Breaking Text-based CAPTCHA Using Various Machine Learning Methods

Martin Roy E. Nabus
martin_roy.nabus@up.edu.ph

Eduardo F. Valdez
eduardofvaldez1@gmail.com

***Abstract*—The abstract goes here.**

1. Introduction

CAPTCHA, an acronym for Completely Automated Public Turing test to tell Computers and Humans Apart [1], is a reverse Turing test (i.e. Turing test administered by a computer program) that is intended to be easily solvable by humans and difficult for computers. CAPTCHAs are used to protect sites from bots and botnets that perform malicious attacks such as sending spam, stuffing online polls, causing distributed denial of service (DDoS), and/or committing online fraud.

One of the earliest descriptions of a process for generating an image to be used for the test was done by Lillibridge et al [2], which involved the generation of a random string and the randomization of its appearance to obfuscate the string; this was implemented in 1997 on the AltaVista web search engine. A commercial implementation was then introduced by David Gausebeck and Max Levchin for identifying fake accounts on PayPal [3].

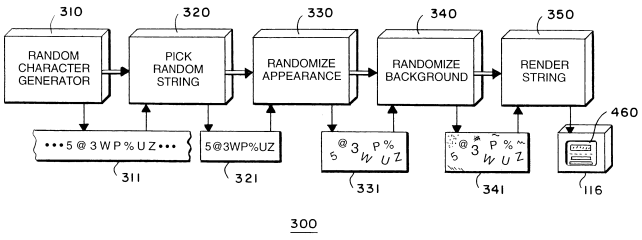


Figure 1: Text-based CAPTCHA generation as described by Lillibridge et al

Nowadays, there are several variations of CAPTCHAs apart from the original text-based CAPTCHA [4]. Examples of such tests are audio-based tests, image-based tests such as identifying images that satisfy certain conditions, and video-based tests such as recognizing objects and/or text from a video. Regardless, text-based CAPTCHAs are still used because they are easily recognizable by humans, they are easy to generate, and because image and video-based CAPTCHAs require large databases of images/videos because reusing the same images/videos may make the system prone to hacking.

Solving a text-based CAPTCHA can be broken down into the following steps [5]:

- The **preprocessing** step involves the removal of backgrounds, lines, and other types of noise that may affect the solving of the CAPTCHA.
- The **segmentation** step is the separation of the text image into individual character images.
- The **post-processing** step is an optional step that cleans up or improves the segmentation output. An example of a process done in this step is the normalization of the sizes of output character images.
- The **recognition** step is the identification of the characters corresponding to the character images.

Since computers can beat humans in recognizing single characters including distortions [6], it is important for a text-based CAPTCHA image to be hard to segment into individual characters. CAPTCHA generators do this by applying anti-segmentation techniques [5] such as applying complex backgrounds, occluding lines, and negative kerning or reducing the space between letters. Anti-recognition techniques such as using multiple fonts, different character sets, variable font sizes, character distortion or blurring, character tilting, and word waving are also applied to the text image to make computer recognition even harder.

[INSERT RELATED WORK HERE]

This project aims to study the performance of various machine learning classifiers on the solving of text-based CAPTCHAs. In particular, it aims to identify how many CAPTCHAs can be solved depending on the machine learning algorithm, the per-letter accuracy of each algorithm, and the time it takes for each algorithm to answer a CAPTCHA.

2. Review of Related Literature

2.1. Morphology

Digital image processing is the application of algorithms to an image represented as a set of pixels mapped to a two-dimensional area for purposes such as classifying the

image or extracting important features from it. Examples of operations used to process images are **morphological operations**. Mathematical morphology [7] is the processing of geometric structures using concepts from set theory. In image processing, it is the combination of an image and a structuring element using set operations such as union and intersection to generate a new image. Since morphological operators are extensively used in binary (i.e. black and white) images, the first step of morphology in image processing is usually the conversion of a colored or a grayscale image to a binary image; this is called thresholding. A thresholding operation is defined as

$$\theta(f,t) = \begin{cases} 1 & f \geq t \\ 0 & \text{otherwise} \end{cases}$$

where f is the function value resulting from some sort of thresholding method, and 1 & 0 are foreground and background values, respectively. Thresholding methods can be based on histogram shape, gray-level clustering, entropy of foreground and background regions, similarity measures between attributes such as edges, and so on [8].

In digital image processing, the structuring element is a convolution matrix consisting of 1's and 0's, which is also called a kernel. The most basic morphological operations are erosion and dilation [9]. For both of these operations, the kernel is first superimposed on top of each pixel and its neighbors such that the center of the kernel is superimposed on top of the pixel itself. Depending on the operation, the new value of each pixel is as follows:

- For **erosion**, if at least one 1-value on the kernel is superimposed on top of a 0-valued image pixel, the new pixel value is 0. Otherwise, it is 1.
- For **dilation**, if at least one 1-value on the kernel is superimposed on top of a 1-valued image pixel, the new pixel value is 1. Otherwise, it is 0.

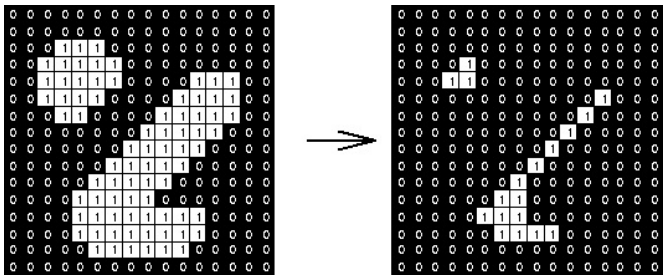


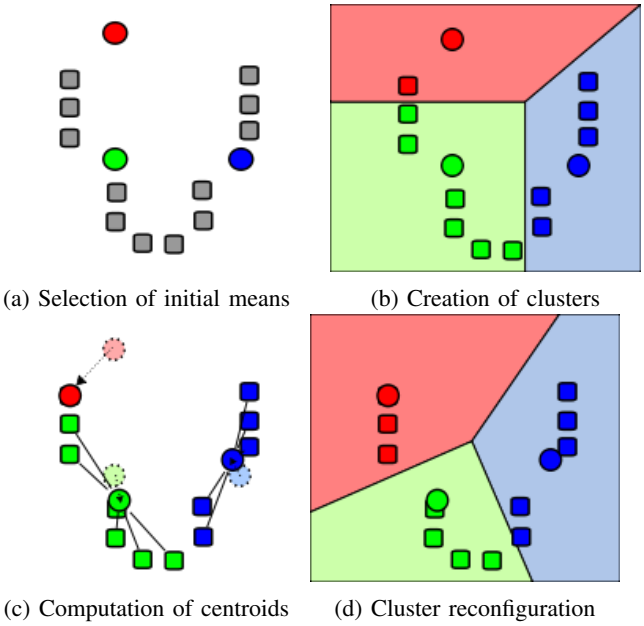
Figure 2: Example of an erosion operation

Kernels can be of different shapes; for example, a square kernel is one whose elements are all 1's while a cross kernel is one whose 1's are on the middle row and middle column only. Figure 2 shows the result of the erosion of an image using a 3×3 square kernel. In general, erosion is used to shrink foreground regions and enlarge holes, while dilation is used to enlarge foreground regions and shrink holes. Other operations include opening and closing, top-hat and black-hat transforms, and the morphological gradient.

2.2. K-means clustering

Clustering is the grouping of a set of objects so that the objects in a group are similar to one another and objects from different groups are different from one another [REF?]. From a mathematical perspective, the goal of clustering is to minimize the intra-cluster distance, a measure of dissimilarity between elements in a cluster, and to maximize the inter-cluster distance, a measure of dissimilarity between different clusters. Clusters can be formed using different types of algorithms; examples of these are hierarchical or connectivity-based algorithms such as SLINK [10], density-based algorithms such as DBSCAN [11], and centroid-based algorithms such as k-means [12].

The k-means clustering algorithm clusters a data set based on the number of clusters specified beforehand. Random points as many as the number of clusters are selected from either the data set or the data domain, and these points shall serve as "means" for the initial clustering. Each point is assigned to a cluster whose "mean" is closest to it. After all the points are assigned, the centroids (the geometric means of the cluster points) are computed and will then serve as means for the next clustering iteration. The assignment of points to clusters and selection of new means are done iteratively until convergence is reached, which is when the resulting clusters of an iteration are the same as the clusters in its previous iteration . An illustration of the algorithm is shown by Figure 3.



(a) Selection of initial means (b) Creation of clusters

(c) Computation of centroids (d) Cluster reconfiguration

Figure 3: Illustration of the K-means algorithm (by Weston Pace, distributed under a CC BY-SA 3.0 license)

2.3. Machine Learning and Classification

Machine learning is the science of giving computers the ability to progressively improve their performance on processing data without being explicitly programmed [13]. It involves the construction of algorithms that can learn from data to make predictions on other data of the same nature. Machine learning has several applications, which include spam filtering [14], medical imaging [15], and intruder detection on computer networks [16]. In general, machine learning is used for tasks such as classification, regression, dimensionality reduction, density estimation, clustering, and so on.

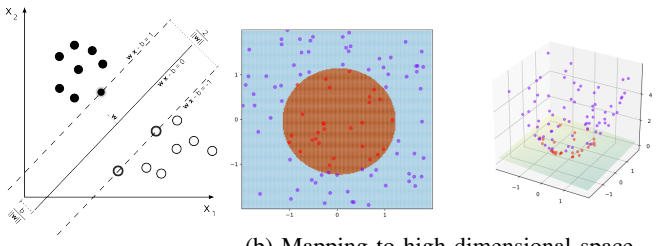
Classification is the identification of the class of a new observation based on its quantifiable properties or features and the information derived from a training set of data whose classes are known [REF?]. Algorithms that perform the mapping of data to classes are called classifiers. Classifiers have different approaches to classifying data, some of which are described below.

2.3.1. K-Nearest Neighbor. The k-nearest neighbor classifier [17] is an instance-based learning algorithm that assigns a class to an observation based on the classes of the training data that are closest to it with respect to some similarity measure. Suppose there exist an observation X_t with an unknown class and pairs $(X_1, Y_1), (X_2, Y_2), ..., (X_n, Y_n)$ where X_i and Y_i are the i$^{\text{th}}$ training datum and its corresponding label, respectively. Let $(X_{(1)}, Y_{(1)}), (X_{(2)}, Y_{(2)}), ..., (X_{(n)}, Y_{(n)})$ be a sequence of order statistics such that $||(X_{(1)} - X_t)|| \leq ||(X_{(2)} - X_t)|| \leq ... \leq ||(X_{(n)} - X_t)||$. The class of X_t is defined as

$$Y_t = \text{mode}(Y_{(1)}, Y_{(2)}, ..., Y_{(k)})$$

where $Y_{(i)}$ is the class of the order statistic $X_{(i)}$ and k is a user-defined, preferably odd positive integer that indicates the number of nearest neighbors to be considered for classification. In case of ties in classification, k is decremented by 1 until the tie is resolved. For example, if applying the k-NN algorithm on an observation with $k = 5$ results into labels $(1, 2, 1, 2, 4)$, k will be reduced to 3 to break the tie and the observation will be given the label 1.

2.3.2. Support Vector Machines. A support vector machine (SVM) [18] is a linear classifier that identifies how to separates two sets of data with the highest margin of separation. More formally, given training data $(x_1, y_1), ..., (x_n, y_n)$ where y_i is the corresponding class label (1 or -1) of the data point x_1, an SVM solves for a hyperplane that maximizes the distance between that hyperplane and the nearest points from both classes. The hyperplane is of the form $w \cdot x - b = 0$, where w is the normal vector to the hyperplane. The distance to be maximized is the distance between two hyperplanes $w \cdot x - b = 1$ and $w \cdot x - b = -1$ such that data from class 1 is on or above the first hyperplane and data from class -1 is on or below the second hyperplane; that distance is twice the norm of the normal vector w. The data points on the parallel hyperplanes are the support vectors for those hyperplanes. Upon identifying the optimal separating hyperplane and its corresponding support vectors, the label $\hat{y}$ for an unclassified data point $\hat{x}$ can be solved by $\hat{y} = \text{sgn}(w \cdot \hat{x} - b)$.



(a) Linear SVM

(b) Mapping to high-dimensional space

Figure 4: Visualization of an SVM classifier

To make the SVM classifier nonlinear, a method known as the kernel trick [19] is used. This trick involves the mapping of data to a higher-dimensional feature space before finding the separating hyperplane. An example of such mapping is shown in Figure 4b, wherein each data point (a, b) is mapped to a 3-dimensional space using the mapping function $\varphi((a, b)) = (a, b, a^2 + b^2)$; it can be seen that the hyperplane is easier to identify in the new 3D space. The label for $\hat{x}$ is then solved by $\hat{y} = \text{sgn}(w \cdot \varphi(\hat{x}) - b)$.

Computation of the mappings of the points can be avoided altogether by using a function called a kernel. The kernel $K(x, \hat{x})$ returns a similarity measure between support vector x and unclassified data point $\hat{x}$ with respect to a higher-dimensional space, which is then used to compute for the label of $\hat{x}$:

$$\hat{y} = \text{sgn}\left(\sum_{i=1}^{n} w_i y_i K(x_i, \hat{x})\right)$$

where n is the number of support vectors. Examples of commonly used kernel functions are the polynomial kernel $K(x, \hat{x}) = (x \cdot \hat{x})^d$ and the radial basis function (RBF) or Gaussian kernel $K(x, \hat{x}) = \exp(-\gamma||x - \hat{x}||^2)$ [20] where $\gamma = \frac{1}{2\sigma^2}$.

2.3.3. Random Forest. A random forest classifier [21] is an ensemble learning algorithm that constructs multiple decision trees using a randomly selected subset from the training data for each decision tree and identifies the class label of an input based on the output of the majority of the decision trees. This classifier is known to be highly resistant to overfitting due to its attempt to reduce the sensitivity of the mode to noise by constructing uncorrelated decision trees. Given training data and their corresponding class labels, a random forest classifier first performs what is called bootstrap aggregation or bagging; this step selects random subsets from the training data is uniformly selected with replacement. During bagging, the features of the data can also be "bagged" (i.e. different features can be selected for each sampling) to further decorrelate the trees. For each random subset, a decision tree is constructed. To identify

the class label of an unclassified data point, the classifier returns the mode of the output labels of all decision trees.

Decision trees are individually constructed using a top-down splitting of the sampled training data based on some feature that yields the highest information gain. Information gain is defined as the difference between the uncertainty value in the parent decision node and the weighted sum of uncertainty values in the child decision nodes. Examples of metrics that compute the uncertainty values of each node are Gini impurity and information entropy [22].

2.3.4. Deep Learning.

3. Methodology

3.1. Procedure for the Machine Learning Algorithms

3.1.1. Preprocessing. Each image is converted into grayscale, then into binary using a thresholding method called the Otsu's method [23], which searches for the threshold value t that minimizes the intra-class variance within both classes. The overall intra-class variance for the grayscale intensities on each image is defined as

$$\sigma_w^2(t) = \sigma_0^2(t) \sum_{i=0}^{t-1} p_i + \sigma_1^2(t) \sum_{i=t}^{255} p_i$$

where $\sigma_0^2(t)$ and $\sigma_1^2(t)$ are the intra-class variances of the class 0 (black) and class (1) pixels, respectively, and the summations are the corresponding probabilities of a pixel to have an intensity within the range dictated by the limits of the summations. According to Otsu, minimizing the overall intra-class variance also minimizes the inter-class variance.

3.1.2. Segmentation and Post-processing.

3.1.3. Recognition.

3.2. Procedure for the Deep Learning Algorithm

4. Results and Discussion

4.1. Experiment specifications

4.2. Experiment results

5. Conclusion

The conclusion goes here.

References

[1] L. von Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Commun. ACM*, vol. 47, no. 2, pp. 56–60, Feb. 2004. [Online]. Available: http://doi.acm.org/10.1145/966389.966390

[2] M. Lillibridge, M. Abadi, K. Bharat, and A. Broder, "Method for selectively restricting access to computer systems," United States of America Patent US6 195 698B1, February 27, 2001.

[3] PayPal, "Max Levchin: Online fraud-buster," October 2002. [Online]. Available: https://www.bloomberg.com/news/articles/2002-09-30/max-levchin-online-fraud-buster

[4] A. B. Jeng, C.-C. Tseng, D.-F. Tseng, and J.-C. Wang, "A study of captcha and its application to user authentication," in *Computational Collective Intelligence. Technologies and Applications*, J.-S. Pan, S.-M. Chen, and N. T. Nguyen, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 433–440.

[5] 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*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 125–138. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046724

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

[7] J. Serra, *Image Analysis and Mathematical Morphology*. Orlando, FL, USA: Academic Press, Inc., 1983.

[8] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation," vol. 13, pp. 146–168, 01 2004.

[9] R. Fisher, S. Perkins, A. Walker, and E. Wolfart. (2003) Morphology. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm

[10] R. Sibson, "SLINK: An optimally efficient algorithm for the single-link cluster method," *The Computer Journal*, vol. 16, pp. 30–34, 1973.

[11] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[12] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theor.*, vol. 28, no. 2, pp. 129–137, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1982.1056489

[13] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Dev.*, vol. 3, no. 3, pp. 210–229, Jul. 1959. [Online]. Available: http://dx.doi.org/10.1147/rd.33.0210

[14] T. S. Guzella and W. M. Caminhas, "Review: A review of machine learning approaches to spam filtering," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10 206–10 222, Sep. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2009.02.037

[15] M. N. Wernick, Y. Yang, J. G. Brankov, G. Yourganov, and S. C. Strother, "Machine learning in medical imaging," *IEEE Signal Processing Magazine*, vol. 27, no. 4, pp. 25–38, July 2010.

[16] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 305–316.

[17] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: https://doi.org/10.1023/A:1022627411411

[19] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, ser. COLT '92. New York, NY, USA: ACM, 1992, pp. 144–152. [Online]. Available: http://doi.acm.org/10.1145/130385.130401

[20] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm," *J. Mach. Learn. Res.*, vol. 11, pp. 1471–1490, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1859899

[21] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[22] ——, *Classification and regression trees*. Routledge, 2017.

[23] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.