchas automatically has been a sequential process wherein a segmentation algorithm splits the image into segments that contain individual characters, followed by a character recognition step that uses machine learning [13]. This approach, known as the *segment then recognize* approach and illustrated in figure 2, works equally well on image and audio captchas (see [37, 13, 43, 47, 9, 8, 15]).

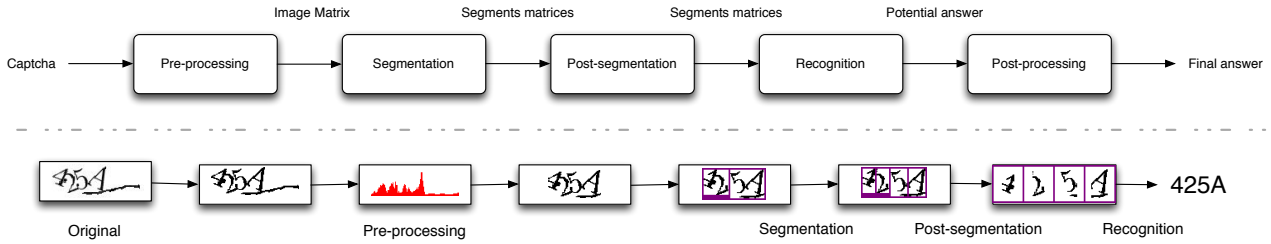

Figure 2: The segment then recognize approach illustrated

Over the last decade, it has become well-established [12, 10] that a captcha's ability to withstand automated solving lies in the difficulty of segmenting the image into individual characters rather than recognizing the characters themselves. A seminal work from 2005 demonstrated that machine learning algorithms consistently outperform humans for single character recognition [12]. Thus the innovation in automated captcha solving shifted from *optical character recognition* (OCR) to solving computer vision problems such as object occlusion in order to segment characters. To date, no general algorithm for character segmentation is known.

Previous work related to automated captcha solving falls roughly into two categories: The first type of attack uses side-channel information unrelated to the distortion itself, e.g., dictionary attacks [6, 4]. We do not dwell on this type of attack because it is usually trivial for the defender to patch, and because the goal of this work is to treat captchas in a generic manner.

The second type of attack focuses on finding weaknesses in the distortion algorithms of particular captcha schemes. One example of a precisely tuned segmentation algorithm is [15], where the authors used a complex image preprocessing phase that relies on character alignment, morphological segmentation with three-color bar character encoding and heuristic recognition to break reCaptcha 2011. While it was very effective against reCaptcha 2011, it does not generalize to other captcha schemes that use similar distortion techniques. Similarly in 2013, a group or researchers examined hollow captcha specifically and were able to solve all of them using an extended segment then recognize approach that involves 9 consecutive steps [21]. Our single step approach yields results that are 4.22% more accurate on the Baidu 2013 scheme. We compare in depth our results with previous work in Section 7.

To date, research in captcha solving has followed the familiar exploit-patch cycle where the attacker finds a flaw in a particular anti-segmentation technique, and the defender patches it or moves on to a new one. The limitation of the segment then recognize approach has been the attacker's ability to find new flaws. As we will show in our evaluation (section 7), our work overcomes this limitation by segmenting and recognizing the captcha simultaneously, thus removing the need for manually discovered heuristics to segment captchas.

4 Dataset

Following the methodology of [10], we created a corpus of real-world captchas to evaluate the effectiveness of our algorithm. We focused on unbroken real-world captcha schemes, and ended up creating our corpus from the schemes depicted in Figure 3.

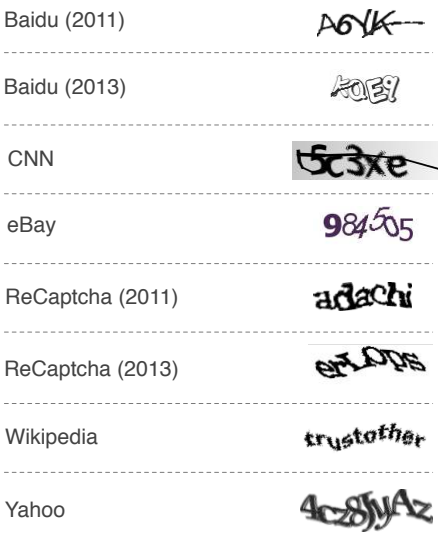

Figure 3: Examples of the captchas of the schemes we evaluated

Unsurprisingly, all of them rely at least to some extent on negative kerning to prevent segmentation. The captchas were directly downloaded from their respective websites and annotated via Mechanical Turk [28], with IRB approval.

Since 2011, some of those schemes, namely Baidu and ReCaptcha, have evolved. To keep our algorithm evaluation relevant to the state of the art in captcha design, we extended our corpus to include the new versions of Baidu and ReCaptcha in 2013.

As visible in figure 3, Baidu and ReCaptcha evolved in two radically different ways: Baidu decided to use hollow letters whereas ReCaptcha introduced more aggressive distortions. As discussed in section 7, our success rate decreased on the new version of ReCaptcha compared to the previous version. On the other hand, surprisingly, its accuracy significantly increased on the newer version of Baidu.

5 Algorithm

In this section we provide an overview of our algorithm and describe its major components. Then we discuss the reinforcement learning process that is central to the accuracy of our algorithm. Finally, we explain how to handle occluding lines in a generic manner since it is a natural extension of our algorithm. We leave the discussion of optimizations and trade-offs that can be applied to the algorithm for the next section.

5.1 Algorithm overview

The main idea is to use a machine learning algorithm to score all possible ways to segment a captcha and decide which combination is the most likely to be the correct one. Analyzing all the possible segmentation paths allows the algorithm to find the path (set of segments) that globally maximizes the recognition rate. We contrast this with the segment then recognize approach where an uninformed segmentation algorithm passes at most a small number of possible segmentations to an independent recognition algorithm.

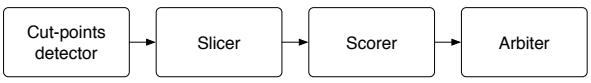

Figure 4: Overview of the algorithm's four components

As depicted in figure 4, the algorithm is composed of four components: the *Cut-point Detector* that finds all the potential ways to segment a captcha, the *Slicer* that is responsible for extracting the segments and combining them into a graph, the *Scorer* that performs OCR on the segments and assigns a recognition confidence score to each one of them, and the *Arbiter* that is responsible for processing the scores and determining what are the most likely letters.

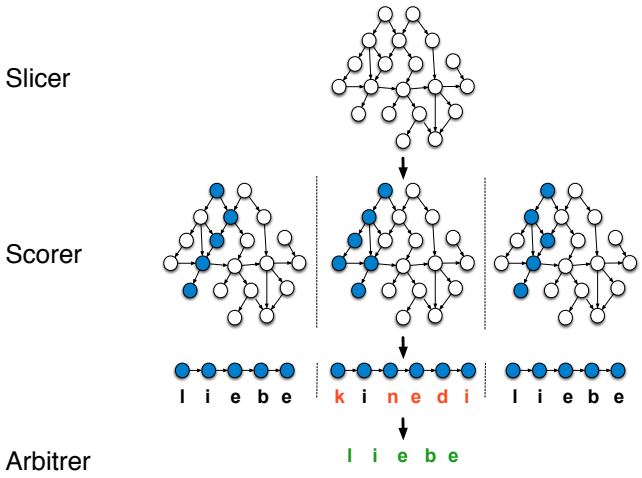

Figure 5: Illustration of how the algorithm works

Figure 5 illustrates the graph representation that we use to keep track of all possible segmentations at once. This structure is what enables us to simultaneously solve the recognition and segmentation problems. A concrete example of the algorithm's output while operating on a Yahoo captcha is shown in Figure 6. We now describe each component in turn.

Cut-point detector: The cut-point detector is responsible for finding all the possible cuts along which to segment a captcha into individual characters. The potential cuts are found by examining the second derivative of the curve generated by following the bottom pixels of the captcha, and the curve generated by following the top pixels of the captcha. An example of this step is shown in figure 6 in the second image from the top. We use the inflection points as potential end points for each cut. These are marked in red for the top line and in blue for the bottom one. Each cut is constructed by connecting the inflection points - one from the top, and one from the bottom.

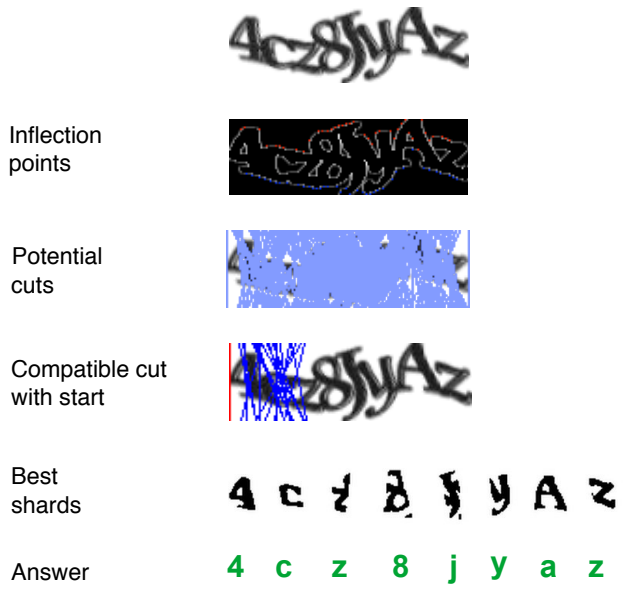

Figure 6: Example of the algorithm successfully applied to a Yahoo captcha

As is shown in the center image in figure 6, this procedure generates a large number of cuts that will be processed later by the slicer. We emphasize that this technique is purely geometric and is designed to be simple; no specific tuning is necessary. In section 6, we discuss changing the number of cuts to make a trade-off between speed and accuracy.

Slicer: The slicer applies some heuristics to extract the meaningful potential segments based on the cut points and builds the graph in figure 5. A potential segment is considered meaningful if the two cuts that define its left and right boundaries are sufficiently far apart (90% of the width of the smallest character observed in training), yet not too far apart (110% of the width of the largest character seen in training). The selection process is illustrated in the second image from the bottom in figure 6. As is visible in the figure, even for the first character we end up with a large number of potential segments, each of which will generate even more subsequent potential segments because we use them as right boundary for the next one. This exponential number of segments caused early versions of our algorithm to be very slow (but still computationally feasible). During the early phase of research, it was not uncommon that a 12 letter long captcha took up to 9 hours of computation. We have since improved performance, discussed in section 6.

Scorer: The scorer traverses the graph of potential segments, applies OCR to each potential segment, and assigns a recognition confidence score. It uses a modified version of KNN [16] to perform the OCR. KNN works by computing how far each potential segment is from the k closest known (learned) examples to decide the most likely character depicted in the potential segment. Segments are processed at the pixel level, as this has been demonstrated to be the best approach for text recognition [31]. The fact that it is easy to produce a mathematically sound score from the recognition process was one of three factors that led us to settle on KNN. The other two factors were noise resistance and computational speed given our feature set domain.

The noise resistance arises from using a relatively small k (less than 10) in our KNN to identify the nearest neighbors. This is essential in our case because most of the potential segments generated by the slicer are meaningless and belong to the garbage class. If we had used a margin based classifier (e.g. SVM), we would have had to compensate for this bias at the expense of accuracy due to the class *contiguity hypothesis* [34]. Our brief experimentation with an SVM linear classifier supports this assumption.

Using a simple metric distance produced a poor recognition rate, so we modified our distance function. We realized that the problem was assigning an equal weight to each pixel regardless of its position in the segment or its grayscale value. It turns out that pixels on the edge of segments are less meaningful than pixels in the center precisely because they are shared between characters that have been collapsed together.

We achieved very good results on all captcha schemes by assigning higher weight to pixels nearer the center of the segment, and to darker ones. While this approach is adequate, we believe that the algorithm could be improved by learning the weights from the data itself. We discuss this idea in more detail in section 8.

Arbiter: The final component of our algorithm is the arbiter that selects the final value for the captcha. The arbiter uses an ensemble learning approach [19] where each sequence of segments has a vote weighted by the recognition score confidence. This is very similar to the random forests algorithm except that trees are replaced with paths in our case. Similarly, random features are replaced by multiple segmentations of the same set of characters.

Ensemble learning, while deceptively simple, is very robust to noise and leverages the fact that there is not one but multiple ways to segment an occlusion based captcha that will yield the correct answer. The Condorcet theorem [40] implies that combining paths yields better predictive performance than could be obtained from a single path in isolation. In our experience, the right voting approach is critical to the algorithm's success. While this approach is accurate, it is also very slow to the point of being impractical, which is why we discuss trade-offs that can be applied to make the algorithm orders of magnitude faster while retaining most of its accuracy in Section 6.

5.2 Reinforcement learning

The traditional way to train a character classifier is to provide a set of labeled captchas already segmented and let the classifier learn to recognize each character from those segments using the labels. This process assumes that the classifier is given the correct number of segments - one for each letter in the captcha. In the segment then recognize approach, this assumption holds because the segmentation is handled by a vision algorithm that is not part of the classifier itself.

In our case, this assumption does not hold. Our algorithm produces a huge number of segments, most of which are garbage, and many paths in the graph that do not correspond to valid segmentations at all. To overcome this, we use a reinforcement learning approach where the human fills an entirely different role: instead of providing the classifier with labeled examples of valid segments, the algorithm asks the human to annotate segments that have been misclassified, and then learns from the feedback. The algorithm is able to initiate the learning process because the first two components of the algorithm (the cut-point detector and the slicer) are fully unsupervised.

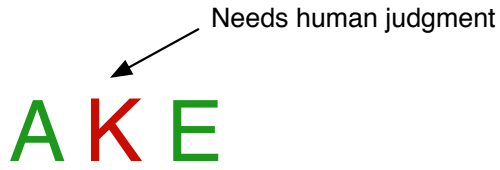

Figure 7: The algorithm asks for feedback when a segment surrounded by two correctly classified segments is misclassified

In a nutshell, our reinforcement learning works as follows: during training, the algorithm processes a set of labeled captchas and isolates the captchas that were not successfully recognized. For the failed captchas, the algorithm asks for human feedback when a segment surrounded by two correctly classified segments is misclassified (see figure 7). In those cases, the algorithm needs the human expertise because the misclassification could be due either to improper segmentation, or to bad recognition and the algorithm is unable to tell them apart by itself. Examples of improper segmentation and bad recognition are shown in figure 8. If the error was due to improper segmentation, the segment is discarded. If the error was due to a recognition error, the segment is added to the classifier training set. When all the cases are reviewed, the algorithm is retrained with the enriched dataset. In practice, even a single round of reinforcement learning is enough to significantly improve accuracy, as summarized in table 3. The number of cases requiring manual intervention was small enough where we performed the corrections ourselves.

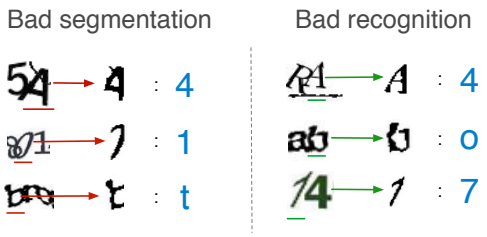

Figure 8: Examples of misclassification due to bad segmentation and bad recognition

5.3 Occluding lines

The last fundamental hurdle we needed to overcome to produce a fully generic algorithm was to deal with occluding lines. While not as popular as negative kerning, lines are used by some captcha providers. For example Baidu 2011 (figure 9) uses both lines and negative kerning to defend against automated solvers.

	Without the line class	With the line class
Baidu	8.78%	17.27%

Table 1: Impact of adding a line class on the recognition rate for the Baidu scheme without iterative learning

Initially we struggled with lines and unsuccessfully tried numerous approaches, including an independent line removal algorithm based on a soft margin classifier. In the end the solution was surprisingly simple: we added a new character class to the scorer for line segments. For example, we were able to increase our recognition rate by *8.49%* on the Baidu 2011 scheme when this class is added, as reported in table 1. Although the solution is simple, the reasons why it works are not. First, we believe adding a class is effective because of how our algorithm is designed; the KNN algorithm permits discontinuous character classes, which allows many different shapes of line to fall into the same class. Secondly, the cut point detector uses the second derivative to find potential cuts, which is well suited for ignoring flat parts. This is especially clear in figure 9; there are very few cut points on the line itself.

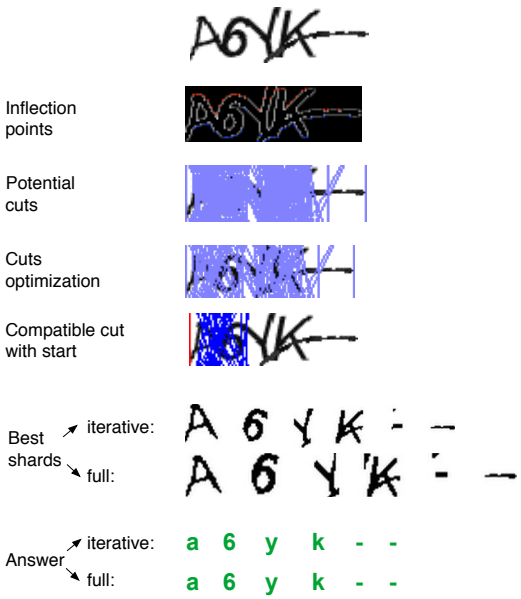

Figure 9: Example of adding a line class to the classifier, illustrated on a Baidu captcha. The "-" in the answer represents the line class

6 Optimizations

In this section we discuss optimizations that make the algorithm run faster at the expense of accuracy. Drastically reducing the computation time with a limited loss of accuracy was necessary to make our algorithm practical. We note that all heursitics proposed in this section apply equally well to all types of distortions, and thus do not impact the generality of our approach.

6.1 Reducing the number of cuts

Since the computation time is directly related to the number of potential segments, our first optimization was to come up with heuristics to reduce the number of cut points considered by the cut point detector. This optimization works by pruning near-duplicate and improbable cuts from the set of potential cut points. First, we removed all the cuts that have an angle $> 30^o$. Then we examined the ratio of white pixels to black pixels to eliminate cut lines that pass through too many black pixels, since they are most likely cutting through the middle of a letter. Finally we compared the pixel intensities of the left and right boundaries to estimate whether the cut marks a transition between two letters. As illustrated in figure 10, when these heuristics are applied, the number of potential cuts decreases visibly.

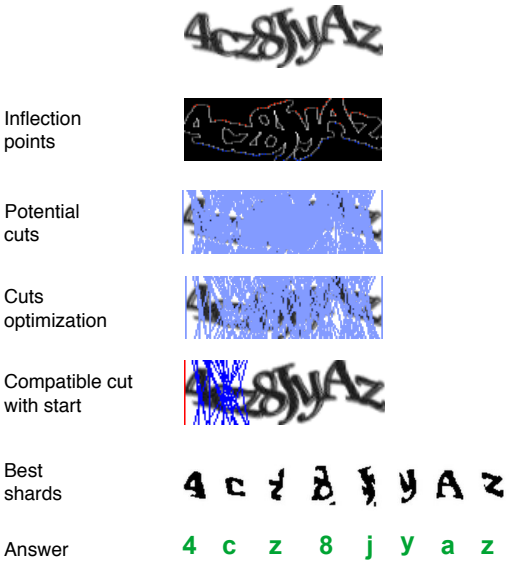

Figure 10: Example of the cut point reduction heuristic applied to a Yahoo captcha

As a more quantitive example, the number of segments considered on Baidu 2011 decreased by a factor of 8, as reported in table 2. We chose to use the Baidu 2011 corpus to evaluate this optimization because the Baidu captchas are the shortest and therefore the fastest, with only 4 letters. As reported in the table, with this optimization accuracy dropped from 21.05% to 17.39% (-3.6%). However, we also observe close to a 63x speed up. The time spent per captcha decreases from 246 seconds to 3.9 seconds. We ran this experiment without the reinforcement learning in order to produce a fair accuracy comparison, since the set of segments that the algorithm asks a human to evaluate will differ due to the optimization.

	Recognition	Time	Segments
Baidu all cuts	21.05%	246s	8820
Baidu pruned cuts	17.27%	3.9s	1192

Table 2: Algorithm recognition rate (recognition), average solve time (time) and average number of segments (segments) with and without the cut point reduction heuristic

6.2 Sequential recognition

The computational cost of our algorithm increases exponentially with the length of the captcha, to the point of becoming prohibitive on long captchas such as the Yahoo scheme. As explained in section 5, the complexity is due to generating the graph of all possible segment combinations and then evaluating all paths.

To mitigate this problem, we developed a version of the algorithm that makes local decisions for character recognition rather than looking at the entire captcha. As depicted in figure 11, we found out that the best approach when making a local decision is to consider a window of two letters at a time. Considering two characters at a time yielded significantly better results than looking at one or three characters at a time. To make a local decision, the algorithm attempts all possible cuts for a given window and keeps the pair of letters and cuts that maximizes the recognition score for the pair. Making a local decision is of course suboptimal and prone to errors that don't affect the global approach, but in practice accuracy decreased by less than 5% in all cases.

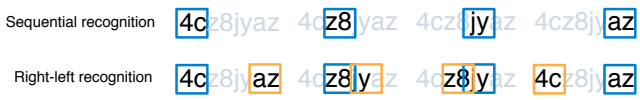

Figure 11: The sequential and right-left recognition processes

Improving the sequential approach We discovered that the accuracy of the solver depends heavily on the position of the character in the captcha as shown in figures 12 and 13. The closer a character is to the center of the captcha, the less accurate the algorithm is. We also observed that the sequential recognition approach is less accurate on the right side of the captcha than on the left side. This is particularly true for long captchas such as the Yahoo scheme (figure 13).

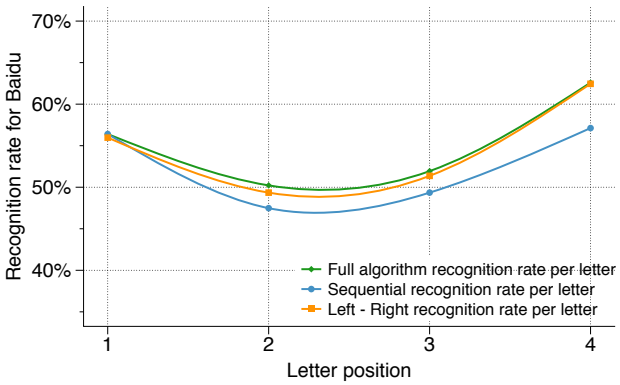

Figure 12: Recognition rate per letter for the each learning approach on the Baidu 2011 scheme

This observation led us to the idea of performing the sequential recognition from both directions and then combining the two recognition scores to improve the overall accuracy, as illustrated in figure 11. We call this the *left-right* approach. Figures 13 and 12 show that this approach greatly increases the accuracy on the right side of the captcha and often improves the overall accuracy compared to the simple sequential approach (table 3).

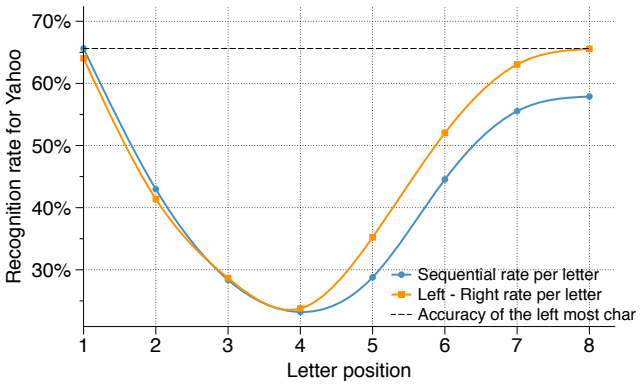

Figure 13: Recognition rate per letter for the various approaches for the Yahoo! scheme.

Moving from a global recognition to a local one significantly improved the speed of our algorithm. For example, it reduced the computation time for an eBay captcha from *35.79* seconds down to *2.36* seconds, which is a 15x speed up on top of the gains from the cut-point elimination heuristic. These optimizations allowed us to run our algorithm over large corpuses of captchas and made our approach practical.

	Reinforcement learning				Simple learning			Previous Work		
	Full	L-R	L-R Time	Seq.	Full	L-R	Seq	Accuracy	Delta	Ref.
Baidu (2011)	38.68%	33.42%	3.94 s	36.58%	17.27%	16.55%	16.69%	5%	+33.6%	[10]
Baidu (2013)	55.22%	54.38%	1.9 s	54.38%	-	-	-	51%	+4.22%	[21]
CNN	-	51.09%	4.9 s	48.54%	-	46.40%	45.96%	16%	+35.09%	[10]
eBay	51.39%	47.92%	2.31 s	48.61%	39.43%	40.14%	36.29%	43%	+11.4%	[10]
ReCaptcha (2011)	22.67%	21.74%	7.16 s	19.25%	19.86%	18.25%	17.10%	40.4%	-17.73%	[15]
ReCaptcha (2013)	22.34%	19.22%	4.59 s	19.74%	20%	14.61%	12.77%			
Wikipedia	-	28.29%	-	26.36%	-	27.02%	26.24%	25%	+3.3%	[10]
Yahoo	-	3.67%	7.95 s	5.33%	-	2.72%	2.29%			

Table 3: Recognition rates for real-world schemes. Full denote the full algorithm, L-R denote the Left-Right algorithm , Seq denote the Sequential algorithm, "L-R time" for the time the Left-Right algorithm takes to solve a captcha on average.

7 Evaluation

In this section we report how our algorithm performed against real-world captchas schemes. Table 3 summarizes our results. Following the best practices proposed in [10], our evaluation was performed on a test set of approximately 1000 captchas for each captcha scheme that were not used during training. The evaluation was performed on a core-i5M laptop. The algorithm was trained the same way once for each scheme without changing any of the algorithm's parameters.

We ran the algorithm with and without the various optimizations described earlier to evaluate how they impact recognition rate. All results include the cut-point elimination heuristic.

For certain cases the recognition rates are not available because the computational cost of running the algorithm was prohibitive on our test set (over 24 hours of computation). We compare our results to previous work when the data is available in column *Ref*. To establish a fair comparison between the various schemes we normalized the number of examples per character to 26, which in practice meant a very small training set of well under 1000 captchas in all cases. We believe that normalizing the number of examples per character results in a more accurate comparison because different schemes use different character sets.

Our algorithm was able to solve every scheme with accuracy significantly above the 1% success rate necessary to deem a captcha scheme insecure [10]. The algorithm in its best configuration is able to achieve *38.68%* on Baidu 2011, *55.22%* on Baidu 2013, *51.39%* on eBay, *51.09%* on CNN, *22.67%* on ReCaptcha 2011, *22.34%* on ReCaptcha 2013, *28.29%* on Wikipedia, and *5.33%* on Yahoo. On average the reinforcement learning provides a *6.7%* accuracy improvement. Using the sequential decision instead of the global decision decreases accuracy on average by *3.42%*, the left-right only decreases it by *1.75%*. In terms of speed, the algorithm takes on average *6.22s* to process a captcha, which makes it not only practical, but indeed comparable to the speed at which humans are able to solve captchas [11].

While still in its infancy, our algorithm in most cases outperforms previous work that relies on manually generated segmentation algorithms. More importantly, our approach does not suffer from the brittleness inherent in attacks manually tuned against particular distortions. For instance our approach outperforms [10] and [21]. While [15] outperforms our algorithm on reCaptcha 2011, the authors acknowledge that their approach is not able to solve the CNN captcha scheme whereas our algorithm solves both without modification or tuning. Overall there is no previous work that is effective against the breadth of captchas schemes presented in this paper. This leads us to believe that a unified approach is likely to replace the segment then recognize approach.

7.1 Learnability

Figure 14 shows how overall accuracy improves as a function of the number of training examples per character. We confirm the findings of [10] that it does not take very many examples to achieve a sufficient accuracy rate. This figure also shows that the left-right approach does not seem to require more examples than the global one.

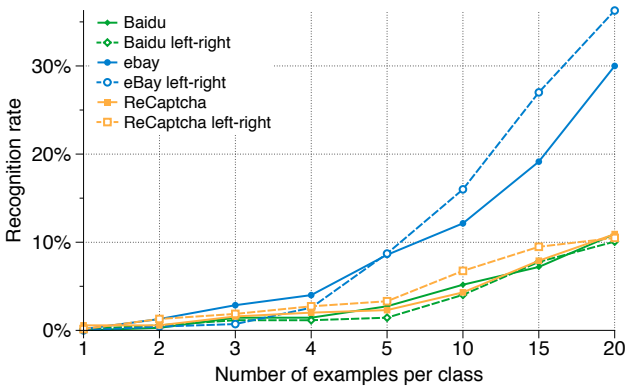

Figure 14: Recognition rate as a function of the number of example in each class.

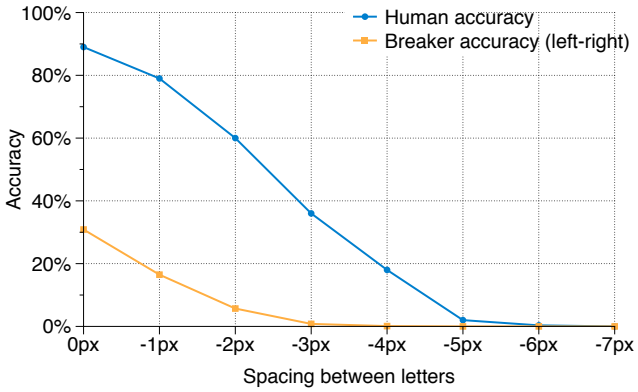

Figure 15: Human and Algorithm accuracy vs. spacing between letters in pixel

7.2 Human accuracy

To complete our investigation of negative kerning, we also quantified human accuracy compared to the accuracy of our algorithm at different levels of distortion. We ran an experiment, with IRB approval, on Internet users using Amazon Mechanical Turk [28]. We asked Turkers to solve 2000 captchas for each kerning that ranged from 0 pixels to -7 pixels (16 000 captchas total). The captchas were 6 characters long, drawn at random from the character set a-z and use the Arial font in 20px. We discarded captchas that were solved too quickly ($< 4s$) or to slowly ($> 15s$) as they were most likely not honest attempts to solve the captcha. We ran our algorithm on those 8 kerning variations as well.

Figure 15 shows the result of the experiment. The gap between human and machine accuracy for negative kerning based distortion is too narrow to be used reliably as a reverse Turing test. Driving the algorithm's recognition rate close to 0% using solely this type of distortion will lead to a horrendous human recognition rate ($< 20\%$). We acknowledge that fully understanding human vs. machine ability to process distortion is a fascinating problem in its own right, and we leave it to future work. Nevertheless this experiment supports our claim that devising effective text-based captchas is very difficult.

8 Areas of improvement

While our algorithm produces good results, it is just the first rough implementation of the new holistic approach. This section highlights some of the most promising directions for improvement.

Learn the KNN weights The current implementation uses a single manually chosen set of weights for the KNN distance computation that performed well on our corpus. We believe that automatically learning those weights for each captcha scheme would improve accuracy, particularly for schemes that use unusual fonts or specific distortions. We believe that it is possible to accomplish this fully unsupervised, similar to the cut-point detector and slicer phases of our algorithm.

Improve cut-point elimination As discussed in section 6.1, we rely on a set of heuristics to remove unlikely cut lines to increase the speed of the algorithm. As our evaluation suggests, this heuristic generates a drop in accuracy (*-3.6%* on Baidu). Finding a better set of heuristics that are both generic and more precise is an open question.

Additional Occlusion As pointed out earlier, Baidu and CNN captcha schemes use occluding lines with low curvature. While our results on these captcha schemes are very good and our algorithm properly detects lines (see section 5.3), future work should investigate in depth how various types of lines, e.g., sine waves that have a high curvature, impact the recognition rate. It should also consider other types of occlusion, e.g., blobs. To date, we have not found real world captcha schemes that employ this type of occlusion; perhaps occlusion of this type presents usability challenges that make it impractical for humans.

Explore deep neural networks A primary contribution of this work is to empirically demonstrate the effectiveness of performing segmentation and recognition simultaneously. Accordingly, we have considered other algorithms that are able to process captchas in a holistic manner.

In particular, with collaborators, we have experimented with deep convolutional neural networks, similar to those in [29]. These experiments have confirmed the benefits of a unified approach, and have achieved captcha-solving results that equal or improve upon those presented in this paper. For certain ReCaptcha data sets, these new results show such dramatic improvement in accuracy, while using large-scale training sets, that they suggest that deep neural networks may hold a substantial advantage over humans for solving text-based captchas [23].

9 The future of captchas

When captchas were invented, the community realized that with the passage of time one of two things would happen: either captchas would remain an invaluable way to differentiate humans and computers, or very high quality OCR would become readily available [29]. We believe that our approach of solving the segmentation and recognition problems in a single step ushers in the latter. This breakthrough not only affects the way solvers are designed, but also forces us to reconsider from ground up how reverse Turing tests should be done. This is a very complex open question that is generally outside the scope of this work, but we have gleaned some insights on potential promising directions for developing next-generation reverse Turing tests, both from this work and from our experience at Google.

Moving to a more complex domain The first potential direction is simply to find a more difficult problem in computer vision. Incorporating video or requiring the user to perform a higher order cognitive task such as circling or rotating an object [24] are examples. A significant amount of work has been produced on alternative captchas of this type [36, 20, 17]. However, many proposals do not meet a high enough standard of universal accessibility, since reverse Turing tests must be solvable by any human.

For example, English language comprehension is clearly not a general reverse Turing test. Past experience has shown that finding a better common denominator than text transcription that is also difficult for machines is elusive. A good example is the Asirra captcha, depicted in figure 16, which asked users to distinguish between cats and dogs. Less than a year after its release it was successfully broken using a classifier trained to recognize image textures [22]. More recently, the MintEye captcha scheme [3], that relies on "undistorting" an image (figure 17) was broken by a very simple attack based on Sobel operators that only required 23 lines of Python [2].

Figure 16: Example of the Asirra captcha that asks users to distinguish between cats and dogs

On the other hand, casting the problem of text transcription into a more complex domain (video) has also proven difficult. NuCaptcha attempted to do this, and was broken by two different teams [7, 45] using different approaches. Mitra et. al. have suggested using emergent images as an alternative way to encode information in video that might be robust against computer vision algorithms [36]. Whether there exist situations where the human brain is definitively able to process information more efficiently than machines remains an open question.

Recently game-based captchas have been developed [1], however implementing this idea as proven to be difficult, as the game captcha schemes for the leading game captcha provider "Are you a human" have been broken [42].

Figure 17: Example of the MintEye captcha that ask users to "undistort" an image.

Using cognitive behavior Another direction to expand the space in which reverse Turing tests operate is to consider how the test is solved in addition to the underlying difficulty of the problem. For example, our experiment with Mechanical Turk shows that human solving time is very predictable for random strings, as visible in figure 18. We observe that the brain has a "start-up" time of *2.6s* during which scientists believe the brain performs something akin to Gabor filters [18] to preprocess shapes. Then the brain takes *0.97s* to process each character.

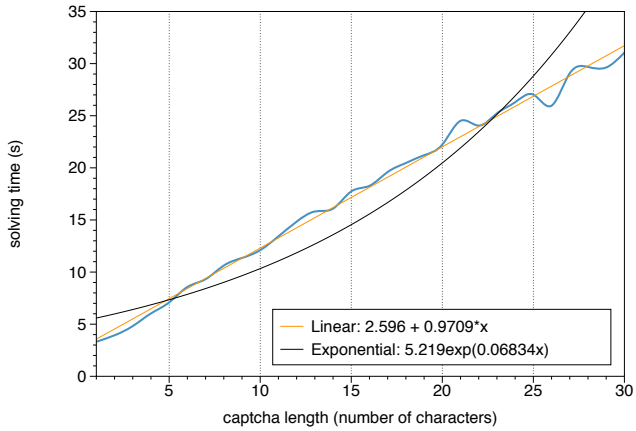

Figure 18: Human solving time increases linearly with the length of the captcha

This linear relation between the length of a captcha and the time it takes to process is quite apparent in figure 18. Clearly solving time is not a good reverse Turing test since it is easy for machines to fake, but other examples might exist, especially for captcha schemes that require a prolonged user engagement.

Leveraging reputation. In addition to considering how a reverse Turing test is solved, captcha providers could consider the identity of the solver, for example the IP address, the geographic location, etc. If a good enough proof of identity can be established, providers can use this reputation to adapt the difficulty of the reverse Turing test. This opens the door to easier tests for users in good standing, which will alleviate the captcha burden. It also empowers providers to challenge suspicious solvers more aggressively, and serves as a signal that is entirely orthogonal to the difficulty of the reverse Turing test, which will in turn result in a higher quality anti-abuse system. On the other hand, employing such a risk-based strategy requires a great deal of knowledge about network behavior and a large user base, which makes it feasible only for large providers.

10 Related Work

Alternate captcha schemes In [20] the authors proposed the Asirra captcha scheme, which was broken within a year [22]. In [27] the authors proposed using a 3D model as captcha. In [5] the authors proposed animated captchas. In [36] the authors present the concept of emergent images, and propose to use animated emergent images as captcha. In [24] the authors proposed using the task of rotating images as a captcha.

NuCaptcha was one of the first to deploy video captchas [30], and was first broken first by [7] with motion tracking and shortly after by a second group using a variant of this technique [45].

Captcha solving In [13] the authors propose using machine learning classifiers to attack captchas. In [12] the same authors study how efficient statistical classifiers are at recognizing captcha letters. Almost a decade ago, the authors of [13] mentioned in the discussion section that unifying the segmentation and recognition steps was a promising direction. However until this work this direction was not explored and we are not aware of any prior work that successfully applied this idea to captcha solving. In [47] the authors were able to solve the 2008 Microsoft captcha using the segment then recognize approach. In [44] the author proposes using an erode and dilate filter to segment captchas. [46] is one of the first papers to propose the use of histogram-based segmentation techniques against captchas. Yan et al. where able to devise heuristics to segment the easy version of reCaptcha 2010 [4]. In [6] the authors were able to break an older version of reCaptcha that used English words with a dictionary attack. In [8], the authors successfully applied the segment then recognize approach to audio captcha schemes. In 2011 [10] the authors successfully attacked numerous captchas schemes using an improved version of the segment then recognize approach, but failed to break reCaptcha. In 2012, the authors of [15] used a complex image preprocessing phase that relies on character alignment, morphological segmentation with three-color bar character encoding and heuristic recognition to crack the reCaptcha 2011. In 2013 the authors of [21] looked at hollow captchas specifically.

Machine learning algorithms The perceptron, the simplest neural network, has been used as a linear classifier since 1957 [41]. Convolutive neural networks, which are considered to be the most efficient neural networks to recognize letters, were introduced in [32]. Space displacement neural network that attempt to recognize digits without segmentation were introduced in [35]. Support vector machines were introduced in [14]. The KNN algorithm is described in detail in [16]. Recently deep belief networks, which aim at mimicking the human brain, have emerged has the best way to classify complex data such as images [25, 29]. While still in their infancy, deep belief networks represent a major break thought that yield unparalleled accuracy. The use of a bag of features to recognize objects in images is a very active field. The closest work to ours in this area is by [33], where the authors try to segment and categorize objects using this approach.

11 Conclusions

This paper introduces a novel approach to solving captchas in a single step that uses machine learning to attack the segmentation and the recognition problems simultaneously. Performing both operations jointly allows our algorithm to exploit information and context that is not available when they are done sequentially. At the same time, we remove the need for any hand-crafted component, making our approach generalize to new captcha schemes where the previous approach can not.

We were to solve many prominent real-world captcha schemes that use both negative kerning and occluding lines without any modification to the algorithm. Improving on the best previous results, our algorithm was able to achieve a *38.68%* recognition rate on Baidu 2011, *55.22%* on Baidu 2013, *51.09%* on CNN, *51.39%* on eBay, *22.67%* on ReCaptcha 2011, *22.34%* on ReCaptcha 2013, *28.29%* on Wikipedia, and *5.33%* on Yahoo. The breadth of distortions our algorithm is able to solve shows that it is a general solution for automatically solving captchas. Based on our experience, we also provide suggestions on how reverse Turing tests might be improved going forward.

The effectiveness and universality of our results suggests that combining segmentation and recognition is the next evolution of catpcha solving, and that it supersedes the sequential approach used in earlier works. With these advances, it seems that purely text-based captchas are likely to have declining utility; significant effort may be needed to rethink the way we perform reverse Turing tests.

Acknowledgment

We thank Aleksandra Korolova, Matthieu Martin, Celine Fabry and our anonymous reviewers for their comments and suggestions. This work was partially supported by the National Science Founda- tion, the Air Force Office of Scientific Research, and the Office of Naval Research.

References

[1] Are you human ? http://areyouahuman.com/.

[2] Breaking the minteye image captcha in 23 lines of python. Blog post http://www.jwandrews.co.uk/2013/01/breaking-the-minteye-image-captcha-in-23-lines-of-python/.

[3] Minteye captcha. website: http://www.minteye.com/, 2013.

[4] A. S. E. Ahmad, J. Yan, and M. Tayara. The robustness of google captchas. Technical report, Newcastle University, 2011.

[5] E. Athanasopoulos and S. Antonatos. Enhanced captchas: Using animation to tell humans and computers apart. In *IFIP International Federation for Information Processing*, 2006.

[6] P. Baecher, N. Büscher, M. Fischlin, and B. Milde. Breaking recaptcha: A holistic approach via shape recognition. In *Future Challenges in Security and Privacy for Academia and Industry*, pages 56–67. Springer, 2011.

[7] E. Bursztein. How we broke the nucaptcha video scheme and what we propose to fix it. blog post http://elie.im/blog/security/how-we-broke-the-nucaptcha-video-scheme-and-what-we-propose-to-fix-it/, February 2012.

[8] E. Bursztein, R. Bauxis, H. Paskov, D. Perito, C. Fabry, and J. C. Mitchell. The failure of noise-based non-continuous audio captchas. In *Security and Privacy*, 2011.

[9] E. Bursztein and S. Bethard. Decaptcha: breaking 75% of eBay audio CAPTCHAs. In *Proceedings of the 3rd USENIX conference on Offensive technologies*, page 8. USENIX Association, 2009.

[10] E. Bursztein, M. Martin, and J. Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, CCS '11, pages 125–138, New York, NY, USA, 2011. ACM.

[11] E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, D. Jurafsky, and J. C. Mitchell. Easy does it: More usable captchas. *CHI*, 2014.

[12] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Computers beat humans at single character recognition in reading based human interaction proofs (hips). In *CEAS*, 2005.

[13] K. Chellapilla and P. Simard. Using machine learning to break visual human interaction proofs (HIPs). *Advances in Neural Information Processing Systems*, 17, 2004.

[14] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] C. Cruz-Perez, O. Starostenko, F. Uceda-Ponga, V. Alarcon-Aquino, and L. Reyes-Cabrera. Breaking recaptchas with unpredictable collapse: heuristic character segmentation and recognition. In *Pattern Recognition*, pages 155–165. Springer, 2012.

[16] B. Dasarathy. Nearest Neighbor ({NN}) Norms:{NN} Pattern Classification Techniques. 1991.

[17] R. Datta. Imagination: A robust image-based captcha generation system. In *ACM Multimedia Conf.*, 2005.

[18] J. G. Daugman et al. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Optical Society of America, Journal, A: Optics and Image Science*, 2(7):1160–1169, 1985.

[19] T. G. Dietterichl. Ensemble learning. *The handbook of brain theory and neural networks*, pages 405–408, 2002.

[20] J. Elson, J. Douceur, J. Howell, and J. Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In *4th ACM CCS*, 2007.

[21] H. Gao, W. Wang, J. Qi, X. Wang, X. Liu, and J. Yan. The robustness of hollow captchas. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1075–1086. ACM, 2013.

[22] P. Golle. Machine learning attacks against the asirra captcha. In *ACM CCS 2008*, 2008.

[23] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

[24] R. Gossweiler, M. Kamvar, and S. Baluja. What's up captcha? a captcha based on image orientation. In *World Wide Web*, 2009.

[25] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[26] R. Hof. Ai startup vicarious claims milestone in quest to build a brain: Cracking captcha. http://www.forbes.com/sites/roberthof/2013/10/28/ai-startup-vicarious-claims-milestone-in-quest-to-build-a-brain-craking-captcha/, November 2013.

[27] M. Hoque, D. Russomanno, and M. Yeasin. 2d captchas from 3d models. In *IEEE SoutheastCon 2006*, 2006.

[28] A. Kittur, E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 453–456, New York, NY, USA, 2008. ACM.

[29] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *ICML*, 2011.

[30] Leapmarketing. Video-based captchas now available for sites and blogs. http://www.prnewswire.com/news-releases/video-based-captchas-now-available-for-sites-and-blogs-97471319.html, 2008.

[31] Y. Lecun. The mnist database of handwritten digits algorithm results. http://yann.lecun.com/exdb/mnist/.

[32] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[33] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1):259–289, 2008.

[34] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[35] O. Matan, C. Burges, and J. Denker. Multi-digit recognition using a space displacement neural network. *Advances in Neural Information Processing Systems*, pages 488–488, 1993.

[36] N. J. Mitra, H.-K. Chu, T.-Y. Lee, L. Wolf, H. Yeshurun, and D. Cohen-Or. Emerging images. *ACM Transactions on Graphics*, 28(5), 2009. to appear.

[37] G. Mori and J. Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *CVPR 2003*, pages 134–144, 2003.

[38] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. Voelker, and S. Savage. Re: CAPTCHAs–Understanding CAPTCHA-solving services in an economic context. In *Proceedings of the 19th USENIX conference on Security*, pages 28–28. USENIX Association, 2010.

[39] M. Naor. Verification of a human in the loop or identification via the turing test. Available electronically: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps, 1997.

[40] L. Rokach. *Pattern classification using ensemble methods*, volume 75. World Scientific, 2009.

[41] F. Rosenblatt. The perceptron: a perceiving and recognizing automation (projet PARA), Cornell Aeronautical Laboratory Report. 1957.

[42] Spamtech. Cracking the areyouahuman captcha. http://spamtech.co.uk/software/bots/cracking-the-areyouhuman-captcha/, 2012.

[43] J. Tam, J. Simsa, S. Hyde, and L. von Ahn. Breaking audio captchas. In *Advances in Neural Information Processing Systems*, 2008.

[44] J. Wilkins. Strong captcha guidelines v1. 2. *Retrieved Nov*, 10:2010, 2009.

[45] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. van Oorschot. Security and usability challenges of moving-object captchas: Decoding codewords in motion. In *Usenix Security*, 2012.

[46] J. Yan and A. Ahmad. Breaking visual captchas with naive pattern recognition algorithms. In *ACSAC 2007*, 2007.

[47] J. Yan and A. El Ahmad. A Low-cost Attack on a Microsoft CAPTCHA. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 543–554. ACM, 2008.