# No Bot Expects the DeepCAPTCHA! Introducing Immutable Adversarial Examples, With Applications to CAPTCHA Generation

Margarita Osadchy, Julio Hernandez-Castro, Stuart Gibson, Orr Dunkelman, and Daniel Pérez-Cabo

***Abstract*— Recent advances in deep learning (DL) allow for solving complex AI problems that used to be considered very hard. While this progress has advanced many fields, it is considered to be bad news for Completely Automated Public Turing tests to tell Computers and Humans Apart (CAPTCHAs), the security of which rests on the hardness of some learning problems. In this paper, we introduce DeepCAPTCHA, a new and secure CAPTCHA scheme based on *adversarial examples*, an inherit limitation of the current DL networks. These adversarial examples are constructed inputs, either synthesized from scratch or computed by adding a small and specific perturbation called *adversarial noise* to correctly classified items, causing the targeted DL network to misclassify them. We show that plain adversarial noise is insufficient to achieve secure CAPTCHA schemes, which leads us to introduce *immutable adversarial noise*—an adversarial noise that is resistant to removal attempts. In this paper, we implement a proof of concept system, and its analysis shows that the scheme offers high security and good usability compared with the best previously existing CAPTCHAs.**

***Index Terms*— CAPTCHA, deep learning, CNN, adversarial examples, HIP.**

## I. Introduction

CAPTCHAS are traditionally defined as automatically constructed problems, that are very difficult to solve for artificial intelligence (AI) algorithms, but easy for humans. Due to the fast progress in AI, an increasing number of CAPTCHA designs have become ineffective, as the underlying AI problems have become solvable by algorithmic tools. Specifically, recent advances in Deep Learning (DL) reduced the gap between human and machine ability in solving problems that have been typically used in CAPTCHAs in the past. A series of breakthroughs in AI even led some researches to claim that DL would lead to the "end" of CAPTCHAs [4], [14].

However, despite having achieved human-competitive accuracy in complex tasks such as speech processing and image recognition, DL still has some important shortcomings with regards to human ability [42]. In particular, they are vulnerable to small perturbations of the input, that are imperceptible by humans but can cause misclassification. Such perturbations, called *adversarial noise*, can be specially crafted for a given input that forces misclassification by the Machine Learning (ML) model.

Although initially discovered in the specific context of Deep Learning, this phenomenon was observed later in other classifiers, such as linear, quadratic, decision trees, and KNN [15], [33], [35], [42]. Moreover, Szegedy et al. [42] showed that adversarial examples designed to be misclassified by one ML model are often also misclassified by different (unrelated) ML models. Such transferability allows adversarial examples to be used in misclassification attacks on machine learning systems, even without having access to the underlying model [33], [35]. Consequently, adversarial examples pose a serious security threat for numerous existing machine learning based solutions such as those employing image classification (e.g., biometric authentication, OCR), text classification (e.g., spam filters), speech understanding (e.g, voice commands [6]), malware detection [48], and face recognition [40].

On the other hand, adversarial examples can be used in a constructive way and improve computer security. In this paper, we propose using adversarial examples for CAPTCHA generation within an object classification framework, involving a large number of classes. Adversarial examples are appealing for CAPTCHA applications as they are very difficult for Machine Learning tools (in particular advanced DL networks) and easy for humans (adversarial noise tends to be small and does not affect human perception of image content).[1]

To provide a secure CAPTCHA, adversarial examples 1) should be effective against any ML tool and 2) should be robust to preprocessing attacks, that aim to remove the adversarial noise.

### A. Effectiveness of Adversarial Examples Against ML

The ML community has been actively searching for methods that are robust to adversarial examples. The most effective

Manuscript received December 9, 2016; revised March 13, 2017 and May 20, 2017; accepted June 12, 2017. Date of publication June 21, 2017; date of current version July 26, 2017. This work was supported by the U.K. Engineering and Physical Sciences Research Council under Project EP/M013375/1 and in part by the Israeli Ministry of Science and Technology under Project 3-11858. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Shouhuai Xu. *(Corresponding author: Margarita Osadchy.)*

M. Osadchy and O. Dunkelman are with the Computer Science Department, University of Haifa, Haifa 31905, Israel (e-mail: rita@cs.haifa.ac.il).

J. Hernandez-Castro is with the School of Computing, University of Kent, Canterbury, CT2 7NF, U.K.

S. Gibson is with the School of Physical Sciences, University of Kent, Canterbury, CT2 7NH, U.K.

D. Pérez-Cabo is with Gradiant, Campus Universitario de Vigo, 36310 Vigo, Spain.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2718479

[1]The idea of using adversarial images for CAPTCHA was independently suggested in [41] as a general concept.

among the proposed solutions, but still very far from providing sufficient robustness, is training the model on adversarial examples [15], [42]. The more sophisticated approaches include 1) training a highly specialized network to deal with very specific types of adversarial noise [34]; 2) combining autoencoders, trained to map adversarial examples to clean inputs, with the original network [16]. The combined architecture adds a significant amount of computation and at the same time is vulnerable to adversarial examples, crafted for the new architecture.

We note (and later discuss) that high-capacity models (such as Radial Basis Functions (RBF)) are more robust to adversarial examples, but they are unable to cope with large-scale tasks, for example those involving more than 1000 categories. To conclude, current ML solutions do not provide a generic defense against adversarial examples.

### *B. Resilience to Preprocessing Attacks*

In this paper we performed an analysis of robustness of different types of adversarial examples against preprocessing attacks. We discovered that filtering attacks which try to remove the adversarial noise could be effective and could even remove the adversarial noise completely in specific domains (such as black and white images). Thus in order to use adversarial examples in CAPTCHA settings, one should improve the robustness of adversarial noise to filtering attacks.

### *C. Our Contribution*

This paper proposes DeepCAPTCHA – a new concept of CAPTCHA generation that employs specifically designed adversarial noise to deceive Deep Learning classification tools (as well as other ML tools due to transferability of adversarial examples [33]). The noise is kept small such that recognition by humans is not significantly affected, while resisting the removal attacks.

Previous methods for adversarial noise generation lack the robustness to filtering or any other attacks that attempt to remove the adversarial noise. We are the first to address this problem and we solve it by generating *immutable* adversarial noise with emphasis on image filtering. We analyze the security of our construction against a number of complementary attacks and show that it is highly robust to all of them.

Finally, we introduce the first proof-of-concept implementation of DeepCAPTCHA. Our results show that the approach has merit in terms of both security and usability.

## II. Related Work

We start our discussion with reviewing the most prominent work in CAPTCHA generation and then we turn to the Deep Learning area, focusing on methods for creating adversarial examples.

### *A. A Brief Introduction to CAPTCHAs*

Since their introduction as a method of distinguishing humans from machines [45], CAPTCHAs (also called inverse Turing tests [30]) have been widely used in Internet security for various tasks. Their chief uses are mitigating the impact of Distributed Denial of Service (DDoS) attacks, slowing down automatic registration of free email addresses or spam posting to forums, and also as a defense against automatic scraping of web contents [45].

Despite their utility, current CAPTCHA schemes are not considered popular by users as they present an additional obstacle to accessing internet services and many schemes suffer from very poor usability [5], [51].

*1) Text Based Schemes:* The first generation of CAPTCHAs used deformations of written text. This approach has now became less popular due to its susceptibility to segmentation attacks [50]. In response, some developers increased distortion levels, using methods such as character overlapping, which increases security [8]. Unfortunately, such measures have also resulted in schemes that are frequently unreadable by humans. We note that some text-based implementations are susceptible to general purpose tools [4].

*2) Image Based Schemes:* Motivated by the vulnerability of text based schemes, image based CAPTCHAs have been developed, following the belief that these were more resilient to automated attacks [10], [11], [13], [53]. For example, early text based versions of the reCAPTCHA [46] system were superseded by a combined text and image based approach. However, the new scheme was also subsequently attacked in [14].

An alternative approach is CORTCHA (Context-based Object Recognition to Tell Computers and Humans Apart) that claims resilience to machine learning attacks [53]. This system uses the contextual relationships between objects in an image, in which users are required to re-position objects to form meaningful groupings. This task requires a higher level reasoning in addition to simple object recognition.

*3) Alternative Schemes:* Considerable effort is currently being invested in novel ways of implementing secure and usable CAPTCHAs. Two of the most popular research themes are video-based CAPTCHAs such as NuCAPTCHA [32], and game-based CAPTCHAs [28]. The former have generally shown inadequate security levels so far [3], [49]. The latter designs are in general inspired by the AreYouHuman CAPTCHA [1]. One of the most interesting proposals in this group is [28], an example of a DGC (Dynamic Cognitive Game) CAPTCHA that has the additional advantage of offering some resistance to relay attacks, and a high usability. Unfortunately, in its current form, it is vulnerable to automated dictionary attacks. One can also argue that recent developments in game playing by computers, that match or improve human abilities by using deep reinforcement learning [27], question the prospects of future game based proposals. Finally, a number of puzzle-based CAPTCHAs that seemingly offered some promise have recently been subjected to devastating attacks [17].

*4) Deep Learning Attacks:* The general consensus within the cyber security community is that CAPTCHAs, that simultaneously combine good usability and security, are becoming increasingly hard to design due to potential threats from bots armed with Deep Learning [4], [14], [41] capabilities. This has led to the popularity of Google's NoCAPTCHA re-CAPTCHA

despite its violation of a number of important CAPTCHA and general security principles.[2]

*Definition of Secure CAPTCHA:* Different authors claim different security levels as the minimal standard for new CAPTCHA designs. In the literature we can find requirements for false positive rate (the probability of automatic bypassing of the CAPTCHA system) ranging from 0.6% to around 5%. Throughout this paper we define the threshold of at most 1.5% false positive rate to be the security requirement for a CAPTCHA.

### B. Deep Learning and Adversarial Examples

Deep Learning networks are designed to learn multiple levels of representation and abstraction for different types of data such as images, video, text, speech and other domains. Convolutional Neural Networks (CNNs) are DL algorithms that have been successfully applied to image classification tasks since 1989 [22]. Hereafter, we will use the terms CNN and DL network interchangeably.

*1) Foundations of Adversarial Examples:* Machine learning models are often vulnerable to adversarial manipulations of their input [9]. It was suggested [12] that this limitation could be expressed in terms of a distinguishability measure between classes. Using this measure, which is dependent on the chosen family of classifiers, they showed a fundamental limit on the robustness of low-capacity classifiers (e.g., linear, quadratic) to adversarial perturbations. It was also suggested that higher capacity models with highly non-linear decision boundaries are significantly more robust to adversarial inputs.

Deep neural networks were also shown to be vulnerable to adversarial perturbation. First examples of adversarial perturbations for deep networks were proposed in [42] as inputs, constructed by adding a small tailored noise component to correctly classified items that cause the DL network to misclassify them with high confidence.

Neural Networks can learn different capacity models, ranging from linear to highly non-linear. DL architectures are considered to have very large capacity, allowing highly non-linear functions to be learned. However, training such DL networks is hard and doing it efficiently remains an open problem. The only architectures (and activation functions) that are currently practical to train over complex problems have a piecewise linear nature which is the most likely reason for their vulnerability to adversarial examples [15].

Previous work [15], [35], [42] showed that adversarial examples generalize well across different models and datasets. Consequently, adversarial examples pose a security threat even when the attacker does not have access to the target's model parameters and/or training set [35].

*2) Constructing Adversarial Examples:* Different techniques for constructing adversarial inputs have been proposed in recent works. The approach in [31] causes a neural network to classify an input as a legitimate object with high confidence, even though it is perceived as random noise or a simple geometric pattern by a human. The techniques proposed in [15], [18], and [42] compute an image-dependent and small-magnitude *adversarial noise* component such that, when added to the original image, results in a perturbation that is not perceptible to the human eye but causes the DL network to completely misclassify the image with high confidence. The method in [34] focuses on making the adversarial noise affect only a small portion of the image pixels, but the noise itself could be larger than in previous methods. In contrast to the approach of targeting the prediction of the classifier (as discussed above), Sabour et al. [38] proposed adversarial examples that change a hidden representation of the network, making it very close to an example with a different label. Miyato et al. [26] considered a different setting in which labels of images are unavailable. This approach targeted the posterior distribution of the classifier corresponding to small perturbations of the original image.

*3) Robustness to Adversarial Examples:* Previous work [12], [15] outlined a number of solutions for adversarial instability. One of them was to switch to highly non-linear models, for instance, RBF Neural Networks or RBF Support Vector Machines. These are shown to be significantly more robust to adversarial examples, but are currently considered impractical to train for the large-scale problems (for example 1000-way categorization).

Improving the robustness of DL tools against adversarial perturbations has been an active field of research since their discovery. The first proposition in this direction was to train DL networks directly on adversarial examples. This made the network robust against the examples in the training set and improved the overall generalization abilities of the network [12], [15]. However, it did not resolve the problem as other adversarial samples could still be efficiently constructed. A method called defensive distillation was proposed in [36]. This approach provides a high level of robustness, but only against a very specific type of adversarial noise (see [36] for details.) Gu and Rigazio [16] trained an autoencoder to predict the original example from the one with adversarial perturbations. However, the combination of such an autoencoder and the original network was shown to be vulnerable to new adversarial examples, specially crafted for such combined architectures. Moreover, such combinations increase the classification time. Hence, this approach seems to be somewhat unsuitable for computer security applications, where efficiency and versatility are crucial. To conclude, the current state of technology does not offer a solution for a large scale (+1000 categories) multi-class recognition problem that is robust to adversarial examples. Moreover, it was shown that adversarial examples are consistently difficult to classify across different network architectures and even different machine learning models (e.g, svm, decision trees, logistic regression, KNN classifier) [15], [33], [35], [42].

These limitations, combined with the fact that adversarial noise could be made almost imperceptible to the human eye, render the idea of using adversarial examples as the basis for new CAPTCHA challenges very appealing. However, in order to use adversarial noise in CAPTCHAs or other security applications, it should be resistant to removal attacks which can

[2]For example, the **P** in CA**P**TCHA stands for Public, and NoCAPTCHA inner functioning is not public, based on the time-dishonored concept of "security by obscurity" by employing heavily obfuscated Javascript code.

employ alternative tools and, in particular, image processing methods. We show here that none of the existing methods for adversarial example construction are sufficiently robust to such attacks. Even though the approach in [34] proposed the construction of adversarial examples in a computer security context, it also lacks the necessary robustness.

## III. Test Bed Details

We have used two sets of problems for the adversarial examples discussed in the paper. The first one is the MNIST database for digits [23]. The second, which was used in the majority of our experiments, is the ILSVRC-2012 database [37], containing 1000 categories of natural images. We used MatConvNet [44] implementations of CNN for MNIST classification and of CNN-F deep network [7] for object classification.

We crafted adversarial examples using these two DL networks. The networks were trained on the training set of the corresponding database. The adversarial examples were created using the validation (test) set.

All experiments described in this work were conducted on a Linux 3.13.0 Ubuntu machine with an Intel(R) QuadCore(TM) i3-4160 CPU @ 3.60GHz, 32GB RAM, with a GTX 970 GPU card, using MATLAB 8.3.0.532 (R2014a.)

## IV. Immutable Adversarial Noise Generation

Adversarial noise is specifically designed to deceive DL networks. However, an attacker can preprocess network inputs in an attempt to remove this adversarial perturbation. Hence, in a computer security setting, adversarial noise must withstand any general preprocessing technique aimed at cancelling its effects.

We introduce the concept of *Immutable Adversarial Noise* (IAN), as an adversarial perturbation that withstands these cancellation attempts. We explicitly define the requirements for creating IAN that are useful for CAPTCHA generation. Then, we analyze previous algorithms for adversarial example generation and show that they do not meet these requirements. Finally, we present our new scheme for IAN generation, that satisfies these new requirements.

### A. Requirements for IAN in CAPTCHA

An algorithm for the construction of immutable adversarial noise useful in CAPTCHA generation needs to meet the following requirements:

1) **Adversarial:** The added noise should be successful in deceiving the targeted system according to the defined security level (specifically to our requirement, at least 98.5% of the time.)
2) **Robust:** The added noise should be very difficult to remove by any computationally efficient means; for example by filtering or by ML approaches.
3) **Perceptually Small:** The added noise should be small enough to not interfere with a successful recognition of the image contents by humans.
4) **Efficient:** The algorithm should be computationally efficient, to allow for the generation of possibly millions of challenges per second. This is fundamental for deploying the CAPTCHA successfully in production environments.

A basic requirement for CAPTCHAs is that challenges do not repeat and are not predictable (i.e., guessing one out of $m$ possible answers should succeed with probability $1/m$). Hence, the source used for generating adversarial examples should be bottomless and uniform. An algorithm that can create an adversarial example from an arbitrary image, together with such a bottomless and uniform source of images can certainly generate a bottomless and uniform set of challenges, as required.

### B. Previous Methodologies for Generating Adversarial Examples

We briefly introduce in the following the most popular methods for adversarial noise generation, and discuss why they do not meet the above requirements.

Our idea is simple: use images that are easily recognized by humans but are adversarial to DL algorithms. Consequently, methods that cause a DL network to classify images of noise or geometric patterns as objects such as the one in [31] are not adequate for our goal.

We analyzed in detail the optimization method proposed in [42] and the fast gradient sign method suggested in [15]. These two methods for constructing adversarial perturbations are the most mature, and have been previously considered in the literature when exploring countermeasures against adversarial examples. We believe that methods such as [18] and [34] will show a similar behavior, as they rely on the similar concept of adding a noise component to the original image. The methods in [26] and [38], on the other hand, considered a completely different setting which may be useful for future works.

To exemplify the removal of adversarial noise, we focus on two classical image classification tasks: digit recognition and object recognition. Digit recognition was used in earlier generations of CAPTCHA, but, as we show later, object recognition provides a significantly better base for our proposal, both from security and usability points of view.

*1) The Optimization Method:* Szegedy et al. [42] introduced the concept of adversarial examples in the context of DL networks and proposed a straightforward way of computing them using the following optimization problem:

$$\arg\min_{\Delta_I} \|\Delta_I\|^2 \quad s.t.\ Net(I + \Delta_I) = C_d \tag{1}$$

where $I$ is the original input from class $C_i$, $\Delta_I$ is the adversarial noise, $Net$ is the DL classification algorithm, and $C_d$ is the deceiving class, such that $C_d \neq C_i$. Once the adversarial noise is computed, the corresponding adversarial image is constructed by adding the adversarial noise to the original input $I$.

We implemented and tested the optimization method described by Eq. (1), over a set of 1000 images on the MNIST and ILSVRC-2012 datasets.

Fast computation of adversarial examples is an essential requirement for any viable CAPTCHA deployment, since it will need to generate millions of challenges per second.

TABLE I

COMPARISON BETWEEN ADVERSARIAL NOISE GENERATION METHODS. REPORTED TIMES SHOW THE EFFICIENCY OF THE GENERATION ALGORITHM; ADVERSARIAL SUCCESS INDICATES THE PERCENTAGE OF EXAMPLES THAT SUCCEEDED TO FORCE THE TARGET DL NETWORK TO CLASSIFY THE ADVERSARIAL EXAMPLE WITH THE TARGET CATEGORY (CHOSEN AT RANDOM); RESISTANCE LEVEL INDICATES THE PERCENTAGE OF ADVERSARIAL INPUTS (OUT OF 1000) THAT **WERE NOT** REVERTED TO THEIR ORIGINAL CATEGORY BY APPLYING THE BEST PERFORMING PREPROCESSING METHOD (THRESHOLDING FOR THE MNIST SET AND MEDIAN FILTER OF SIZE 5×5 FOR THE ILSVRC-12 SET), WHERE BIGGER NUMBERS REPRESENT BETTER PERFORMANCE

| Method | MNIST (digits) | | | | ILSVRC-2012 (images) | | | |
|---|---|---|---|---|---|---|---|---|
| | Generation Time | | Adversarial | Resistance | Generation Time | | Adversarial | Resistance |
| | Average | Std | Success | Level | Average | Std | Success | Level |
| Optimization [42] | 3.83s | 2.87s | 20% | 5% | 120.94s | 98.19s | 85.2% | 83.8% |
| Fast Gradient Sign[3] [15] | 0.0677s | 0.0268s | 100% | 2.4% | 0.33s | 0.03s | 97.8% | 39.19% |
| Our IAN Scheme | Unlikely to exist, see discussion. | | | | 1.01s | 0.80s | 100% | 100% |

The optimization method described above is unfortunately too slow, hence for practical purposes we limited the number of iterations to a fixed threshold (it stops when this limit is reached). This, however, resulted in a failure to produce the desired class in some cases. The timing statistics of the experiment and the success rate are shown in Table I.

Based on these results, we can conclude that the optimization algorithm is not suitable for our needs: it is computationally expensive and it does not converge in some cases. The inefficiency of this method has been reported before, and is explicitly mentioned in [15] and [42].

Despite its poor efficiency, we analyzed the resistance to preprocessing attacks of the adversarial noise created by the optimization method. We computed 1000 adversarial images (as described above) for the MNIST and ILSVRC-2012 datasets. Then we tested various filters and parameters and found that for the MNIST data set, which is exclusively formed of images of white digits on a black background (just two intensity values, 255 and 0), the adversarial noise can be successfully removed by applying a half range threshold (128) on the pixel values. This is an extremely simple and fast procedure that cancels the adversarial effect in **95%** of the tested images.

It was also noted in [16] that applying a convolution with a Gaussian kernel of size 11 to the input layer of the CNN trained on MNIST, helps in classifying correctly 50% of the adversarial examples created by the optimization method. We achieved an even better result of 62.7% with a median filter of size 5x5, but both of these results are way below the success of the much simpler thresholding method.

These findings demonstrate that images composed of only two colors (such as those in MNIST) are a poor source of adversarial examples.

Canceling adversarial noise in natural RGB images is more challenging, and can not be generally achieved by a simple thresholding. We found that a 5x5 pixel median filter was most successful in removing adversarial noise from the images drawn out of the ILSVRC-2012 data set. Note that for an attack on a CAPTCHA to be successful, it is important for the machine classification to match human classification accuracy. Thus the classification of the filtered samples should be compared to the true label (rather than to the classification label of the original input, that could be wrong some times).

The classification label computed by the network on filtered adversarial examples constructed from the ILSVRC-2012 dataset matched their true label in 16.2% of the cases (see Table I). This result is not very high, but it shows that it fails by quite some margin to provide the required security level for modern CAPTCHAs.

*2) The Fast Gradient Sign Method:* A much faster method for generating adversarial examples was proposed in [15]. The approach is called the *fast gradient sign* method (FGS) and it computes the adversarial noise as follows:

$$\Delta_I = \epsilon \cdot sign(\nabla_I J(W, I, C_i))$$

where $J(W, I, C_i)$ is a cost function of the neural network (given image $I$ and class $C_i$), $\nabla_I J(W, I, C_i)$ is its gradient with respect to the input $I$, $W$ are the trained network parameters, and $\epsilon$ is a constant which controls the amount of noise inserted. Similarly to [42], the adversarial image is obtained by adding the adversarial noise $\Delta_I$ to the original image $I$.

The FGS method does not produce adversarial examples that deceive the system with the chosen *target label*. Its goal is simply to change the classification of the adversarial example, away from the original label. This is done by shifting the input image in the direction of the highest gradient by a constant factor. The bigger this constant is, the larger both the adversarial effect and the degradation of the image are. The FGS method is significantly faster than the previous optimization approach (Table I.)

The adversarial noise for the MNIST dataset produced by the FGS method was also easily removed by thresholding the pixel values at the half range threshold, achieving an even higher success of 97.60% (out of 1000 images).[4] An example of FGS adversarial noise removal from an MNIST sample is shown in Figure 1.

For the object recognition task (over the ILSVRC-2012 dataset) the FGS succeeded in creating an adversarial example 97.8% of the time, as shown in Table I. Unfortunately, the median filter (of size 5x5) was able to restore the classification of the adversarial examples to their true label in a staggering 60.81% of the cases, deeming this method unusable for our purposes.

[4]A median filter of size 5x5 succeeded in removing the adversarial noise in 55.20% of the cases.

(a) (b) (c) (d) (e)

Fig. 1. Adversarial example on MNIST: The original black and white image is shown in (a), an adversarial example, produced by the FGS method is shown in (b) including gray-level intensities. (c) shows the corresponding adversarial noise changing the classification into an 8. This noise is successfully removed by thresholding (the resulting image is depicted in (d)) as well as by a median filter of size 5x5 (whose result is shown in (e).)

Since the FGS method does a very small step away from the correct label, the resulting noise 1) has a very small magnitude and 2) the dependency of the noise on the source image is low and it is similar in appearance to "salt and pepper" noise. These properties explain the success of the median and other standard filters.

### *C. IAN: Our New Approach to Adversarial Noise*

As we showed earlier, adversarial noise can be easily removed from black and white images, thus MNIST is clearly not a good dataset for creating immutable adversarial examples, despite the existing literature. We therefore choose natural images of objects, containing richer variations in color, shapes, texture, etc. as the platform for IAN construction.

We base our method for adversarial noise construction on the FGS method, since CAPTCHA applications requires a very fast computation of adversarial examples, and speed is one of the main advantages of this algorithm. However, the FGS lacks two important properties. First, it only perturbs the original label of the classification, but it offers no guarantees that the new label would be semantically different from the original one. Our experiments on ImageNet with 1000 categories showed that the FGS changes the label to semantically similar classes in many cases (for example, hand held computer to cellular telephone). This can have seriously effects on the usability, and possibly the security, of the CAPTCHA system.

We propose an iterative version of FGS that accepts in addition to the original image also a target label and a confidence level and guarantees that the produced adversarial example is classified with the target label and at the desired confidence (while keeping the added noise minimal).[5] To this end, we run a noise generation step with a small $\epsilon$ in the direction that increases the activation of the target label (as specified in Eq. 2) for several iterations until it reaches the target label and the desired confidence level. We call this method an iterative fast gradient sign (IFGS). To increase the activation of the target label we update the input image $I$ as follows:

$$I = I - \epsilon \cdot sign(\nabla_I J(W, I, C_d)) \tag{2}$$

The second property that FGS lacks is resilience to filtering attacks. As discussed in Section IV-B.2 the success of standard filtering stems from the small magnitude of the noise. Increasing the magnitude of the noise would significantly damage the content of the image resulting in poor human recognition. Moreover, we observed that FGS noise tends to be stationary over the image, which facilitates standard filtering. To resolve these problems, we suggest the following approach: Our construction for the generation of immutable adversarial noise starts with an adversarial image, produced by the IFGS, with a small noise constant $\epsilon$. It then filters the adversarial image and tries to recognize it. If it succeeds, then we increase the noise and run IFGS with the new noise constant. We iterate the process until the noise cannot be removed. The iterative approach guarantees that the increase in magnitude does not exceed the desired level (allows for easy human recognition). In addition running IFGS several times increases the dependency of the noise on the source image, preventing standard filters from removing it. We detail the construction in the pseudocode shown in Algorithm 1.

A median filter of size 5x5 was used in our construction, as it showed experimentally the highest success in removing the adversarial noise generated by the fast gradient sign method when compared with other standard filters such as the average, Gaussian lowpass and Laplacian, and was faster than more complex filters such as non-local means [2] and wavelet denoising [25].

**Algorithm 1** IAN_Generation

**Require:** $Net$ a trained DL network; $I$ a source image; $C_i$ is the true class of $I$; $C_d$ a deceiving class; $p$ a confidence level of the network; $M_f$ a Median filter.
**Begin:**
$adv(I, C_d, p) \leftarrow I$; {$adv(I, C_d, p)$ *the adversarial example*}
$\Delta \leftarrow 0$;
**while** $Net(M_f(adv(I, C_d, p))) = C_i$ **do**
  **while** $Net(adv(I, C_d, p)) \neq C_d$ **or** confidence $< p$ **do**
    $\Delta = -\epsilon \cdot sign(\nabla_I Net(I, C_d))$;
    $adv(I, C_d, p) \leftarrow adv(I, C_d, p) + \Delta$;
  **end while**
  $\epsilon = \epsilon + \delta_\epsilon$; {Increase the noise constant;}
**end while**
**Output:** $\Delta$

We tested the proposed method on the same set of 1000 images with initial value of $\epsilon = 5$ and $\delta_\epsilon = 5$. The evaluation results, shown in Table I, prove that our method for IAN generation satisfies all four requirements, as defined in Section IV-A. It is important to note that the additional checks

[5] A similar algorithm was independently discovered in [20].

to ensure robustness against the median filter $M_f$ do not slow down the generation process significantly.

Figure 2c shows an example of an adversarial image, created by adding the IAN (Figure 2b) produced by our novel algorithm to the original image (Figure 2a). Figure 2d depicts the outcome of applying the median filter to the adversarial image. The resulting image is not recognized correctly by the DL network. Moreover, the filtering moved the classification to a category which is further away (in terms of the distance between the class positions in the score vector) from the true one. The distance between the true and deceiving classes is 214, and between the true class and the class assigned to the image after filtering (a removal attack) is 259. At the same time, while being more noticeable than in the previous algorithms, the relatively small amount of added noise still allows a human to easily recognize the image contents.

## V. DeepCAPTCHA

We now propose a novel CAPTCHA scheme that we named DeepCAPTCHA, which is based on a large-scale recognition task, involving at least 1,000 different categories. The scheme utilizes a DL network, trained to recognize these categories with high accuracy.

DeepCAPTCHA presents an adversarial example as an image recognition challenge. The adversarial example is obtained by creating and adding IAN to its source image. The deceiving class in IAN must differ from the true class of the source image (both classes are from the 1,000 categories involved in the recognition task). The source image is chosen at random from a very large (bottomless) source of images with uniform distribution over classes, and discarded once the adversarial image is created. The label of the image is obtained by classifying it using the deep network and verifying that the top score is over a predefined confidence level.

Contrary to previous CAPTCHAs that use letters or digits, we use objects in order to make the classification task larger and to provide enough variability in the image to make it robust to attacks that aim to remove the adversarial noise. Using object recognition as a challenge poses two usability issues: 1) object names are sometimes ambiguous, 2) typing in the challenge solution requires adapting the system to the user's language. We propose to solve these issues by providing a set of pictorial answers, i.e., a set of images, each representing a different class. Obviously, the answers contain the correct class, as well as random classes (excluding the deceiving class).

The task for the user is to choose (click on) the image from the supplied set of answers that belongs to the same class as the object in the test image – the adversarial example. Since we keep the adversarial noise small, a human could easily recognize the object in the adversarial example and choose the correct class as the answer. The only possible ML tool that can solve such a large-scale image recognition problem is a DL network. However, the adversarial noise used to create the adversarial example is designed to deceive the DL tools into recognizing the adversarial image as from a different category. Hence, the proposed challenge is designed to be easy for humans and very difficult for automatic tools.

### A. The Proposed Model

We now provide a formal description of our proposed design. Let $Net$ be a DL network trained to classify $n$ ($n \geq 1000$) classes with high (human-competitive) classification accuracy. Let $C = \{C_1, ..., C_n\}$ be a set of labels for these $n$ classes. Let $I$ be an image of class $C_i \in C$. Let $C_i^* = C \backslash \{C_i\}$,[6] and let $C_d$ be a deceiving label which is chosen at random from $C_i^*$. The DeepCAPTCHA challenge comprises the following elements:

- An adversarial image $adv(I, C_d, p)$, constructed from $I$ by the addition of an immutable adversarial noise component (constructed by Algorithm 1) that changes the classification by the DL $Net$ to class $C_d$ with confidence at least $p$.[7]
- $m-1$ answers, which can be fixed images corresponding to $m-1$ labels chosen at random and without repetition from $C_i^* \backslash \{C_d\}$;
- A fixed image with label $C_i$, different from $I$.

The $m-1$ suggestions and the true answer are displayed in a random order. The challenge for the user is to choose the representative image of $C_i$ from the answers. The original image $I$ is assumed to be a fresh image which is randomly picked from different sources (databases and/or online social networks), and it is **discarded** after creating the adversarial example (i.e., we never use the same source image twice).

The pseudocode for the DeepCAPTCHA challenge generation is shown in Algorithm 2 and an example, generated by our proof-of-concept implementation (detailed in Section VII), is depicted in Figure 3.

**Algorithm 2** Compute a DeepCAPTCHA challenge

**Require:** $[C_1, \ldots, C_n]$ a set of $n$ classes; $\{I_j\}_{j=1}^n$ fixed answers of the $n$ classes; $i \leftarrow_r [1, 2, \ldots, n]$ the index of a random class $C_i$, and $I \in_R C_i$ a random element; $m$ the number of possible answers; $p$ the desired misclassification confidence; $Net$ a trained DL network; $M_f$ a Median filter.
**Begin:**
Randomly pick a destination class $C_d, d \neq i$;
Set $\Delta$ = IAN_Generation($Net$, $I$, $C_i$, $C_d$, $p$, $M_f$);
$adv(I, C_d, p) = I + \Delta$; {The immutable adversarial example}
Discard I;
Randomly select $m-1$ different indexes $j_1, \ldots, j_{m-1}$ from $[1, \ldots, n] \backslash \{i, d\}$;
Choose the representative images $[I_{j_1}, \ldots, I_{j_{m-1}}]$ of the corresponding classes;
**Output:** $adv(I, C_d, p)$, and a random permutation of $m$ possible answers $\{I, I_{j_1}, \ldots, I_{j_{m-1}}\}$.

## VI. Security Analysis

In the following we analyze several different but complementary approaches that potential attackers could use against

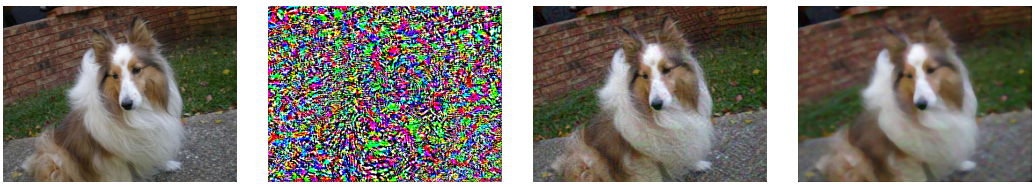
[6] We note that in some cases, depending on the variability of the data set and other circumstances, it could be advisable to remove classes *similar* to $C_i$ from $C_i^*$.

[7] In our experiments we have used $p = 0.8$.

TABLE II

FILTERS EMPLOYED IN THE FILTERING ATTACK, AND THEIR RESPECTIVE SUCCESS RATES (OUT OF 1000 TRIALS). NOTE THAT THE MEDIAN FILTER WAS USED IN THE GENERATION PROCESS, THUS THE CHALLENGE IS ROBUST TO THE MEDIAN FILTER BY CONSTRUCTION

| Median (Size 5x5) | Averaging (Size 5x5) | Circular Averaging (Radius 5) | Gaussian Lowpass (Size 5x5, std = 0.5) | Laplacian (Size 3x3, $\alpha = 0.2$) | Non-local Means (1/2 Patch size = 3, 1/2 Window size = 2, Weighting = 0.1) | Wavelet Denoising ($\sigma = 3$, Num. levels = 3) |
|---|---|---|---|---|---|---|
| 0% | 0% | 0.1% | 0.1% | 0.2% | 0.3% | 0.2% |



(a) (b) (c) (d)

Fig. 2. An example of the IAN generation algorithm. Image (a) is the original image, correctly classified as a Shetland sheepdog with a high confidence of 0.6731 (b) is the computed immutable adversarial noise, (c) is the adversarial image (the sum of the image in (a) and the IAN in (b)), classified as a tandem or bicycle-built-for-two with a 0.9771 confidence and (d) the result of applying $M_f$, classified as a chainlink fence with confidence 0.1452.

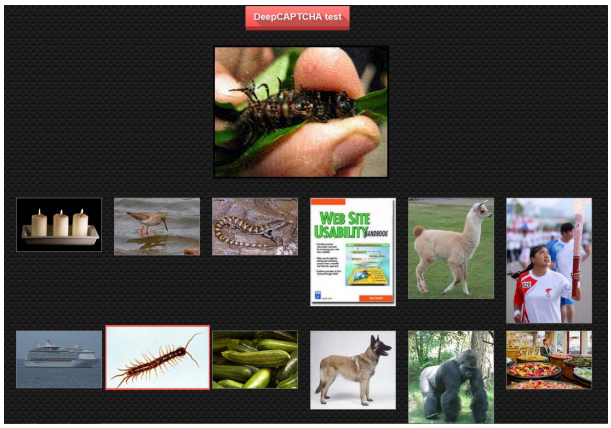



Fig. 3. An example of a DeepCAPTCHA challenge. The large image above is the computed adversarial example, and the smaller ones are the set of possible answers.

the proposed DeepCAPTCHA system. We start the analysis by discussing a straightforward guessing attack, we then continue to evaluate attacks that use image processing techniques, aiming to revert the adversarial image to its original class by applying image processing filters. We then turn to more sophisticated attacks that employ machine learning tools. Finally, we discuss possible solutions to relay attacks. We set the security requirement for the success of an attack to a false acceptance rate (FAR) of at most 1.5%.

### A. Random Guessing Attack

Using $m$ answers per challenge provides a theoretical bound of $(\frac{1}{m})^n$ for the probability that a bot will successfully pass $n$ challenges.[8] Therefore, $n = \frac{-\log p}{\log m}$ are required for achieving a False Acceptance Rate (FAR)[9] of $p$. As we show later (in Section VII-A), $m = 12$ offers sufficient usability (low False Rejection Rate (FRR) and fast enough answers), hence for our target FAR of at most 1.5%, $n$ should be greater than 1.67, e.g., $n = 2$ (resulting in an FAR of 0.7%).[10]

One can, alternatively, combine challenges with different numbers of answers in consecutive rounds, or increase $n$. These allow a better tailoring of the FAR and the FRR (both can be computed following the figures shown on Table IV). The latter approach offers a finer balance between security and usability.

### B. Filtering Attacks

We examined the robustness of our IAN generating algorithm to a set of image filters particularly aimed at removing the added noise. Any of these attacks will succeed if they are able to remove sufficient noise to correctly classify an adversarial example into the class of its original image.

We tested seven filters with a wide range of parameters on a set of 1000 adversarial examples, created with the generation algorithm presented in Algorithm 1. This set of filters included the median filter, averaging filter, circular averaging filter, Gaussian lowpass filter, a filter approximating the shape of the two-dimensional Laplacian operator, non-local means [2], and wavelet denoising [25] filters. Table II shows the success rates of the different filters (along with the optimal parameter choice for the filter). The success rates of all filters are significantly below the security requirement of 1.5%.

[8]Assuming independence between tests.

[9]In our context, FAR stands for the probability that a bot succeeds to pass the DeepCAPTCHA whereas FRR stands for the probability that a human fails to pass DeepCAPTCHA.

[10]We note that increasing the permissable FAR to 1.5625% would allow using two challenges of 8 answers each. This will improve the usability of the system as shown in Section VII-A. However, we prioritize the security and thus choose 12 answers for challenge.

### C. Machine Learning Based Attacks

We start by defining the attacker model, and then analyze in-depth the most prominent attacks that could be applied against DeepCAPTCHA.

*1) The Attacker Model:*

**Knowledge of the algorithm and its internal parameters:** The attacker has a full knowledge of the CNN (its architecture and parameters, or knowledge of the training set that allows training of a similar CNN), used in the adversarial noise generation algorithm and of the generation algorithm itself, including its internal parameters.

**Access to DeepCAPTCHA challenges:** The attacker has access to all generated adversarial examples (but not to their source) as well as to the images which serve as the representatives of the classes (one or more per class).

**No Access to the Source Images:** The source images (used to generate the adversarial examples) are chosen at random from crawling a number of high volume online social media and similar sites, thus the size of the source image pool can be considered infinite for all practical purposes. Once the adversarial image is created, the corresponding original image is discarded instantly from the DeepCAPTCHA and never reused.[11] Theoretically, the attacker may access the sources as well (or have access to an indexing service such as Google), but if the chosen image is "fresh" (not indexed yet) and chosen from a large set of sources, he has no knowledge about the particular image used for generating the adversarial example.

**Access to other machine learning tools:** The attacker has the ability to use any other classifier in an attempt to classify the adversarial examples or to train the same or other DL networks on them. This can be done with the aim of finding alternative networks with similar accuracy over the baseline classification problem, but having more robustness against adversarial examples.

Therefore, in the highly likely case that the attacker does not have access to the source images, the DeepCAPTCHA scheme is secure and usable even when all other aspects of the attacker model are satisfied.

*2) Alternative Classifier Attack:* The most straightforward attack on DeepCAPTCHA is probably the one that tries to use other classifiers, in an attempt to correctly recognize the adversarial example.

A machine learning algorithm, to be used successfully in such an attack, should be 1) robust to adversarial examples in general or at least to those used in DeepCAPTCHA; 2) scalable to a large number of categories (+1000).

[11]Obviously, unless the system stores all previous source images, then repetitions may exist by random chance (depending on the sampling process). Designing efficient methods to ensure that no such repetitions exist, is outside the scope of this paper.

TABLE III

TRANSFERABILITY OF ADVERSARIAL EXAMPLES TO OTHER DEEP NETWORKS. 1000 ADVERSARIAL EXAMPLES WERE CREATED USING CNN-F FROM CLEAN IMAGES IN THE VALIDATION SET OF ILSVRC-12. THE TABLE REPORTS THE CLASSIFICATION ACCURACY OF THE TESTED NETWORKS ON THE CLEAN AND ADVERSARIAL VERSIONS OF THESE IMAGES

| Network | Clean Images | Adversarial Images |
|---|---|---|
| CNN-F | 55.2% | 0% |
| CNN-M | 58.7% | 0.1% |
| CNN-S | 60.5% | 0.1% |
| AlexNet | 55.6% | 0% |

Highly non-linear models such as RBF-SVM, or RBF networks are known to be more robust to this adversarial phenomenon [12], [15]. But these non-linear classifiers are currently not scalable to be able to cover +1000 categories. Thus, they do not offer a practical method for breaking DeepCAPTCHA or future similar schemes until major breakthroughs in ML allow for training highly non-linear models over problems with a large number of classes.

Since the adversarial generation algorithm uses a specific network, one can consider a potential attack using another DL network with a different architecture and/or parameterization. However, it was previously shown that adversarial examples generalize well to different architectures and initializations [15], [24], [42].

To verify the robustness of our construction against attacks that use alternative DL algorithms, we tested several publicly available DL networks trained on the same set of images to classify the adversarial examples in DeepCAPTCHA. Specifically, we used the CNN-F network from [7] to generate the CAPTCHA and we tested the ability to recognize the adversarial examples using three other deep learning networks. Two of these networks have a different architecture: CNN-M is similar to Zeiler and Fergus [52] and CNN-S is similar to OverFeat [39]. The third network — AlexNet from [19], has an architecture similar to CNN-F, with the difference that CNN-F has a reduced number of convolutional layers and a denser connectivity between convolutional layers. Table III compares the classification results of the tested networks on the clean and adversarial version of 1000 images. The results show that none of these tools reached the 1.5% threshold.

We verified the transferability of adversarial examples, created by Algorithm 1, across different classification models. Specifically, we tested a linear SVM classifier trained on a bag of visual words (as provided in the toolkit of ILSVRC-10 competition) extracted from the training set of ILSVRC-12 and tested their classification rate on the clean validation set of ILSVRC-12 and on the adversarial set used to test deep networks. This model achieved 23.99% classification rate on the clean validation set (which is comparable to the results reported in the literature). However, the classification rate on the adversarial set achieved only 0.5% which is below our security threshold.

*3) Adversarial Training:* Since adversarial examples are effective at fooling other ML tools trained on clean examples, another attack to consider is fine tuning an existing DL network on adversarial examples. Previous work suggested to improve the robustness of DL to adversarial examples by an iterative process that alternates between creating adversarial examples for a current network and fine-tuning this network on a mixed set of regular and adversarial examples [42]. This approach was only tested on the MNIST data set, which is relatively small. Still, the reported results show very limited success. Namely, running five iterations of training on adversarial examples improved the error rate from 96.1% to 92.1% [43]. Combining networks produced during the iterative adversarial training into a committee and taking an average prediction of all committee members as the score for classification, improved the error to 35.5% [43]. Finally, Goodfelow et al. [15] suggested adversarial training which combines the loss of a training sample with the loss of its adversarial variant. Such training reduced the error on adversarial examples generated via the original model to 19.6%.

Even though the new networks enjoy a somewhat increased robustness to adversarial examples, generated by the network prior to adversarial training, one can easily generate adversarial examples against the retrained networks (as noted by Warde-Farley and Goodfellow [47]). Moreover, the results of adversarial training were reported for the MNIST set, which is composed of images with low entropy, and thus the adversarial noise could be easily neutralized by very simple tools (See section IV). Previous quantitative results for natural images categories are restricted to one-step methods, which improve their robustness as a result of adversarial training [21]. However, as we showed in Section IV-B, adversarial noise created by one-step methods can also be removed with high success using a very simple filtering.

An adversary wishing to use adversarial training for attacking the DeepCAPTCHA system would need to obtain adversarial examples which are correctly labelled (the ground truth labels). Given the success of deceiving the existing network, this would force the adversary to employ humans to provide the labels (at a higher cost). Moreover, the DeepCAPTCHA system can also be retrained (to imitate the process done by the adversary) to produce new adversarial examples against the newly trained network. As the DeepCAPTCHA system knows the true labels, the defender has the upper hand — for a smaller cost and effort she can alter her network to imitate the adversary.

We ran an instance of an adversarial training attack on DeepCAPTCHA using the ILSVRC-2012 database [37]. We used a two step process that iteratively improves the robustness of the input network by adversarial training. The process inputs CNN-F network [7] (used for DeepCAPTCHA) fully trained on clean examples. The first step of the training process produces adversarial examples using our novel algorithm (see Algorithm 1) via the current network and adds them to the pool of adversarial examples. The second step fine tunes the current network on the mix of clean and adversarial examples from the updated pool. We ran the process for 5 iterations adding 2000 adversarial examples in each iteration. After training, the error on a validation set constructed from the previous generations of adversarial examples was reduced to 92.3% on average (over the intermediate networks) with a minimal error of 87%. However, newly produced adversarial examples deceive all these networks in 100% of the cases.

Building a committee from the intermediate networks (produced by the iterative training process) achieved a minor reduction in error on the newly generated adversarial examples (crafted via the last version of the network). Specifically, basing the classification on the sum of scores of all committee members was able to reduce the error from 100% to 98.9%. Devising adversarial examples that can deceive multiple models has been recently shown in [24]. Similar strategies can be followed to improve robustness against a committee of classifiers.

To conclude, it seems that adversarial training attacks against DeepCAPTCHA can be mitigated by retraining the network (periodically) on previously created adversarial examples.

*4) Noise Approximation Attack:* Given that the challenges were generated by adding adversarial noise, the attacker may hope to approximate this noise (to remove it) using DL. We show next that for suitably chosen image sources, this attack is successful less than 1.5% of the time.

Recall that the images belong to known classes. Therefore, the attacker can try and explore the similarity between images of the same class in order to approximate the noise that changes the classification from the true category ($C_i$) to the deceiving one ($C_d$). To approach this goal one would consider collecting representative samples of a category and learning a noise per each sample in that class and for each other category in the system.

For the attack to be effective, the variation between the instances of the same class should be small, for example a category comprising images of the letter 'A' printed with a similar font. In other words, the adversarial noise that takes an element from $C_i$ and "transforms" it into an element in $C_d$, should be relatively independent of the actual element.

Fortunately, this property rarely holds for general object categories like the ones we are using for DeepCAPTCHA. In fact, this is exactly what causes the baseline classification to be difficult in the first place, requiring a sophisticated feature extraction process (such as a CNN) to overcome the very high intra-class variation.

Along these lines, we implemented and tested an attack we have named the *noise approximation attack*. Consider a working example with the following settings: a thousand image categories, where each category is represented by 1200 images[12] and there are 12 candidate answers per DeepCAPTCHA challenge. If the images used for answers are static, then their labels could be pre-computed by running the network over all classes only once. Then, for each challenge, the labels of the answers could be retrieved very efficiently.

In the pre-computation step, the attacker can compute the adversarial noises that transform every image in the dataset

[12]To make the CAPTCHA more secure, we chose classes with large variability between the categories.

into every other category. This implies a total of $1,200*999 = 1,198,800$ adversarial noise images (i.e., for a representative image $I' \in C_i$ and a target category $d$ compute all its $\Delta^d_{I',i} = I' - adv(I, C_d, p)$ values).

In the online phase of the attack, the attacker is presented with the challenge, including the adversarial example[13] $adv(I, C_d, p)$ and a random permutation of 12 possible answers $\{I_i, I_{j_1}, \ldots, I_{j_{11}}\}$ (where $i$ is the label of the correct class, and $d$ is the decoy label of the adversarial example). Then, the attacker runs the network over $adv(I, C_d, p)$ and retrieves the decoy label $d$. As the attacker knows that the noise caused the image $I$ to be classified in $C_d$ (rather than one of the 12 classes represented by the set of answers), he tries to remove the adversarial noise that transforms $I' \in C_i$ into $C_d$ from $adv(I, C_d, p)$. Specifically, for each class $j$ of the 12 answers, and for each representative image $I' \in C_j$, the attacker computes the estimation of the original image as: $I^* = adv(I, C_d, p) - \Delta^d_{I',j}$, and then runs the network on the estimate $I^*$, which results in $1200 * 12 = 14400$ attempts per challenge (as the representative sets are of size 1200 images, and there are 12 candidate sets). This is a large number, but if the images in the same category are very similar (e.g., same letter), then even the first attempt could be successful. To prevent such security issues one should exclusively use natural images of real objects with moderate to high intra-class variation as a source for CAPTCHA generation.

We ran an instance of the noise approximation attack, where the true category was *lion* (that exhibits moderate intra-class variation) and the target category was *rooster*. A total of 3 out of 1200 challenges were broken using this approach. This implies that the noise approximation attack is interesting and potentially relevant, and that despite its low success rate of 0.25% it needs to be taken into account in future implementations, to ensure it stays below the 1.5% threshold.

We also verified that categories with a low inner-class variability are highly susceptible to the proposed noise approximation attack. Specifically, we used MNIST data set to collect adversarial noises that cause CNN to classify images of digit '1' as '2' for a set of 200 adversarial examples. We tested the noise approximation attack using these noises on a different set of 200 adversarial examples (of '1' recognized as '2'). The attack succeeded to remove the adversarial noise in all tested images. Furthermore, it was very effective computationally, as it succeeded to remove the adversarial noise from a test image on the first attempt (subtracting the first stored noise) 90% of the time. Consequently, MNIST and similar low variation data collections are not a suitable source for adversarial examples.

### D. Relay Attacks

Relay attacks are becoming increasingly relevant in the context of CAPTCHAs, and have been revealed to be very difficult to fight against and relatively easy to deploy. They are also called 'human relay' attacks, 'human farms' or 'sweatshop' attacks in the literature. They are based on exploiting cheap labor for relaying CAPTCHA challenges to humans who can solve thousands of them per hour at a low cost (e.g., around $1, as reported in [29].)

These attacks are very difficult to stop, and there are very few proposals in the literature that offer any real protection against them. One such approach is a Dynamic Cognitive Game CAPTCHA [28] that, while offering some resistance to relay attacks is, in its current form, vulnerable to low-complexity automated dictionary attacks.

We note that any generic defense against relay attacks can be applied to the DeepCAPTCHA system: from relying on the client's original IP address to browser-specific characteristics, or the use of timing information. For example, as we have good timing estimates for the average solution times of DeepCAPTCHA challenges, one can easily introduce a time threshold to detect such attacks as suggested in [28].

## VII. PoC: DeepCAPTCHA-ILSVRC-2012 System

We implemented a proof-of-concept system using the CNN-F deep network from [7], trained on the ILSVRC-2012 database [37]. This set contains 1000 categories of natural images from ImageNet. The DL network was trained on the training set of the ILSVRC-2012 database, and we used the validation set that contains 50,000 images as a pool for source images (such a pool is used *only for the PoC system*; in a real-life system, a source image should be taken from a web source and discarded after creating the challenge). For each challenge we picked an image at random and produced an adversarial example for it using the IAN generation method, detailed in Algorithm 1. We selected one representative image per category from the training set (to guarantee that the answers do not contain the image, used to generate adversarial examples) for the answers.

The PoC system was implemented as a web application in order to conduct a number of usability tests. In our implementation we varied the number of answers to test the best trade-off between usability and security (more choices increase the security, but are harder for users and the solution takes more time). The number of challenges per session was set to 10 (note that our security analysis suggests that 2–3 answers are enough to reach the desired security level). An example of a challenge from the PoC system is shown in Figure 3.

### A. Usability Analysis of the PoC System

We tested the proof-of-concept implementation of our DeepCAPTCHA system using 472 participants contacted using the Microworkers.com online micro crowd sourcing service. Each participant was requested to provide anonymous statistical data about their age, gender and familiarity with computers before starting the test. Participants were next presented with 10 DeepCAPTCHA challenges of varying difficulties and gave feedback on usability once they had completed the challenges. This provided us with 4720 answered tests, of which we removed 182 (approx. a 3.85%) to avoid outliers. In particular, we removed tests or sessions if they fall into any of these three categories[14]: 1. Sessions with average time per test longer than

[13] We remind the reader that $I$ is **not** available to the adversary as per our assumptions.

[14] We assume that long solving times are due to users that were interrupted during the tests, and the low success rates are due to users that did not follow the instructions, or chose their answers at random.

TABLE IV
USABILITY RESULTS FOR THE DEEPCAPTCHA PROOF OF CONCEPT IMPLEMENTATION, WITH DIFFERENT NUMBER OF ANSWERS

| | Overall Results | 8 answers | 12 answers | 16 answers | 20 answers |
|---|---|---|---|---|---|
| Total test count | 4538 | 1257 | 990 | 1144 | 1147 |
| Success rate | 82,57 % | 89,18 % | 86,67 % | 79,98 % | 74,37 % |
| Average time | 7,89s | 6,04s | 7,66s | 8,36s | 9,66s |
| Median time | 5,49s | 4,24s | 5,18s | 5,89s | 7,34s |

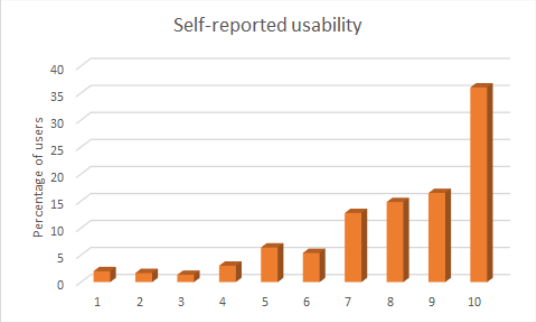



Fig. 4. Self reported user friendliness of DeepCAPTCHA. Answers in the range 1-10, 10 being best.

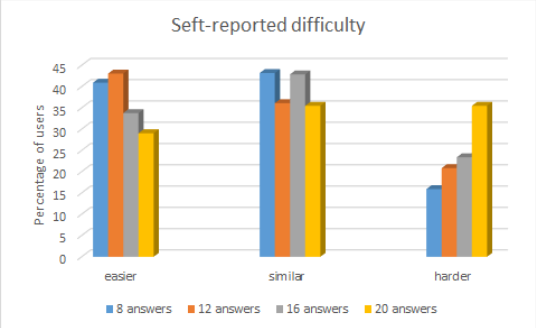



Fig. 5. Self reported DeepCAPTCHA difficulty, compared with existing CAPTCHAS, for variants from 8 to 20 answers.

40 seconds, 2. Tests with answer times above 45 seconds, and 3. Sessions with a success rate of 10% or lower.

We tried to get some insights into the best trade-off between usability and security by testing different numbers of answers, in the range $8+4k, k \in \{0,\ldots,3\}$, so users were randomly assigned variants of the tests with different number of answers for studying the impact of this change. The most relevant usability results are shown in Table IV. The participants reported high satisfaction with DeepCAPTCHA usability (see Figure 4). The data shown in Figure 4 is an average across all variants, from 8 to 20 answers. As expected, the perceived user-friendliness and difficulty (see Figure 5) of the DeepCAPTCHA deteriorated steadily from the versions with 8 answers to those with 20.

It is interesting to note that participants who declared their gender as female performed significantly better than the males, across all variants, the gap becoming wider with the increasing difficulty of the CAPTCHA task, as seen in Figure 6. Consistent with this finding is the additional fact that females not only achieved better accuracy but also did it using less time on average than males.

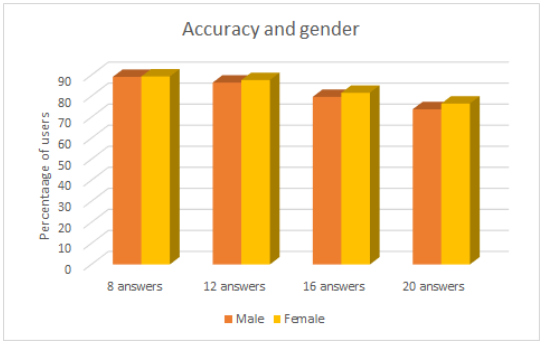



Fig. 6. Accuracy across self-reported gender for variants from 8 to 20 answers.

We define a secure CAPTCHA as one that has a less than 1.5% chance of being successfully attacked by a bot, and a usable CAPTCHA as one with a challenge pass rate above 75% when attempted by a human within an average time of 15s. These thresholds are in line with those previously reported and with other CAPTCHA schemes.

Based on the results collected so far in our preliminary tests, and the security analysis in Section VI, we conclude that the best trade-off between security and usability is met by the version of our test with 12 answers per challenge and two challenges in a CAPTCHA session. This configuration meets the accepted security and usability requisites. Namely, humans showed a success rate of 86.67% per challenge, hence the overall success probability is (assuming independence) about $0.8667^2 = 0.751$. The average time for the session was about $2 \cdot 7.66s = 15.32s$ (the median is significantly faster — $10.4s$). The security analysis showed that a probability of a bot bypassing the scheme is not higher than 0.7% (by random guessing).

We expect that once users will become more familiar with the task and the system (as the system gains popularity), the solution times and the success rates would improve.

## VIII. CONCLUSIONS AND FUTURE WORK

In this work, we introduced DeepCAPTCHA, a secure new CAPTCHA mechanism based on immutable adversarial noise that deceives DL tools and cannot be removed using preprocessing. DeepCAPTCHA offers a playful and friendly interface for performing one of the most loathed Internet-related tasks — solving CAPTCHAs. We also implemented a first proof-of-concept system and examined it in great detail.[15]

[15]DeepCAPTCHA can be accessed at http://crypto.cs.haifa.ac.il/~daniel

We are the first to pose the question of adversarial examples' immutability, in particular to techniques that attempt to remove the adversarial noise. Our analysis showed that previous methods are not robust to such attacks. To this end, we proposed a new construction for generating immutable adversarial examples which is significantly more robust to attacks, attempting to remove this noise, than existing methods.

There are three main directions for future CAPTCHA research:

- Design a new large-scale classification task for DeepCAPTCHA that contains a new data set of at least 1000 dissimilar categories of objects. This task also includes collecting (and labelling) a new data set for training of the CNN.
- Adversarial examples are trained per classification problem, meaning that they can operate on the set of labels they have been trained for. Switching to an alternative set of labels is likely to reduce their effectiveness. Another interesting future research topic could be to develop IANs for these scenarios, e.g., for hierarchy-based labels (such as Animal-Dog-Poodle.)
- The study and introduction of CAPTCHAs based on different modalities, such as sound/speech processing (e.g., to address users with visual impairments).

Finally, we believe that IANs could have a wide range of applications in computer security. They may be used to bypass current ML-based security mechanisms such as spam filters and behavior-based anti-malware tools. Additionally, our proposed attacks on adversarial noise may be of independent interest and lead to new research outcomes.

## ACKNOWLEDGEMENTS

The authors thank Daniel Osadchy for his worthy contributions to the paper and the anonymous reviewers for their ideas and suggestions.

## REFERENCES

[1] *Are You a Human*, accessed on May 2016. [Online]. Available: http://www.areyouahuman.com/

[2] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 60–65.

[3] E. Bursztein. *How we Broke the NuCaptcha Video Scheme and What we Proposed to Fix it*, accessed on May 2016. [Online]. Available: http://elie.im/blog/security

[4] E. Bursztein, J. Aigrain, A. Moscicki, and J. C. Mitchell, "The end is nigh: Generic solving of text-based CAPTCHAs," in *Proc. 8th USENIX Conf. Offensive Technol.*, Berkeley, CA, USA, 2014, pp. 1–15.

[5] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving CAPTCHAs? A large scale evaluation," in *Proc. IEEE Symp. Secur. Privacy.*, Washington, DC, USA, May 2010, pp. 399–413.

[6] N. Carlini *et al.*, "Hidden voice commands," in *Proc. USENIX Secur. Symp. (Security)*, Aug. 2016, pp. 513–530.

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014.

[8] K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski, "Designing human friendly human interaction proofs (HIPs)," in *Proc. SIGCHI Conf. Human Factors Comput. Syst.*, New York, NY, USA, 2005, pp. 711–720.

[9] N. Dalvi, P. Domingos, S. S. Mausam, and D. Verma, "Adversarial classification," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2004, pp. 99–108.

[10] R. Datta, J. Li, and J. Z. Wang, "IMAGINATION: A robust image-based CAPTCHA generation system," in *Proc. 13th ACM Int. Conf. Multimedia*, Singapore, 2005, pp. 331–334.

[11] J. Elson, J. R. Douceur, J. Howell, and J. Saul, "Asirra: A CAPTCHA that exploits interest-aligned manual image categorization," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 366–374.

[12] A. Fawzi, O. Fawzi, and P. Frossard, "Analysis of classifiers' robustness to adversarial perturbations," [Online]. Available: CoRR abs/1502.02590, (2015).

[13] P. Golle, "Machine learning attacks against the Asirra CAPTCHA," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2008, pp. 535–542.

[14] I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. D. Shet, "Multi-digit number recognition from street view imagery using deep convolutional neural networks," [Online]. Available: CoRR abs/1312.6082, (2013).

[15] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," [Online]. Available: CoRR abs/1412.6572, (2014).

[16] S. Gu and L. Rigazio. (2014). "Towards deep neural network architectures robust to adversarial examples." [Online]. Available: https://arxiv.org/abs/1412.5068

[17] C. J. Hernández-Castro, M. D. R-Moreno, and D. F. Barrero, "Using JPEG to measure image continuity and break capy and other puzzle CAPTCHAs," *IEEE Internet Comput.*, vol. 19, no. 6, pp. 46–53, Nov./Dec. 2015.

[18] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári, "Learning with a strong adversary," [Online]. Available: CoRR abs/1511.03034, (2015).

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[20] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," [Online]. Available: CoRR abs/1607.02533, (2016).

[21] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," [Online]. Available: CoRR abs/1611.01236, (2016).

[22] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[24] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," [Online]. Available: CoRR abs/1611.02770, (2016).

[25] M. K. Mihçak, I. Kozintsev, and K. Ramchandran, "Spatially adaptive statistical modeling of wavelet image coefficients and its application to denoising," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Mar. 1999, pp. 3253–3256.

[26] T. Miyato, S.-I. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," in *Proc. ICLR*, 2016, pp. 1–12.

[27] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[28] M. Mohamed *et al.*, "Three-way dissection of a game-CAPTCHA: Automated attacks, relay attacks, and usability," in *Proc. 9th Symp. Inf., Comput. Commun. Secur. (ASIA)*, 2014, pp. 195–206.

[29] M. Motoyama, K. Levchenko, C. Kanich, D. McCoy, G. M. Voelker, and S. Savage, "Understanding captcha-solving services in an economic context," in *Proc. USENIX Security.*, 2010, vol. 10.

[30] M. Naor. *Verification of a Human in the Loop or Identification via the Turing Test*. [Online]. Available: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human_abs.html

[31] A. M. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 427–436.

[32] NUCAPTCHA. *NuCaptcha & Traditional Captcha*, accessed on May 2016. [Online]. Available: http://nudatasecurity.com

[33] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," *CoRR*[Online]. Available: CoRR abs/1605.07277, (2016).

[34] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," [Online]. Available: CoRR abs/1511.07528, (2015).

[35] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," [Online]. Available: CoRR abs/1602.02697, (2016).

[36] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. 37th IEEE Symp. Secur. Privacy*, May 2016, pp. 582–597.
[37] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015.
[38] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet, "Adversarial manipulation of deep representations," [Online]. Available: CoRR abs/1511.05122, (2015).
[39] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," [Online]. Available: CoRR abs/1312.6229, (2013).
[40] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1528–1540.
[41] S. Sivakorn, I. Polakis, and A. D. Keromytis, "I am robot: (Deep) learning to break semantic image CAPTCHAs," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS P)*, Saarbrücken, Germany, Mar. 2016, pp. 388–403.
[42] C. Szegedy *et al.*, "Intriguing properties of neural networks," [Online]. Available: CoRR abs/1312.6199, (2013).
[43] M. Ulicný, J. Lundström, and S. Byttner, "Robustness of deep convolutional neural networks for image recognition," in *Proc. Int. Symp. Intell. Comput. Syst.*, Cham, Switzerland, 2016, pp. 16–30.
[44] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd Annu. ACM Conf. Multimedia*, 2015, pp. 689–692.
[45] 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.
[46] L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum, "reCAPTCHA: Human-based character recognition via Web security measures," *Science*, vol. 321, no. 5895, pp. 1465–1468, 2008.
[47] D. Warde-Farley and I. Goodfellow, "Adversarial perturbations of deep neural networks," in *Advanced Structured Prediction*, T. Hazan, G. Papandreou, and D. Tarlow, Eds. Cambridge, MA: MIT Press, 2016.
[48] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. 23nd Annu. Netw. Distrib. Syst. Secur. Symp.*, 2016, p. 15.
[49] Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. C. van Oorschot, "Security analysis and related usability of motion-based CAPTCHAs: Decoding codewords in motion," *IEEE Trans. Depend. Sec. Comput.*, vol. 11, no. 5, pp. 480–493, Sep./Oct. 2014.
[50] J. Yan and A. S. El Ahmad, "Breaking visual CAPTCHAs with naive pattern recognition algorithms," in *Proc. 23rd Comput. Secur. Appl. Conf.*, Dec. 2007, pp. 279–291.
[51] J. Yan and A. S. El Ahmad, "Usability of CAPTCHAs or usability issues in CAPTCHA design," in *Proc. 4th Symp. Usable Privacy Secur. (SOUPS)*, New York, NY, USA, Jul. 2008, pp. 44–52.
[52] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 818–833.
[53] B. B. Zhu *et al.*, "Attacks and design of image recognition CAPTCHAs," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, Oct. 2010, pp. 187–200.

**Margarita Osadchy** received the Ph.D. degree (Hons.) in computer science from the University of Haifa, Israel, in 2002. From 2001 to 2004, she was a Visiting Research Scientist with the NEC Research Institute and then a Post-Doctoral Fellow with the Department of Computer Science, Technion. Since 2005, she has been with the Computer Science Department, University of Haifa. Her main research interests are machine learning, computer vision, and computer security, and privacy.

**Julio Hernandez-Castro** received the degree in mathematics in 1995, the M.Sc. degree in coding theory and network security in 1999 and the Ph.D. degree in computer science in 2003. He is currently a Senior Lecturer in computer security with the School of Computing, University of Kent, U.K. His interests are in computer forensics, especially in steganography and steganalysis, but also in machine learning applications to cybersecurity, RFID/NFC Security, and in advancing the study of randomness generation and testing. He is also involved in analyzing ransomware and other types of malware. He is a keen chess player and Python enthusiast.

**Stuart Gibson** received the B.Sc. and Ph.D. degrees in physics from the University of Kent, U.K., in 1999 and 2007, respectively. He is currently a Senior Lecturer and the Director of Innovation and Enterprise with the School of Physical Sciences, University of Kent. His research interests include image and signal processing, facial identification, photo forensics, machine learning, and artificial intelligence.

**Orr Dunkelman** received the B.A. and Ph.D. degrees in computer science from Technion in 2000 and 2006, respectively. He is currently an Associate Professor with the Computer Science Department, University of Haifa. His interests are cryptography, with a strong emphasis on cryptanalysis, computer security, and privacy.

**Daniel Pérez-Cabo** received the B.Sc. degree in telecommunications engineering majoring in communication systems from the University of Vigo, Spain, in 2011, and the M.Sc. degree in telecommunication in 2013. He is currently pursuing the Ph.D. degree with the University of Vigo. He is also a Face Recognition Researcher in the multimodal information area with Gradiant, Spain.