Computers & Security 88 (2020) 101635

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

Is image-based CAPTCHA secure against attacks based on machine learning? An experimental study

Fatmah H. Alqahtani, Fawaz A. Alsulaiman*

King Saud University, College of Computer and Information Sciences, Department of Computer Science, Riyadh, Saudi Arabia

ARTICLE INFO

Article history:
Received 23 November 2018
Revised 6 September 2019
Accepted 5 October 2019
Available online 8 October 2019

Keywords:
CAPTCHAs
Deep learning
Machine learning
Security
User authentication

ABSTRACT

The *completely automated public Turing test to tell computers and humans apart* (CAPTCHA) is among the most common methods of authentication used by websites and web services. It is intended to protect online services from automated scripts and malicious programs. Text-based and audio CAPTCHA are two of the earliest such methods, and have been shown to be inadequate at protecting systems and services. Image-based CAPTCHA has been introduced to address the limitations of previous CAPTCHA methods. It uses image recognition tasks to determine whether the user is a human or a malicious program. In light of the sensitivity of protected resources, challenges to their security arising from advances in machine learning algorithms are investigated here. This study examines the strength of image-based CAPTCHA by proposing an image-based CAPTCHA breaking system. The proposed system can automatically answer challenges posed by the recently proposed Google image reCAPTCHA with minimal human intervention. It employs deep learning technologies and machine learning algorithms, including random forest, classification and regression trees (CART), bagging with CART, and Naïve Bayes to automatically answer challenges. The proposed attack mechanism achieved an average accuracy of 85.32% while successfully solving 56.29% of reCAPTCHA challenges posed to it. The results show current image-based CAPTCHAs to deter automated scripts and malicious programs provide a false sense of security.

1. Introduction

Since the early days of the Internet, different services have been provided to serve a variety of users. Service providers aim to allocate these services and resources for the benefit of target users (Vikram et al., 2011). They soon encountered the problem of automated scripts (BOTs) and malicious programs that were accessing these services and distributing malicious information, wasting resources, and degrading system performance (Ponec, 2006). One user authentication approach to prevent automated scripts is the completely automated public Turing test to tell computers and humans apart (CAPTCHA) (Fidas et al., 2011). CAPTCHAs have been introduced to prevent automated scripts from launching attacks by introducing a simple test that is easy to solve by a human user but poses a challenge to automated scripts (Yan and El Ahmad, 2011). Therefore, the security of CAPTCHAs has to be guaranteed to ensure the protection of systems and services.

Two early CAPTCHA methods are text-based and audio CAPTCHA. They present a user with distorted text and audio, respectively, and the user must be able to recognize the actual words. Optical character recognition (OCR) and automatic speech recognition (ASR) have since managed to breach the security of these CAPTCHAs (Snyder et al., 2010). In text-based CAPTCHAs, the image containing the letters or digits is captured and OCR software is used to recognize them. ASR breaks audio CAPTCHAs by extracting voice features and recognizing letters or digits accordingly (Shirali-Shahreza and Shirali-Shahreza, 2008; Yan and Yu, 2009).

Image-based CAPTCHAs are among the most effective solution to the limitations faced by aforementioned security methods (Yamamoto et al., 2010). It presents image-based cognitive challenges to determine whether a user is human or a malicious program. These challenges rely on the fact that recognizing objects and describing images are natural cognitive processes in humans but too complex for computers (Karpathy and Fei-Fei, 2015). Because of the importance of image-based CAPTCHAs in securing websites and services, challenges to their security posed by advances in machine learning algorithms need to be further investigated.

This paper investigates the strength of image-based CAPTCHA by proposing an automated breaking system. The breaking system recognizes images displayed by CAPTCHA challenges and makes an

* Corresponding author.
E-mail addresses: 436203639@student.ksu.edu.sa (F.H. Alqahtani), falsulaiman@ksu.edu.sa (F.A. Alsulaiman).

https://doi.org/10.1016/j.cose.2019.101635
0167-4048/© 2019 Elsevier Ltd. All rights reserved.

automatic, knowledgeable attempt via machine learning algorithms to choose the appropriate answers.

The remainder of this paper is structured as follows: Section 2 presents some general background information about CAPTCHAs and semantic similarity. Section 3 reviews the literature on different types of CAPTCHAs as well as machine learning attacks on image-based CAPTCHAs. Section 4 explains the methodology used to attack the selected image-based CAPTCHA, and Section 5 details an experiment on a dataset, the results obtained, and the metrics used to assess the results. Section 6 contains the conclusions of this study and directions for future research.

2. Background

2.1. CAPTCHAs

In the mid-20th century, the British mathematician Alan Turing formulated a test, the Turing test, to identify situations in which machine-generated and human-generated responses are indistinguishable. To secure websites against automated attacks, the opposite of the Turing test is needed. A test that aims to distinguish a machine's response from that of a human is called a reverse Turing test (commonly known as a CAPTCHA test) (Snyder et al., 2010). This test presents a user with a challenge that is intended to be easy to solve for humans but difficult for machines (Pope and Kaur, 2005; Snyder et al., 2010).

CAPTCHAs have been incorporated extensively into online systems. Various free e-mail providers integrate CAPTCHAs into their systems to block spammers from creating accounts by employing automated scripts. Websites administrators use CAPTCHAs to prohibit web crawlers and bots from taking part in online polls. They are also used to prevent the indexing of websites for search engines using web spiders to avoid system overload and protect private information. They also help avoid dictionary attacks that attempt random passwords on login forms, and protect search engine databases and their sensitive data from being exploited (Pope and Kaur, 2005; Shirali-Shahreza and Shirali-Shahreza, 2008).

There are different categories of CAPTCHAs: text based, audio, video, and image based (Chandavale and Sapkal, 2015). Text-based CAPTCHAs are easy to design and implement, and are based on distorted words chosen randomly from a dictionary. Users are required to recognize and submit these words (Brodic et al., 2016; Bursztein et al., 2011; Chandavale and Sapkal, 2015). A variety of distortion techniques can be applied to these words, including grids and the generation of background noise (Moy et al., 2004).

Audio CAPTCHAs sample the pronunciation of letters or digits over a noisy background and randomly spaced intervals of speech. Users are required to submit the decoded sound content (Zhang, 2010). These sound-based systems are often used to complement other types of CAPTCHAs to help visually impaired people (Chandavale and Sapkal, 2015; Shirali-Shahreza and Shirali-Shahreza, 2008). Video CAPTCHAs represent a more recent method in which a video is played and users are then asked questions about its content (Brodic et al., 2016; Chandavale and Sapkal, 2015). Another form of CAPTCHAs is image-based CAPTCHA, where the user is presented with a question and required to select a corresponding image or a set of images from a list of choices (Brodic et al., 2016; Shirali-Shahreza and Shirali-Shahreza, 2008).

2.2. Machine learning

Machine learning (ML) is the field of computer science devoted to enabling computers to learn from their environment without being explicitly programmed (Wikipedia, 2017b). Enabling machines to learn automatically has opened the door to a wide range of successful applications, including detecting spam documents, tagging parts of speech, recognizing images, recommending systems, and playing games (Mohri et al., 2012).

In machine learning, three key elements are needed to enable the automation of learning (Paluszek and Thomas, 2017). First, datasets are essential to train and test a system. Second, models are used in learning systems to provide mathematical frameworks for learning. Finally, sufficient training is required for systems that map an input to an output to obtain the desired system goals. Training is carried out by allowing the system to learn from data.

Several learning approaches can be used, such as supervised, unsupervised, semi-supervised, and reinforcement learning. In supervised learning, the training data are associated with a target value, and the goal of the algorithm is to explore a way to estimate this value (Paluszek and Thomas, 2017). This learning approach is commonly associated with classification, regression, and ranking problems (Mohri et al., 2012).

On the contrary, unsupervised learning algorithms use unlabeled data (data instances without target values) to discover hidden patterns (Cady, 2017). In this learning approach, a quantitative evaluation of the learning algorithm's performance is challenging. Clustering and dimensionality reduction are examples of this learning approach (Mohri et al., 2012).

In semi-supervised learning, a portion of the training data is associated with labels while the rest are not. This can substantially help reduce the cost of labeling the data, which requires a skilled human (Mohri et al., 2012). Some problems can be framed as those of semi-supervised learning, such as classification, regression, and ranking (Paluszek and Thomas, 2017).

Active learning is considered a special type of semi-supervised learning. It aims to interactively make queries for the user to obtain labels for new data (Cady, 2017; Mohri et al., 2012). It targets achieving results comparable to those of supervised learning with a minimal amount of labeled data. This approach can be applied when labels are too expensive to acquire (Paluszek and Thomas, 2017).

2.2.1. Naïve Bayes

Naïve Bayes (NB) is a probabilistic supervised learning algorithm intended mainly for classification tasks. It is based on Bayes' theorem, which states that given a data tuple (or evidence) X and a hypothesis H that the data tuple X belongs to a specified class C, the posterior probability that the hypothesis H holds given the data tuple X is (Friedman et al., 1997):

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)}$$

where $P(X|H)$ is the posterior probability of X conditioned on H, $P(H)$ is the prior probability of H and $P(X)$ is the prior probability of X. The NB classifier utilizes Bayes' theorem, with m classes $C_1, C_2, \ldots, C_m$ and training set tuples, each of which consists of an n-dimensional attribute vector associated with the class label $X = \{x_1, x_2, \ldots, x_n\}$. NB classifies the tuple X to the class with the highest posterior probability conditioned on X, that is: $P(C_i|x) > P(C_j|X)$ for $1 \le j \le m,\ j \ne i$. Therefore, the probability that $p(C_i|x)$ is maximized and class c_i that gain the maximum $P(C_i|x)$ named as maximum posteriori hypothesis (Han et al., 2011). According to Bayes' theorem:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

NB simplifies the computational complexity by eliminating the prior probability $P(X)$ that is constant for all classes, and making a strong assumption that the attributes are conditionally independent given the class (Sammut and Webb, 2011). This assumption significantly reduces the computational cost, where only the class distribution is counted (Friedman et al., 1997; Han et al., 2011). Thus: $P(C_i|X) = P(X|C_i)P(C_i)$ where $P(X|C_i) = \prod_{k=1}^{n} P(x_k|c_i)$.Hence, to predict the class label of the instance X, for each class C_i, $P(X|C_i)P(C_i)$ will be evaluated and the predicted class label is the class that has the maximum $P(X|C_i)P(C_i)$ value (Han et al., 2011).

2.2.2. Classification and regression trees

Classification and regression tree (CART) is a binary decision tree that splits a decision tree into two sub-branches, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. It utilizes the Gini index as impurity measure for selecting an attribute for branching (Han et al., 2011; Tian et al., 2017). Node impurity using the Gini index increases when all classes are equally mixed together, and is the smallest when the node contains only one class. The termination process of tree growth is determined by a heuristic rule: When a node is reached where no significant decrease in impurity is possible, it becomes a terminal node. To classify an instance, its features are tested against the decision tree. A path is traced from the root to a leaf node, and holds the class prediction for that instance (Breiman, 2017). CART has been used in a diversity of fields. It has been used to develop intelligent medical decision support systems for diseases classification (Sathyadevi, 2011), in e-learning systems to predict the marks scored by university students in final exams of courses (Romero et al., 2013), and in environmental and ecological applications to construct discrimination rules that sample the spatial data over the studied area (Bel et al., 2009).

2.2.3. Bagging

Bootstrap aggregating (bagging) is an ensemble learning method that generates multiple bootstrap training sets from an original training set (by using sampling with replacement) and uses each of them to generate a classifier for inclusion in the ensemble (Wang, 2008). The steps followed in the bagging algorithm are shown in Pseudocode 1. In each iteration i, the bootstrap training set D_i is formulated with d instances selected from the original set D with replacement. Therefore, some of the instances in the original training set are not included in the bootstrap training set, and others may be chosen one or several times. Based on the formulated bootstrap training set, a classifier model M_i is generated. To classify a testing instance, X, each generated mode M_i returns its class prediction, and the bagged classifier M_* assigns the class with the most votes to X (Han et al., 2011).

Pseudocode 1 Bagging algorithm (Han et al., 2011).

```
1:  Input:
2:  D, a set of d training instances;
3:  k, the number of models in the ensemble;
4:  learning scheme (e.g.,CART)
5:
6:  Method:
7:  for  i = 1 to k do // create k models: do
8:      create bootstrap sample, $D_i$, by sampling $D$ with replacement;
9:      use $D_i$ to derive a model, $M_i$;
10:
11: Output: A composite model, $M_*$
```

2.2.4. Random forest

Random forest (RF) is a type of ensemble classifier proposed by Leo Breiman (Breiman, 2001) for classification and prediction tasks. This ensemble classifier involves a set of decision trees and outputs a class prediction based on the decision made by individual decision trees in the forest (Breiman, 2001). RF has two major ingredients that play a major role in the algorithm, bagging and CART-split criterion. Bagging (or bootstrap aggregation) is devised to create bootstrap samples from the original dataset and a classifier based on each bootstrap sample, where decision is taken by averaging the votes cast by individual trees. This technique leads to variance reduction and bias increase for procedures like trees. The CART-split criterion helps in deciding the best split at each node while building each tree in the forest based on the smallest value of Gini impurity (Biau and Scornet, 2016; Hastie et al., 2009).

The RF algorithm performs B iterations, for each iteration b a bootstrap sample of size N from the training set is first formulated. A random forest tree T is then generated for the bootstrapped sample by recursively choosing m random features from p features, where $m < p$, selecting the best splitting feature among the m features, and splitting the node into two daughter nodes until the minimum node size n_{min} is reached. Each tree is tested with out-of-bag (oob) samples that are not selected in the bootstrap samples. This process is repeated until B random forest trees have been created. To classify a new instance x, if the class prediction of the b_{th} random forest tree is $C_b(x)$, then the final class prediction is $C_{rf}^{B}(x) = majority\, vote\ C_b(x)_1^B$ (Hastie et al., 2009).

2.2.5. Convolutional neural networks

The machine learning methodology that uses neural networks (N) with a greater number of parameters and supplementary layers is known as deep learning (DL). A major advantage of DL over traditional ML is the automated feature extraction (Aghdam and Heravi, 2017). This type of learning requires much more computing power, and a complex architecture is established through the use of parallelism and graphics processing units (GPUs) (Patterson and Gibson, 2017).

Convolutional neural networks (CNNs) or (ConvNets) are among the most commonly used and powerful deep networks (Sermanet et al., 2012). They are designed to process grid-like data, such as 1D grid samples from time-series data or a 2D grid of image pixels. Therefore, they are well suited and have been applied to recognizing objects in images with great success (Gulli and Pal, 2017; LeCun et al., 2015). Neurons in CNNs assume that individual cortical neurons respond to stimuli. This response is restricted to a region of the visual field called the receptive field (Aghdam and Heravi, 2017). Thus, it is connected to a relatively small region of the previous layer and arranged in three dimensions-width, height, and depth-representing the number of filters used (Convolutional Neural Networks, 2017; Patterson and Gibson, 2017).

A typical CNN architecture is composed of an input, multiple hidden layers through which an input image passes, and an output layer. The input layer contains the row pixel values of the input image. The hidden layers can be either convolutional Rectified Linear Units (ReLU) or pooling, and the output layer is a fully connected layer (Patterson and Gibson, 2017). The convolutional layers involve a set of learnable filters that are small along the width and height but have the same depth. Each filter is convolved across the width and height of the input, computes the dot product of filter entries and the input at any position, and sums it and offsets the result by the bias. This layer generates an activation map (Gulli and Pal, 2017; Patterson and Gibson, 2017). Pooling layers are implanted between convolution layers. They reduce the number of parameters and thus the computation needed by down-sampling the special size and representation of the activation map produced by the convolutional layers. They aggregate values of the activation map by applying certain operations, such as max-pooling, that output the maximum value from the region specified by the filter (Duffner, 2009; Gulli and Pal, 2017).

Rectified linear units (ReLU) are layers that perform non-saturation as activation function, as $f(x) = \max(0, x)$. Different nonlinearity functions can be applied as well, for instance, the saturating hyperbolic tangent $f(x) = |\tanh(x)|$ and the sigmoid function $f(x) = (1 + e^{-x})^{-1}$. ReLU layers can effectively reduce the time needed to train the network but incur a significant penalty in terms of generalization accuracy (Vicente et al., 2017).

The input size characterized by high dimensionality, connecting neurons to all prior volume neurons is impractical when the spatial structure of the data needs to be considered. By enforcing the hyperparameter of the reception field, the spatially local correlation is exploited and, consequently, each neuron is connected to a small region of the input volume (Duffner, 2009).

The parameter-sharing scheme helps drastically reduce the number of required parameters by assuming that if some features are useful at a given spatial position, they are useful at another location as well. Therefore, neurons at each depth slice use the same weights and bias (Patterson and Gibson, 2017).

The last layer is the fully connected layer. Neurons of this layer are fully connected to all activations in previous layers, and their activations are computed via matrix multiplication followed by a bias offset. This reduces the full image into a single vector of class scores (Patterson and Gibson, 2017; Wikipedia, 2017a). Beyond feeding forward the network, backpropagation algorithm is employed to train it. During the training, each neuron in the volume calculates the gradient of its weights added up across every depth slice and updates a single set of weights per slice (Wikipedia, 2017a).

2.3. Semantic similarity

Semantic similarity (SS) measurement between a pair of units, such as concepts or words, is a core operation in many research fields, including Artificial Intelligence, natural language processing, knowledge management, information retrieval and mining, information extraction, word sense disambiguation, and biomedicine (Aouicha et al., 2016; Zhou et al., 2008). Humans can easily determine whether a word is more similar to a given word than another, and enabling computers to simulate human thinking in quantifying and measuring semantic similarly is an important area of research (Taieb et al., 2014).

To quantify the semantic, knowledge inside semantic resources is exploited. WordNet (Miller and Fellbaum, 1998; Miller, 1995) is one of the most popular knowledge resources used as an underlying reference ontology to measure the semantic similarity between words (Taieb et al., 2014). It is a lexical English language database that groups nouns, verbs, adjectives, and adverbs in synonym sets referred to as synsets, and offers different semantic relations between them. Gloss is defined for each synset to determine the concept it represents. Some relations among synsets in WordNet are “is-a-kind-of” relations (holonymy), such as hypernym and hyponym for nouns, and hypernym and troponym for verbs, and “is-a-part-of” relations (meronymy) for nouns hierarchies (Corley and Mihalcea, 2005; Wagh and Kolhe, 2012).

WordNet hierarchies render it well suited to test various similarity measures. Based on the parameter utilized to measure similarity, these proposed similarity measures can be classified as edge-based approaches, information content approaches, and the hybrid approach. Edge-based approaches measure similarity using the number of taxonomic links and the minimum path length between concepts, such as the methods proposed by Wu and Palmer (1994), Leacock and Chodorow (1998) and Li et al. (2003). Information content approaches formulate an information content (IC) function of concepts shared in common, where more general entities in a discourse form a smaller IC than more specialized ones. Examples are the methods proposed by Meng et al. (2012) and Sebti and Barfroush (2008). Hybrid approaches incorporate both the aforementioned approaches to measure semantic similarity (Taieb et al., 2014; Wagh and Kolhe, 2012).

3. Literature review

3.1. Types of CAPTCHAs

There are various types of CAPTCHAs that can be categorized as text-based, audio, video, and image-based CAPTCHAs (Chandavale and Sapkal, 2015).

EZ-GIMPY (Blum et al., 2000) is a text-based CAPTCHA with 850 English words in its lexicon that can be rendered in several fonts. The Gimp image processing tool is used to degrade the image. Another example of a text-based CAPTCHA is Baffle Text (Chew and Baird, 2003), which generates a character string that cannot be pronounced in English and then degrades the image via random masking. Pessimal print CAPTCHA (Kluever, 2008) generates an image by pseudo-randomly integrating a word, font, and set of image degradations. Due to advances in OCR and image-processing algorithms, text-based CAPTCHAs can no longer guarantee authentication (Shirali-Shahreza and Shirali-Shahreza, 2008; Yan and Yu, 2009).

Audio CAPTCHAs serve as an alternative for visually impaired people (Lazar et al., 2012; Zhang, 2010). CAPTCHA-instance (or C-instance) (Chan, 2003) is an audio CAPTCHA generated by overlapping utterances in the foreground (a digital sequence), with a background (the same voice speaking English words) produced using a text-to-speech synthesizer (TTS). Then, a classification and regression tree (CART) is used to discard C-instances that are easily solved by a machine. Many studies have explored the usability of this type of CAPTCHA (Fidas et al., 2011; Yan and El Ahmad, 2008), and have concluded that it is not suitable for visually or hearing-impaired people. Moreover, the principle of an attack on text-based CAPTCHAs can be applied to this method so that voice features are extracted and the letters or digits can be recognized (Bursztein et al., 2011; Saha et al., 2015).

A video CAPTCHA that is generated from YouTube videos containing labels (tags), specified by users who uploaded the videos, has been introduced (Kluever and Zanibbi, 2009). The user is asked to submit three tags describing the video, and the challenge is passed if any of the three submitted tags matches tags associated with the video. A CAPTCHA generation engine (Gargi, 2013) chooses a video clip from the database and separates it into several segments. A plurality of related queries is then associated with each segment. Based on this association, a video test is generated and stored in a video tests database. For each user, a CAPTCHA engine selects a video and maintains a user trial counter to determine whether the user is human. Video CAPTCHAs are a recently developed approach, but that they require sufficient bandwidth and variation in user perception are challenging issues (Saha et al., 2015).

Image-based CAPTCHAs are designed to overcome the limitations of text and audio CAPTCHAs (Shirali-Shahreza and Shirali-Shahreza, 2008). Image generation for Internet authentication (IMAGINATION) (Datta et al., 2005) comprises two distinct tests. The first is a click test, in which the user is asked to click the center of any sub-images of a composed image. The second is an annotation test. The selected sub-image is displayed after controlled composite distortions are applied, and the user is asked to select the word that represents the image from a list of possible candidates.

Mosaic-based human interactive proof (MOSAHIP) (Basso and Sicco, 2009) images are composed of smaller, real images that partially overlap with artificially created images. The user is asked to drag a movable text object and drop it in the area of the mosaic picture containing the image indicated in the question. The automated reverse Turing test using facial features (ARTIFACIAL) (Rui and Liu, 2004) automatically synthesizes an image with an embedded distorted face in a cluttered background. The user has to identify and click on the four corners of the eyes and the two corners of the mouth on the distorted face to pass the test.

Microsoft's Asirra (Elson et al., 2007) requires the user to select images of cats from a set of 12 images of cats and dogs. They formed the database through a partnership with Petfinder.com, the largest website for finding homes for homeless pets. Another image-based CAPTCHA proposed in Gao et al. (2010) presents the user with a jigsaw puzzle. The image is divided into $n \times n$ pieces with two pieces misplaced. The value of n can be either 3, 4, or 5 depending on the desired security level. The user is asked to swap the two misplaced pieces.

ESP-PIX (Lorenzi et al., 2012) challenges the user by presenting a set of six images. The user is required to choose a word from a list that describes all the images. SQUIGL-PIX (or SQ-PIX) (Lorenzi et al., 2012) displays the name of an object and a group of three images; the user must select the image containing the named object and trace the object on the image.

One of the most widely used CAPTCHA services is reCAPTCHA offered by Google (Von Ahn et al., 2008). reCAPTCHA has several different versions. Image reCAPTCHA is a new version based on the idea of recognizing images with similar content. Each challenge consists of a question containing a hint, a sample image and nine candidate images. The user is asked to identify images that are similar to the sample image.

3.2. Machine learning attacks on image-based CAPTCHAs

In the literature, several studies have been conducted to defeat different types of image-based CAPTCHAs.

The attack against IMAGINATION click test (Zhu et al., 2010) aims to locate the correct image among other composed images in the CAPTCHA by the following steps: Initially, all possible rectangular regions that represent the location of a candidate image are detected and ranked according to the probability of being a rectangular region. For each candidate rectangle, the two sides along the boundary are compared. The boundary is likely to be false if both sides are similar. By contrast, a boundary that has different textures on both sides is likely to be a true image boundary. This procedure is applied iteratively until an image location is arrived at with a certain confidence. The accuracy obtained was 74.31% over 2000 click test images.

Another attack performed on ARTiFACIAL (Li, 2015) was divided into two basic steps. First, a gradient face detector was learned by utilizing the boosting chain proposed in Xiao et al. (2003) to detect the face. Second, the intensity of the image was corrected to facilitate the detection of the facial components. To this end, facial component detectors trained with AdaBoost were used to detect the eyes, brows, nose, mouth, and the region between nose and mouth as an initial shape, and this was constructed by considering the constraints on the components. The discriminative search algorithm proposed in Liang et al. (2008) was then used to update the shape of the component constraints, and the result was evaluated via the boosted appearance model (BAM)(Liu, 2007). After several iterations, the best component shapes were generated as the final output. On 800 test images, the gradient-based face detector that trained over 10,000 images achieved an accuracy of 42.0% in identifying face locations.

Microsoft's ASIRRA (Lorenzi et al., 2012) was attacked using the following approach: The 12 CAPTCHA images were extracted, and a hierarchical temporal memory (HTM) network was constructed and used based on the collected dataset along with the extracted images for identification. Images of cats with the highest probabilities were then chosen and the remaining probabilities were used to make educated guesses about the remaining images. To set-up the experiments, 50, 100, 200, 400, 800, 1600, and 12,500 were images used for training and testing. The HTM network achieved the best performance when using 12,500 images (74.7% accuracy). The 100-, 400-and 1,600-image networks produced comparable performance at accuracies of 72.5%, 72.9% and 72.5%, respectively.

Another basic attack conducted by Lorenzi et al. (2012) was used to defeat the security of the SQ-PIX CAPTCHA. A keyword was fed into Bing's image search to collect images for an HTM network. The resulting HTM network was used to produce the likelihood of the challenge images, and the image with the highest probability was chosen for the next step. The Sobel edge detection algorithm was used to generate a mask by finding the borders of the object of interest based on the chosen image. This mask was used with a custom JavaScript to solve the challenge. Using 3300 images for training and testing, the HTM network achieved an accuracy of 83.9%.

ESP-PIX CAPTCHA was attacked by using the following steps (Lorenzi et al., 2012): A list of keywords was gathered and used in Bing's image search to collect images to train and test the HTM network. OCR tools and Levenshtein distance techniques were used to detect and compare textual data in images with the challenge list. Four images were entered into the HTM network that output the probabilities. The resulting probabilities and results of textual comparison were used to calculate the highest probability of the challenge response. When 7050 images were used to train and test the HTM network, an accuracy of 83.1% was obtained.

Lorenzi et al. (2012) also attacked ESP-PIX CAPTCHA using the following steps: The list of keywords was gathered and used in Bing image search to collect images to train and test the HTM network. The four challenge images were fed into an OCR tool to check if they contained any textual data. If textual data were detected, they were converted into string and compared with the challenge list. The matched words were stored and a temporary tag was associated with the image. If there was no exact match, Levenshtein distance was used to guess the keywords. The four images were entered into the HTM networks that output the probabilities. The resulting HTM network probabilities and results of textual comparison were used to calculate the highest probability of the challenge response. A total of 7050 images were used to train and test the HTM network, and yielded an accuracy of 83.1%.

Another CAPTCHA-breaking system (Sivakorn et al., 2016a) was designed to break Google's image reCAPTCHA by extracting semantic information from images to solve challenges. The content of an image was identified using the Clarifai image annotation service and libraries that assigned tags describing the challenge images. The returned tags associated with the hints were entered into the Word2Vec classifier (Mikolov et al., 2013), allowing the system to select images with content similar to that of the hint. Using a dataset of 700 challenges, and considering 10-fold cross-validation, the machine learning attack passed 44.71% of the challenges and achieved an average accuracy of 66.57%.

Table 1 provides an overview of the aforementioned attacks based on CAPTCHA, the attack methodology, the number of instances in the training and testing sets, and the resulting accuracy.

4. Attack methodology

The ability to recognize objects and describe images is a natural cognitive process in humans. Enabling computers to process images, recognize objects, and provide semantic information is a complicated process (Karpathy and Fei-Fei, 2015). Therefore, this concept is used in image-based CAPTCHAs to differentiate between humans and automated scripts. However, owing to recent improvements in deep learning and image annotation technologies that

Table 1
Summary of attacks on image-based CAPTCHAs.

DAE8FC	Attacked CAPTCHA	Attack methodology	Training set	Testing set	Accuracy
Zhu et al. (2010)	IMAGINATION	Image processing	–	2000	74.31%
Li (2015)	ARTiFACIAL	Computer vision	10,000	800	42.0% × 42.9%
Lorenzi et al. (2012)	Microsoft Asirra	Machine intelligence	12,500	12,500	74.7%
Lorenzi et al. (2012)	SQ-PIX	Machine intelligence	1650	1650	83.9%
Lorenzi et al. (2012)	ESP-PIX	Machine intelligence	3525	3525	83.1%
Sivakorn et al. (2016a)	Google image reCAPTCHA	Deep learning	630	70	66.57%

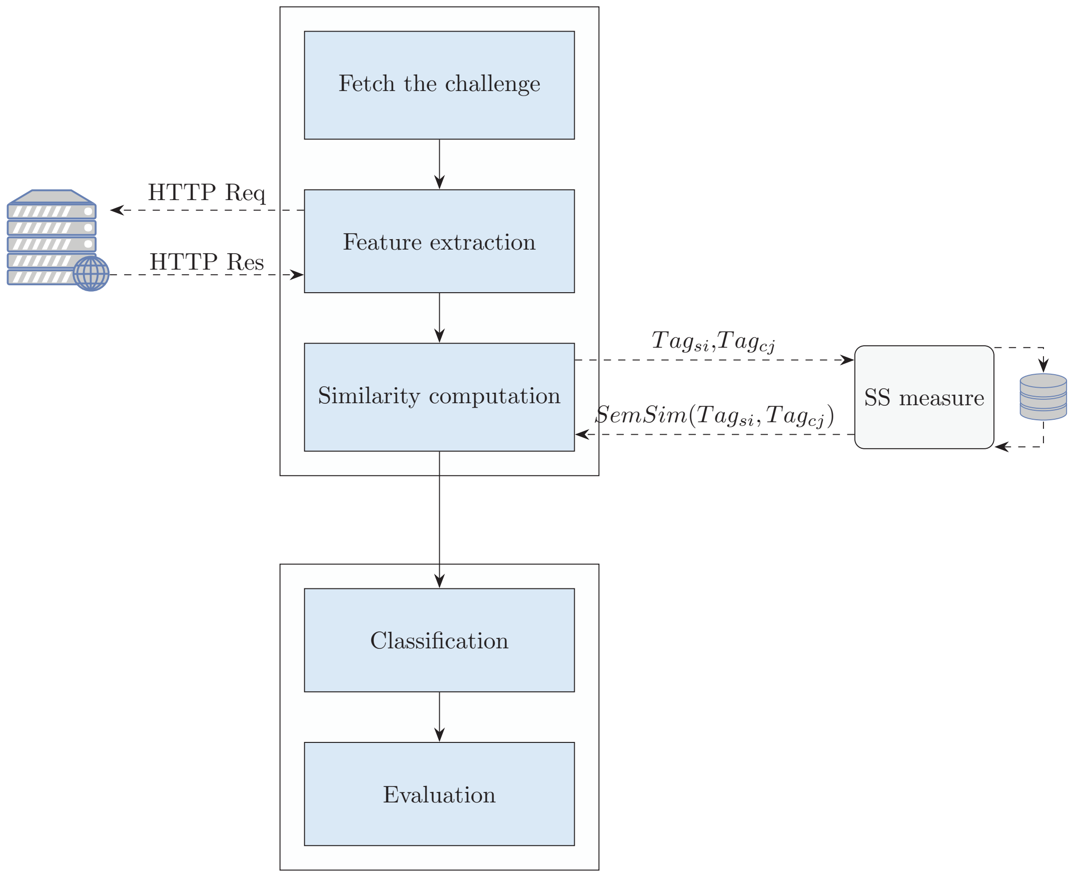


Fig. 1. Attack flow diagram for image CAPTCHA breaking system.

employ deep learning algorithms, their potential for solving image-based CAPTCHAs must be explored. The designed image CAPTCHA-breaking system explores the strength of image-based CAPTCHAs by automatically answering challenges. To answer the challenges, the CAPTCHA-breaking system comprises several stages; Fig. 1 presents a flowchart for the proposed image CAPTCHA-breaking system.

The challenge first needs to be fetched into the image CAPTCHA-breaking system so that it can extract features from the images comprising the challenge. To this end, each image needs to be annotated to acquire tags that describe it by using an annotation technology. For each image, an HTTP request is sent to the offered tagging endpoint, and the HTTP response containing tags describing the recognized entity is received. Each tag is of the noun type, and is associated with a confidence value representing the recognition confidence. The five highest-ranking tags returned by the annotation technology along with their confidence values form features of the initial vector for each image. These vectors contribute to the similarity computation between each candidate image and the sample image from the challenge. Pseudocode 2 explains these steps.

Pseudocode 2 General methodology.

```
1:  while There is a challenge in the dataset do
2:      Fetch challenge from the dataset
3:      Upload the challenge sample image
4:      Tag the challenge sample image
5:      Form the $initialVectorS$ of the sample image
6:
7:      for <each candidate image in the challenge> do
8:          Upload the current candidate image
9:          Tag the current candidate image
10:         Form the $initialVectorC$ of the current candidate image
11:         $SimVec \leftarrow$ Compute_Similarity($initialVectorS, initialVectorC$)
12:         Add $SimVec$ to the results
```

Measuring the similarity between the sample and the initial candidate vectors is an important procedure for determining the level of similarity between them. The Euclidian distance between the confidence values of tags from the sample and candidate initial vectors is first calculated and subtracted from one to allow higher confidence values to reach a higher score. In addition, the semantic similarity between the tags is calculated utilizing the Zhou measure. Zhou et al. (2008) propose a hybrid measurement that quantifies semantic similarity based on the path length between two concepts and the information content value of each concept. A tuning factor (k) is introduced to control the contribution of each path length and IC value. The measure is expressed as:

$$Sim_{Zhou}(c_1, c_2) = 1 - k\left(\frac{log(len(c_1, c_2) + 1)}{log(2(depth\_max - 1))}\right) - (1 - k)\left(\frac{(IC(c_1) + IC(c_2) - 2IC(LCS(c_1, c_2)))}{2}\right)$$

where c_1 and c_2 are two concepts, where the similarity between which is measured, len is the shortest path length between the concepts in the taxonomy, $depth\_max$ is the maximum, LCS is their lowest common subsumer (LCS), which is the most specific ancestor concept for c_1 and c_2, and IC is the information content value assigned for both c_1 and c_2 in the ontology. A recent comparative study (Slimani, 2013) showed that the Zhou measure achieves the highest accuracy in comparison to a set of semantic similarity measurement based on WordNet ontology. Therefore, the similarity score is calculated as:

$$SimilarityScore_{si,cj} = SemSim_{Zhou}(\ tag_{si},\ tag_{cj}) * \left(1 - \sqrt{(conf_{si} - conf_{cj})^2}\right)$$

where $SemSim_{Zhou}(\ tag_{si}, tag_{cj})$ is the semantic similarity between tag_i in the initial sample vector s and tag_j in initial candidate vector c from the challenge. It is calculated using the Zhou measure, which returns values falling in the range [0, 1], with zero for non-semantically similar tags and one for identical tags; $conf_{si}$ is the confidence value of the tag_i in the initial sample vector s; and $conf_{cj}$ is the confidence value of tag_j in the initial candidate vector c. They take values between zero and one, where one is the highest confidence value and zero the lowest. As a result, the similarity scores are in the range [0, 1]. After calculating all similarity scores, they can be accumulated as follows:

$$AccumulatedScore = \sum_{i=1}^{n}\sum_{j=1}^{m} SimilarityScore_{si,cj}$$

where $SimilarityScore_{si,cj}$ is the similarity score between tag i in the initial sample vector s and tag j in the initial candidate vector c; m represents the number of annotated tags from the candidate image, and n is the number of annotated tags from the sample image. In our study n and m were set to five. Hence, after comparing an initial candidate vector with the initial sample vector, the resulting similarity vector contained 25 similarity scores as well as the accumulated score, and these formed the features to be fed into the next classification step. Pseudocode 3 explains the similarity computation.

Another version of the proposed breaking system was designed by modifying the way in which the initial vectors are formulated. This modification was achieved by comparing each returned tag with the relevant question hint. For tags that were identical to the question hint, the confidence values were changed to one.

After formulating all similarity vectors, the classification algorithms are employed, where NB, CART, bagging with CART, and RF performed the learning process and constructed the learned models. By feeding the similarity vectors of all challenges into the dataset, each classification algorithm classified similarity vectors depending on the underlying theory.

NB calculates the probabilities for each class given the similarity vectors, and based on this, each similarity vector is assigned to the class with the highest probability value. CART generates a classification tree based on the Gini index when evaluating a variable as the splitting point. A vector is then classified by testing the features values against the decision tree. A path is traced from the root to a leaf node that contains the class prediction for that vector. Bagging builds several CART models based on different bootstrap training sets and classifies a vector based on the majority vote of the obtained CART models. RF constructs several random trees trained over different bootstrap samples and, based on the majority vote of the trees, predicts the class of the similarity vectors.

Pseudocode 3 Compute_similarity.

```
1: $SimVec$
2: $AccumlatedValue$
3: for <each $tag_{si}$ in the sample initial vector> do
4:     for <each $tag_{cj}$ in the candidate initial vector> do
5:         $SimilarityScore_{si,cj} \leftarrow SemSim_{Zhou}(\ tag_{si},\ tag_{cj}) * \left(1 - \sqrt{(conf_{si} - conf_{cj})^2}\right)$
6:         $AccumlatedValue = AccumlatedValue + SimilarityScore$
7:         Add $SimilarityScore$ to $SimVec$
8: Add $AccumlatedValue$ to $SimVec$
9: return $SimVec$
```

For evaluation, k-fold cross-validation was used. The dataset was divided into k disjoint folds, each containing approximately the same number of similarity vectors. For each fold, a model was constructed by training using all remaining folds and testing using the given fold. This procedure was repeated k times. The generated models for NB, CART, bagging with CART, and RF were then combined to generate the final models.

To evaluate the performance of the proposed CAPTCHA-breaking system, the following performance measures were utilized: average accuracy, percentage of passed challenges, and area under the ROC curve (AUC). Due to the flexibility embedded into Google's image reCAPTCHA owing to the small size of images that might not be discernible even to humans (Sivakorn et al., 2016a), a challenge was considered passed even if no more than one incorrect selection was provided. This helped attackers gain an advantage by being allowed an extra chance to correctly pass the presented challenge even if the classifier yielded an incorrect selection.

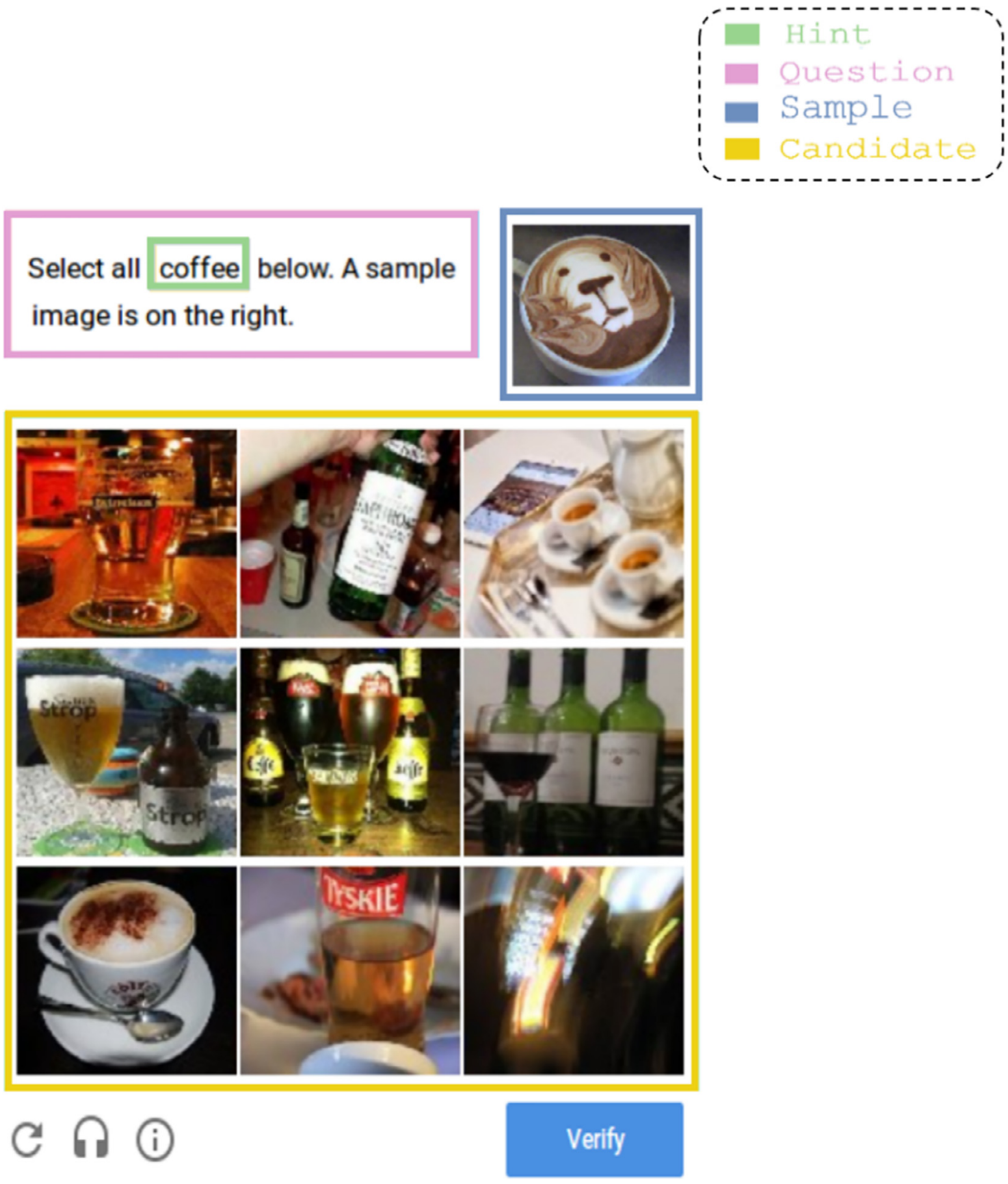
5. Experiment

5.1. Dataset

The Google image reCAPTCHA dataset is a collection of Google image reCAPTCHA challenges (Sivakorn et al., 2016b). It contains 700 challenges, and each consists of a sample image and nine candidate images as well as an instruction file. The instruction file contains information on the challenge, such as the associated challenge instruction and hint. Each image in the dataset is in a portable network graphics (PNG) format with a width and height of 100 × 100 px. Fig. 2 shows an example of a challenge in the dataset. In addition to the challenges, the correct answers have been marked and answered by a human.

5.2. Tools

A variety of tools were needed to conduct the proposed attack. An auto-tagging service was used to return tags describing the challenge images. Imagga (Imagga, 2019) is an impressive image annotation technology that offers several APIs



Select all coffee below. A sample image is on the right.

Fig. 2. Example of image reCAPTCHA dataset challenges (Sivakorn et al.,2016b).

for image analysis as a web service. The auto-tagging API is one of its services based on deep learning, where complex CNN models are employed along with other approaches that employ large sets of human-annotated data and semantic representations. For more reliable tagging, Imagga does not retrieve any tag with confidence value lower than 7%. To compute the semantic similarity between tags of the sample and the candidate images, WordNet-based Semantic Similarity (WNetSS) API was integrated in the proposed image-based CAPTCHA-breaking system. This provided several WordNet semantic similarity measures, categorized as taxonomic features and information content measures. It also offered an assessment model comprising various well-known benchmarks in the semantic similarity field (Aouicha et al., 2016). It maintained a MySQL database to store “is-a” topological parameters of WordNet 3.0, such as depth, hyponyms, ancestors, and lowest common subsumers. For machine learning, the Waikato Environment for Knowledge Analysis (Weka) library (Witten et al., 2016) was used to leverage the built-in NB and RF algorithms.

5.3. Performance measures

The performance of the CAPTCHA-breaking system was evaluated based on a number of performance measures. Average Accuracy: The accuracy measure calculates the overall effectiveness of the selected classifiers. The formula used to calculate the accuracy is:

$$Accuracy = \frac{tp + tn}{tp + fn + fp + tn}$$

where *tp* is the true positives representing correctly recognized images that answered the challenges, *tn* stands for true negatives that represent correctly recognized images that did not answer the challenges, *fp* represents false positives, the number of images incorrectly assigned as answers to the challenges, and *fn* shows false negatives representing the number of correct images misclassified as wrong answers (Sokolova and Lapalme, 2009). The average accuracy was the accuracy obtained by each classification model averaged by the number of models.

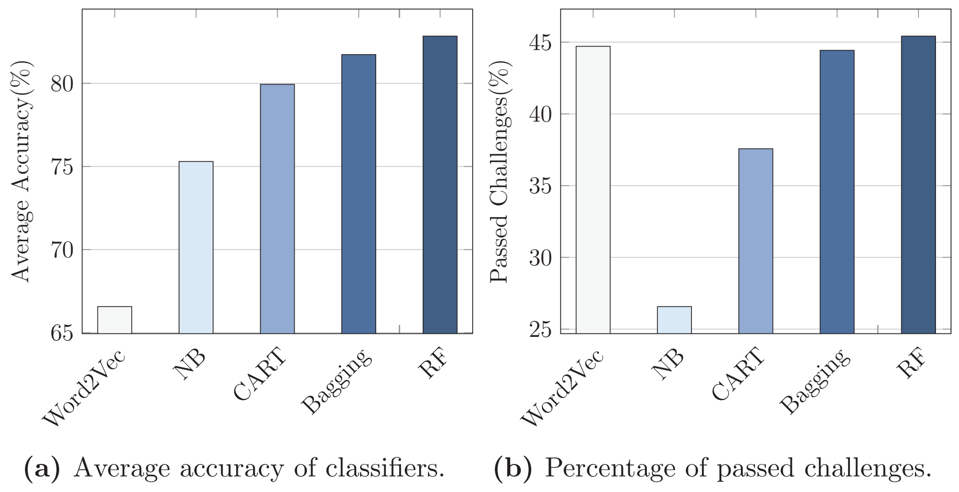


Fig. 3. Average accuracy and percentage of passed challenges by classification models compared with Word2Vec.

Percentage of passed challenges: This is the number of passed challenges divided by the total number of challenges in the dataset, where a challenge was deemed a passed challenge if no or a maximum of one wrong selection was obtained (Sivakorn et al., 2016a).

Area under the ROC curve (AUC): The AUC metric captures a single point on the receiving operating characteristic curve (ROC). It measures the ability of the classifiers to avoid false classification. It was calculated as:

$$AUC = \frac{1}{2}\left(\frac{tp}{tp+fn} + \frac{tn}{tn+fp}\right)$$

where *tp* are the true positives representing the correctly recognized images that answered the challenges, *tn* stands for true negatives that represent correctly recognized images that did not answer the challenges, *fp* are false positives, the number of images that were incorrectly assigned as answers to challenges, and *fn* are false negatives, which represent the number of correct images misclassified as wrong answers (Sokolova and Lapalme, 2009).

5.4. Experimental settings

In the experiment, several parameters needed to be set. The tuning parameter k introduced in the Zhou measure was set to 0.5 as assumed in Zhou et al. (2008), which indicates that the path length and IC value had the same weight. As for setting the parameters for the machine learning algorithms, the nature of the NB algorithm rendered them independent of the need for parameter tuning. Nonetheless, using supervised discretization in processing the attributes can significantly improve the performance of NB prediction. CART had no parameters to set. Bagging was used with CART as base classifier, and the size of each bag was 100.

On the contrary, the RF algorithm had parameters that needed to be tuned: the number of random features, size of the bag, number of trees in the forest, and number of nodes in each tree. Breiman (1999) recommended setting the number of random features m to the square root of the total number of features p ($m = \sqrt{p}$), and the size of the bag was two-thirds the number of instances in the dataset. Therefore, $m = 6$ and the size of the bag $N = 4200$ in this study. The number of trees B in the forest and number of nodes n_{min} in each tree were tuned experimentally, where the best results were achieved when $B = 65$ and $n_{min} = 48$. For training and evaluating the classification models, K-folds cross validation was used with $k = 10$.

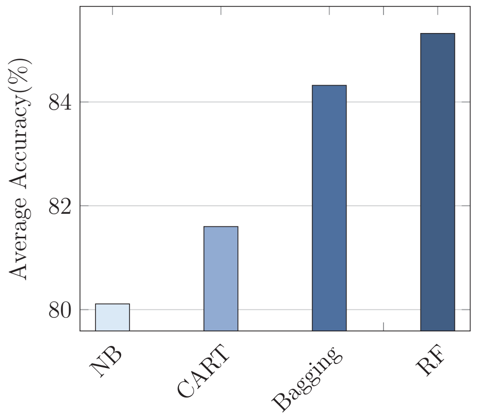
6. Results and analysis

The dataset described in Section 5.1 was used to train and test the proposed CAPTCHA-breaking system. During the feature extraction process, a small number of the tags returned by the utilized tagging service were adjectives, and yielded a zero-value similarity score because the semantic similarity measure was computed using noun tags. As a result, these returned tags needed to be converted into their corresponding nouns, and similarity scores for these tags were recalculated.

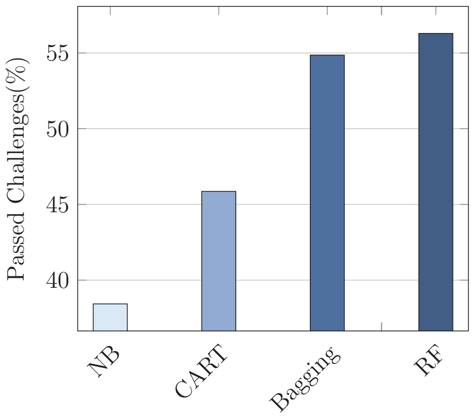
The extracted features were then used to train and evaluate the classifiers based on 10-fold cross-validation. Fig. 3 shows the experimental results, and Fig. 3(a) shows the average accuracy obtained by the NB, CART, bagging, and RF classifiers compared with the Word2Vec classifier that was used in the previous attack on reCAPTCHA (Sivakorn et al., 2016a). The results show that the RF classifier achieved the best performance, with an average accuracy of 82.84% compared with values of 81.73%, 79.93%, 75.30% 66.57% delivered by bagging, CART, NB, and Word2Vec, respectively. Fig. 3(b) shows the percentage of challenges passed by each classifier. RF and Word2Vec had comparable percentages, where the former passed 45.42% and the latter 44.71% of the challenges. Bagging passed 44.43% and CART passed 37.57%. NB achieved the lowest percentage, passing only 26.57% of the challenges.

A study conducted by Sivakorn et al. (2016a) suggests that by considering the hint, the percentage of passed challenges can be significantly increased by 1.5–15.5% (Sivakorn et al., 2016a). Therefore, another version of the proposed breaking system was designed by modifying the way in which initial vectors are formulated. This modification was achieved by comparing each returned tag with the hint for the corresponding question. For tags that were identical to the hint, the confidence values were changed to one. Fig. 4 presents the average accuracy and percentage of passed challenges after considering the hint. Fig. 4(a) shows the improvement in accuracy. RF yielded an accuracy of 85.32%, bagging scored 84.32%, CART yielded 81.60% and NB achieved 80.11%. The improvement in the percentage of passed challenges is shown in Fig. 4(b), where RF passed 56.29% of the challenges, bagging 54.86%, CART 45.86% and NB passed 38.43% of the challenges. Fig. 5 summarizes the average accuracy and percentage of passed challenges results by each classifier before and after adding the hint.

Another comparison between the performance of the NB, CART, bagging, and RF classifiers based on the ROC curve is presented in Fig. 6. RF was more discriminative in classifying the images than the other classifiers. In the first experiment, it achieved an AUC of

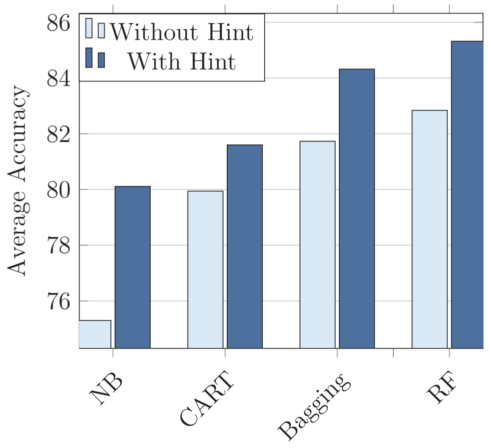


(a) Average accuracy of classifiers.

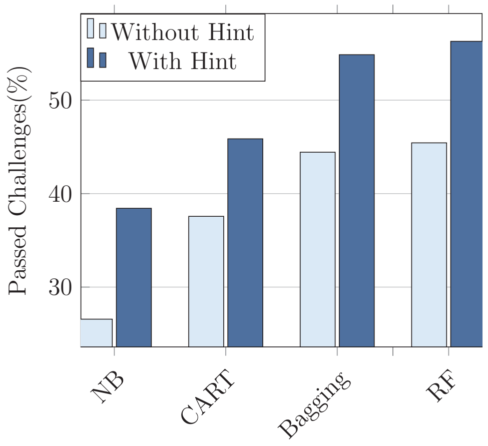


(b) Percentage of passed challenges.

Fig. 4. Average accuracy and passed challenges percentage of classification models after considering the hint.

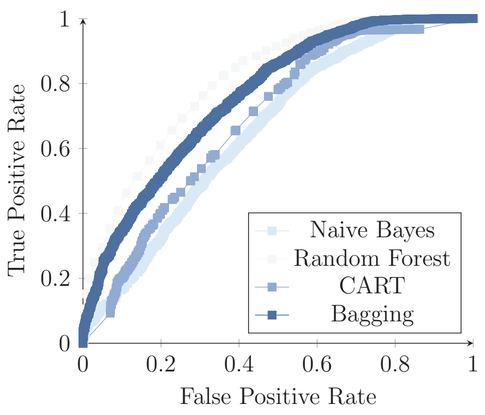


(a) Average accuracy of classifiers.

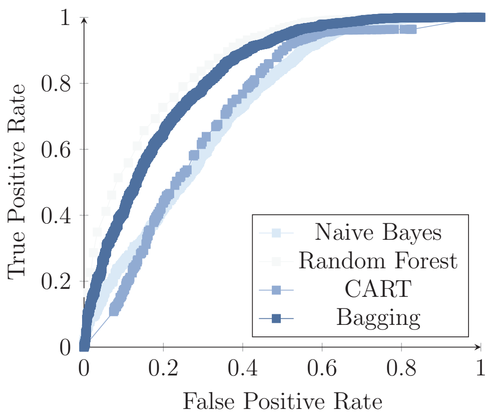


(b) Passed challenges percentage

Fig. 5. Classifiers results before and after considering the hint.



(a) Naive Bayes, Random Forest, CART, and Bagging ROC



(b) Naive Bayes, Random Forest, CART, and Bagging ROC after considering the hint

Fig. 6. Naïve Bayes, random forest, CART, and bagging ROC.

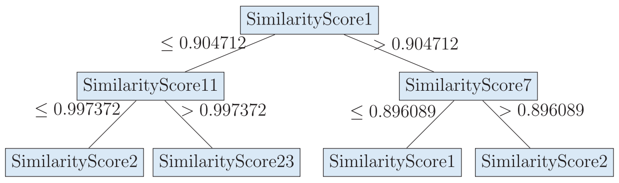


Fig. 7. Part of the obtained CART.

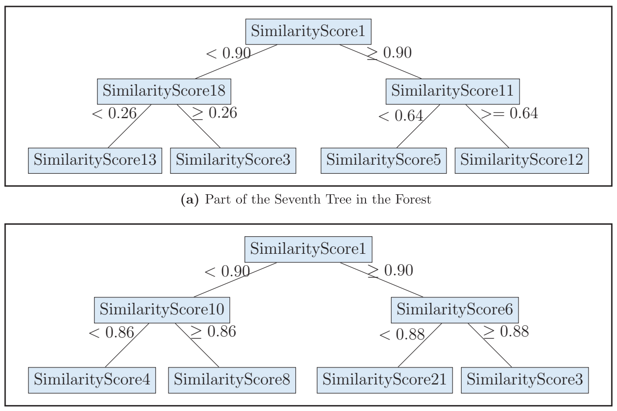


Fig. 8. Sample of the trees in the random forest.

0.81, while the values for bagging, CART, and NB were 0.76, 0.69, and 0.66, respectively. When considering the hint, RF obtained an AUC of 0.86, bagging 0.82, and CART and NB obtained an AUC value of 0.73.

Figs. 7 and 8 show part of the CART and a sample of RF trees, respectively. The first semantic similarity feature was selected as the first splitting node in 23% of the trees in RF while the sixth semantic similarity feature was the first splitting node in 16.90% of the trees, and the second and seventh semantic similarity features were the first splitting nodes in 15.4% of the trees in the forest. The accumulated score was chosen as the first and second splitting points in 0.03% of the RF trees.

A Mann–Whitney *U* test was conducted to observe the statistical significance among the results of all classifiers applied, where the null hypothesis (H0) is stated as there is no difference between the ranks of any pair of compared methods, and the alternative hypothesis (H1) is stated as there is a difference between the ranks of each compared two methods. As $p < 0.05$, the null hypothesis is rejected, meaning that the distribution differs significantly.

The results clearly indicate that recent advances in deep learning and machine learning can be harnessed by attackers to break the security of online systems and services protected by Google's image reCAPTCHA. CNN models play a major role in analyzing and describing challenge images as these models produce precise descriptions of images that help conduct successful attacks. Learning and extracting more discriminative features from the presented images provides more opportunities for classification algorithms to produce models that have the ability to make better classification decisions.

Moreover, the nature of the machine learning algorithms used can significantly affect the success of the attack. Ensemble learning methods, especially those based on bagging, obtained better prediction performance (Wang et al., 2011) because they quantified a classification decision based on the majority vote of several models resulting from training on different samples of the data. This led to a more robust and accurate classification than that obtained by relying on the decision of a single model.

The effectiveness of the RF classifier as an ensemble learning algorithm in answering challenges was proven before and after adding hints. The results obtained by it were superior to those of its base model (CART) as well as bagging with CART (Fig. 5) (Kuhn and Johnson, 2013). It reduced variance and increased bias by creating several trees, each trained over different samples of the dataset, and through the random selection of a feature subset for the split at each node.

NB obtained the lowest average accuracy, percentage of passed challenges, and AUC among the considered classifiers. However, it successfully attacked more than a quarter of the presented challenges, which implies that more sophisticated algorithms, such as deep neural networks and other biologically inspired algorithms, may help answer a greater number of challenges.

The first and sixth semantic similarity features were the most discriminative because they included the results of the two highest-ranking tags were essential to describing objects in the sample and the candidate images. Similarly, the second and seventh were the second-highest discriminative features in the system because they comprised the second-highest-ranking tags acquired from the CNN models.

The performance of the proposed attack was significantly more effective than previous attacks. The average accuracy obtained and the number of challenges the system could pass outperformed the results reported for previously proposed attacks. The use of recent deep learning technologies and machine learning algorithms for the automatic recognition of Google image reCAPTCHA challenges helped carry out successful attacks. Furthermore, leveraging semantic similarity measurements create an added value that helped the machine learning algorithms to make knowledgeable guesses in response to CAPTCHA challenges.

7. Conclusion and future work

This study focused on image-based CAPTCHA, which is an alternative to text-based and audio CAPTCHA schemes. A breaking system was proposed to attack the Google image reCAPTCHA, a service provided by Google, to examine its strength against machine learning attacks. The results indicate that recent advancements in deep and machine learning can significantly help attackers compromise online systems and services protected by Google image reCAPTCHA. Moreover, extracting more discriminative features from images help conduct successful attacks. In future work, lemmatization techniques will be integrated to automate the approach based on detecting adjectives for returned tags. A variety of machine learning algorithms will also be used and their effect on answering challenges will be investigated.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the Research Center of College of Computer and Information Sciences, King Saud University. The authors are grateful for this support.

References

Aghdam, H., Heravi, E., 2017. Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification. Springer International Publishing.

Aouicha, M.B., Taieb, M.A.H., Hamadou, A.B., 2016. SISR: system for integrating semantic relatedness and similarity measures. Soft Comput. 1–25.

Basso, A., Sicco, S., 2009. Preventing massive automated access to web resources. Comput. Secur. 28 (3), 174–188.
Bel, L., Allard, D., Laurent, J., Cheddadi, R., Bar-Hen, A., 2009. Cart algorithm for spatial data: application to environmental and ecological data. Comput. Stat. Data Anal. 53 (8), 3082–3093.
Biau, G., Scornet, E., 2016. A random forest guided tour. Test 25 (2), 197–227.
Blum, M., von Ahn, L., Langford, J., Hopper, N., 2000. The CAPTCHA Project, Completely Automatic Public Turing Test to Tell Computers and Humans Apart. Department of Computer Science, Carnegie-Mellon University.
Breiman, L., 1999. Random Forests–Random Features. Tech. Rep. 567.
Breiman, L., 2001. Random Forests. Mach. Learn. 45 (1), 5–32.
Breiman, L., 2017. Classification and Regression Trees. CRC Press. https://books.google.com.sa/books?id=gLs6DwAAQBAJ.
Brodic, D., Petrovska, S., Jevtic, M., Milivojevic, Z., 2016. The influence of the CAPTCHA types to its solving times. In: Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2016 39th International Convention on. IEEE, pp. 1274–1277.
Bursztein, E., Beauxis, R., Paskov, H., Perito, D., Fabry, C., Mitchell, J., 2011. The failure of noise-based non-continuous audio captchas. In: 2011 IEEE Symposium on Security and Privacy, pp. 19–31. doi:10.1109/SP.2011.14.
Bursztein, E., Martin, M., Mitchell, J., 2011. Text-based CAPTCHA strengths and weaknesses. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. ACM, pp. 125–138.
Convolutional Neural Networks (CNNs / ConvNets), 2017. Stanford University http://cs231n.github.io/convolutional-networks/. [Online; Accessed 5 November 2017].
Cady, F., 2017. Machine Learning Overview. John Wiley & Sons, pp. 87–91. Chapter 6.
Chan, T.-Y., 2003. Using a test-to-speech synthesizer to generate a reverse turing test. In: Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on. IEEE, pp. 226–232.
Chandavale, A.A., Sapkal, A.M., 2015. Systematic approach to measure strength of text based CAPTCHA. In: 2015 International Conference on Information Processing (ICIP), pp. 382–387. doi:10.1109/INFOP.2015.7489412.
Chew, M., Baird, H.S., 2003. BaffleText: a human interactive proof. DRR 5010, 305–316.
Corley, C., Mihalcea, R., 2005. Measuring the semantic similarity of texts. In: Proceedings of the ACL workshop on empirical modeling of semantic equivalence and entailment. Association for Computational Linguistics, pp. 13–18.
Datta, R., Li, J., Wang, J.Z., 2005. Imagination: a robust image-based CAPTCHA generation system. In: Proceedings of the 13th annual ACM international conference on Multimedia. ACM, pp. 331–334.
Duffner, S., 2009. Face Image Analysis with Convolutional Neural Networks. Dokument (GRIN Verlag). GRIN Verlag.
Elson, J., Douceur, J.R., Howell, J., Saul, J., 2007. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In: ACM Conference on Computer and Communications Security, 7. Citeseer, pp. 366–374.
Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Mach. Learn. 29 (2–3), 131–163.
Fidas, C.A., Voyiatzis, A.G., Avouris, N.M., 2011. On the necessity of user-friendly CAPTCHA. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 2623–2626.
Gao, H., Yao, D., Liu, H., Liu, X., Wang, L., 2010. A novel image based CAPTCHA using jigsaw puzzle. In: 2010 13th IEEE International Conference on Computational Science and Engineering, pp. 351–356. doi:10.1109/CSE.2010.53.
Gargi, U., 2013. Video-based CAPTCHA. US Patent 8,510,795. https://www.google.com/patents/US8510795.
Gulli, A., Pal, S., 2017. Deep Learning with Keras. Packt Publishing.
Han, J., Pei, J., Kamber, M., 2011. Classification and Prediction. Elsevier, pp. 285–382. Chapter 6.
Hastie, T., Tibshirani, R., Friedman, J., 2009. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer Series in Statistics, second ed. Springer, New York. https://books.google.com.sa/books?id=tVIjmNS3Ob8C.
Imagga, 2019. Powerful Image Recognition APIs for Automated Categorization & Tagging. https://imagga.com.
Karpathy, A., Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3128–3137.
Kluever, K.A., 2008. Evaluating the Usability and Security of a Video CAPTCHA. Rochester Institute of Technology.
Kluever, K.A., Zanibbi, R., 2009. Balancing usability and security in a video CAPTCHA. In: Proceedings of the 5th Symposium on Usable Privacy and Security. ACM, pp. 11–14.
Kuhn, M., Johnson, K., 2013. Applied Predictive Modeling. SpringerLink : Bücher, Springer, New York. https://books.google.com.sa/books?id=xYRDAAAAQBAJ.
Lazar, J., Feng, J., Brooks, T., Melamed, G., Wentz, B., Holman, J., Olalere, A., Ekedebe, N., 2012. The soundsright CAPTCHA: an improved approach to audio human interaction proofs for blind users. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, pp. 2267–2276.
Leacock, C., Chodorow, M., 1998. Combining local context and WordNet similarity for word sense identification. In: WordNet: An Electronic Lexical Database, 49 (2), pp. 265–283.
LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444.
Li, Q., 2015. A computer vision attack on the artifacial CAPTCHA. Multimedia Tools Appl. 74 (13), 4583–4597.
Li, Y., Bandar, Z.A., McLean, D., 2003. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans. Knowl. Data Eng. 15 (4), 871–882.
Liang, L., Xiao, R., Wen, F., Sun, J., 2008. Face alignment via component-based discriminative search. In: Computer Vision–ECCV 2008, pp. 72–85.
Liu, X., 2007. Generic face alignment using boosted appearance model. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. doi:10.1109/CVPR.2007.383265.
Lorenzi, D., Vaidya, J., Uzun, E., Sural, S., Atluri, V., 2012. Attacking image based CAPTCHAs using image recognition techniques. In: International Conference on Information Systems Security. Springer, pp. 327–342.
Meng, L., Gu, J., Zhou, Z., 2012. A new model of information content based on concept's topology for measuring semantic similarity in WordNet. Int. J. Grid Distrib. Comput. 5 (3), 81–94.
Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781.
Miller, G., Fellbaum, C., 1998. WordNet: An Electronic Lexical Database.
Miller, G.A., 1995. Wordnet: a lexical database for english. Commun. ACM 38 (11), 39–41.
Mohri, M., Rostamizadeh, A., Talwalkar, A., 2012. Foundations of Machine Learning. MIT Press. https://books.google.com.sa/books?id=-ijiAgAAQBAJ.
Moy, G., Jones, N., Harkless, C., Potter, R., 2004. Distortion estimation techniques in solving visual CAPTCHAs. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004., 2 doi:10.1109/CVPR.2004.1315140. II–II.
Paluszek, M., Thomas, S., 2017. An Overview of Machine Learning. Apress, pp. 3–15. Chapter 1.
Patterson, J., Gibson, A., 2017. Deep Learning: A Practitioner's Approach. O'Reilly Media, Inc..
Ponec, M., 2006. Visual reverse turing tests: a false sense of security. In: 7th IEEE Information Assurance Workshop, Westpoint, New York, USA, pp. 305–311.
Pope, C., Kaur, K., 2005. Is it human or computer? defending e-commerce with CAPTCHAs. IT Prof. 7 (2), 43–49.
Romero, C., Espejo, P.G., Zafra, A., Romero, J.R., Ventura, S., 2013. Web usage mining for predicting final marks of students that use moodle courses. Comput. Appl. Eng. Educ. 21 (1), 135–146.
Rui, Y., Liu, Z., 2004. Artifacial: automated reverse turing test using facial features. Multimedia Syst. 9 (6), 493–502.
Saha, S.K., Nag, A.K., Dasgupta, D., 2015. Human-cognition-based captchas. IT Prof. 17 (5), 42–48.
Sammut, C., Webb, G.I., 2011. Encyclopedia of Machine Learning. Springer Science and Business Media.
Sathyadevi, G., 2011. Application of cart algorithm in hepatitis disease diagnosis. In: 2011 International Conference on Recent Trends in Information Technology (ICRTIT). IEEE, pp. 1283–1287.
Sebti, A., Barfroush, A.A., 2008. A new word sense similarity measure in WordNet. In: Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on. IEEE, pp. 369–373.
Sermanet, P., Chintala, S., LeCun, Y., 2012. Convolutional neural networks applied to house numbers digit classification. In: Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pp. 3288–3291.
Shirali-Shahreza, S., Shirali-Shahreza, M.H., 2008. Bibliography of works done on CAPTCHA. In: 2008 3rd International Conference on Intelligent System and Knowledge Engineering, 1, pp. 205–210. doi:10.1109/ISKE.2008.4730926.
Sivakorn, S., Polakis, I., Keromytis, A.D., 2016a. I am robot: (deep) learning to break semantic image captchas. In: 2016 IEEE European Symposium on Security and Privacy (EuroSP), pp. 388–403. doi:10.1109/EuroSP.2016.37.
Sivakorn, S., Polakis, I., Keromytis, A.D., 2016b. Image Captcha Dataset https://www.cs.columbia.edu/~suphannee/captcha/.
Slimani, T., 2013. Description and evaluation of semantic similarity measures approaches. arXiv:1310.8059.
Snyder, C., Myer, T., Southwell, M., 2010. Using CAPTCHAs. Springer, pp. 117–131. Chapter 9.
Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manag. 45 (4), 427–437.
Taieb, M.A.H., Aouicha, M.B., Hamadou, A.B., 2014. Ontology-based approach for measuring semantic similarity. Eng. Appl. Artif. Intell. 36, 238–261.
Tian, D., Xu, X., Tao, Y., Wang, X., 2017. An improved activity recognition method based on smart watch data. In: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), 1. IEEE, pp. 756–759.
Vicente, J., Álvarez-Sánchez, J., de la Paz López, F., Moreo, J., Adeli, H., 2017. Natural and Artificial Computation for Biomedicine and Neuroscience. Springer International Publishing.
Vikram, S., Fan, Y., Gu, G., 2011. Semage: a new image-based two-factor CAPTCHA. In: Proceedings of the 27th Annual Computer Security Applications Conference. ACM, pp. 237–246.
Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M., 2008. reCAPTCHA: human-based character recognition via web security measures. Science 321 (5895), 1465–1468.
Wagh, K., Kolhe, S., 2012. A new approach for measuring semantic similarity in ontology and its application in information retrieval. In: International Workshop on Multi-disciplinary Trends in Artificial Intelligence. Springer, pp. 122–132.
Wang, G., Hao, J., Ma, J., Jiang, H., 2011. A comparative assessment of ensemble learning for credit scoring. Expert Syst. Appl. 38 (1), 223–230.
Wang, J., 2008. Data Warehousing and Mining: Concepts, Methodologies, Tools, and Applications, 3. IGI Global.
Wikipedia, 2017a. Convolutional Neural Network. [Online; Accessed 5 November 2017].

Wikipedia, 2017b. Machine Learning. Wikipedia, the free encyclopedia. [Online; Accessed 3 November 2017].

Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., 2016. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann.

Wu, Z., Palmer, M., 1994. Verbs semantics and lexical selection. In: Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, pp. 133–138.

Xiao, R., Zhu, L., Zhang, H.-J., 2003. Boosting chain learning for object detection. In: Proceedings Ninth IEEE International Conference on Computer Vision, 1, pp. 709–715. doi:10.1109/ICCV.2003.1238417.

Yamamoto, T., Tygar, J.D., Nishigaki, M., 2010. CAPTCHA using strangeness in machine translation. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 430–437. doi:10.1109/AINA.2010.55.

Yan, J., El Ahmad, A.S., 2008. Usability of CAPTCHAs or usability issues in CAPTCHA design. In: Proceedings of the 4th Symposium on Usable Privacy and Security. ACM, pp. 44–52.

Yan, J., El Ahmad, A.S., 2011. Captcha robustness: a security engineering perspective. Computer 44 (2), 54–60.

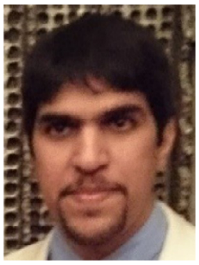
Yan, J., Yu, S.-Y., 2009. Streamlining attacks on CAPTCHAs with a computer game. In: IJCAI–09, Twenty-First International Joint Conference on Artificial Intelligence. AAAI, pp. 2095–2100.

Zhang, W., 2010. Zhang's CAPTCHA architecture based on intelligent interaction via ria. In: Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, 6. IEEE, pp. 57–62.

Zhou, Z., Wang, Y., Gu, J., 2008. New model of semantic similarity measuring in WordNet. In: 2008 3rd International Conference on Intelligent System and Knowledge Engineering, 1, pp. 256–261. doi:10.1109/ISKE.2008.4730937.

Zhu, B.B., Yan, J., Li, Q., Yang, C., Liu, J., Xu, N., Yi, M., Cai, K., 2010. Attacks and design of image recognition CAPTCHAs. In: Proceedings of the 17th ACM Conference on Computer and Communications Security. ACM, pp. 187–200.

Fatmah H. Alqahtani received the B.Sc. degree in information technology from King Saud University, Riyadh, Saudi Arabia in 2014 and the M.Sc. degree in computer science from King Saud University, Riyadh, Saudi Arabia in 2018. She is currently working as a parttime lecturer in Princess Nora bint Abdul Rahman University, Riyadh, Saudi Arabia. Her main research interests are in machine learning and pattern recognition and their applications in the fields of computer security and wireless networks.



Fawaz A. Alsulaiman received the M.S. and Ph.D. degrees in computer science from the University of Ottawa, Ottawa, ON, Canada, in 2006 and 2013 respectively. He is an assistant professor in Computer Science Department, College of Computer and Information Sciences, at King Saud University. His research interests are in the field of haptics, information security, including user authentication and authorization, artificial intelligence, and multimedia.