A General Framework for Adversarial Examples with Objectives

MAHMOOD SHARIF, SRUTI BHAGAVATULA, and LUJO BAUER,
Carnegie Mellon University, USA
MICHAEL K. REITER, University of North Carolina at Chapel Hill, USA

Images perturbed subtly to be misclassified by neural networks, called *adversarial examples*, have emerged as a technically deep challenge and an important concern for several application domains. Most research on adversarial examples takes as its only constraint that the perturbed images are similar to the originals. However, real-world application of these ideas often requires the examples to satisfy additional objectives, which are typically enforced through custom modifications of the perturbation process. In this article, we propose *adversarial generative nets* (AGNs), a general methodology to train a *generator* neural network to emit adversarial examples satisfying desired objectives. We demonstrate the ability of AGNs to accommodate a wide range of objectives, including imprecise ones difficult to model, in two application domains. In particular, we demonstrate *physical* adversarial examples—eyeglass frames designed to fool face recognition—with better robustness, inconspicuousness, and scalability than previous approaches, as well as a new attack to fool a handwritten-digit classifier.

CCS Concepts: • **Security and privacy** → **Biometrics**; • **Computing methodologies** → *Supervised learning by classification*; *Neural networks*;

Additional Key Words and Phrases: Machine learning, neural networks, face recognition, adversarial examples

ACM Reference format:
Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2019. A General Framework for Adversarial Examples with Objectives. *ACM Trans. Priv. Secur.* 22, 3, Article 16 (June 2019), 30 pages.
https://doi.org/10.1145/3317611

1 INTRODUCTION

Deep neural networks (DNNs) are popular machine-learning models that achieve state-of-the-art results on challenging learning tasks in domains where there is adequate training data and compute power to train them. For example, they have been shown to outperform humans in face verification, i.e., deciding whether two face images belong to the same person [32, 74].

This work was supported in part by the Multidisciplinary University Research Initiative (MURI) Cyber Deception grant; by NSF grants 1801391 and 1801494; by the National Security Agency under Award No. H9823018D0008; by gifts from Google and Nvidia, and from Lockheed Martin and NATO through Carnegie Mellon CyLab; and by a CyLab Presidential Fellowship and a Symantec Research Labs Fellowship.

Authors' addresses: M. Sharif, S. Bhagavatula, and L. Bauer, Carnegie Mellon University, Collaborative Innovation Center, 4720 Forbes Ave., Pittsburgh, PA, 15213, USA; emails: {mahmoods, srutib, lbauer}@cmu.edu; M. K. Reiter, University of North Carolina at Chapel Hill, Chapel Hill, Sitterson Hall, 201 S. Columbia St., Chapel Hill, North Carolina, 27599, USA; email: reiter@cs.unc.edu.

2471-2566/2019/06-ART16
https://doi.org/10.1145/ 3317611

Unfortunately, it has also been shown that DNNs can be easily fooled by *adversarial examples*—mildly perturbed inputs that to a human appear visually indistinguishable from benign inputs—and that such adversarial examples can be systematically found [10, 73].

While the early attacks and almost all extensions (see Section 2) attempt to meet very few objectives other than the adversarial input being similar (by some measure) to the original, there are many contexts in which it is necessary to model additional objectives of adversarial inputs. For example, our prior work considered a scenario in which adversaries could not manipulate input images directly but, rather, could only manipulate the physical artifacts captured in such images [68]. Using eyeglasses for fooling face-recognition systems as a driving example, we showed how to encode various objectives into the process of generating eyeglass frames, such as ensuring that the frames were capable of being physically realized by an off-the-shelf printer. As another example, Evtimov et al. considered generating shapes that, when attached to street signs, would seem harmless to human observers but would lead neural networks to misclassify the signs [20].

These efforts modeled the various objectives they considered in an ad hoc fashion. In contrast, in this article, we propose a *general framework* for capturing such objectives in the process of generating adversarial inputs. Our framework builds on recent work in generative adversarial networks (GANs) [25] to train an attack *generator*, i.e., a neural network that can generate successful attack instances that meet certain objectives. Moreover, our framework is not only general, but, unlike previous attacks, produces a large number of *diverse* adversarial examples that meet the desired objectives. This could be leveraged by an attacker to generate attacks that are unlike previous ones (and hence more likely to succeed), but also by defenders to generate labeled negative inputs to augment training of their classifiers. Due to our framework's basis in GANs, we refer to it using the anagram AGNs, for *adversarial generative nets*.

To illustrate the utility of AGNs, we return to the task of printing eyeglasses to fool face-recognition systems [68] and demonstrate how to accommodate a number of types of objectives within it. Specifically, we use AGNs to accommodate *robustness* objectives to ensure that produced eyeglasses fool face-recognition systems in different imaging conditions (e.g., lighting, angle) and even despite the deployment of specific defenses; *inconspicuousness* objectives, so that the eyeglasses will not arouse the suspicion of human onlookers; and *scalability* objectives requiring that relatively few adversarial objects are sufficient to fool DNNs in many contexts. We show that AGNs can be used to target two DNN-based face-recognition algorithms that achieve human-level accuracy—VGG [59] and OpenFace [1]—and output eyeglasses that enable an attacker to either evade recognition or to impersonate a specific target, while meeting these additional objectives. To demonstrate that AGNs can be effective in contexts other than face recognition, we also train AGNs to fool a classifier designed to recognize handwritten digits and trained on the MNIST dataset [43].

In addition to illustrating the extensibility of AGNs to various types of objectives, these demonstrations highlight two additional features that, we believe, are significant advances. First, AGNs are *flexible* in that an AGN can train a generator to produce adversarial instances with only vaguely specified characteristics. For example, we have no way of capturing inconspicuousness mathematically; rather, we can specify it only using labeled instances. Still, AGNs can be trained to produce new and convincingly inconspicuous adversarial examples. Second, AGNs are *powerful* in generating adversarial examples that perform better than those produced in previous efforts using more customized techniques. For example, though some of the robustness and inconspicuousness objectives we consider here were also considered in prior work, the adversarial instances produced by AGNs perform better (e.g., ~70% vs. 31% average success rate in impersonation) and accommodate other objectives (e.g., robustness to illumination changes). AGNs enable attacks that previous methods did not.

We next review related work (Section 2) and then describe the AGN framework (Section 3) and its instantiation against face-recognition DNNs (Section 4). Then, we evaluate the effectiveness of AGNs, including with physically realized attacks and a user study to examine inconspicuousness (Section 5). Finally, we discuss our work and conclude (Section 6).

2 RELATED WORK

In this section, we describe prior work on test-time attacks on machine-learning algorithms, work that focuses on physical-world attacks, and proposed defenses.

Fooling Machine-learning Algorithms. In concurrent research efforts, Szegedy et al. and Biggio et al. showed how to systematically find adversarial examples to fool DNNs [10, 73]. Given an input x that is classified to $F(x)$ by the DNN, Szegedy et al.'s goal was to find a perturbation r of minimal norm (i.e., as imperceptible as possible) such that $x + r$ would be classified to a desired target class c_t. They showed that, when the DNN function, $F(\cdot)$, and the norm function are differentiable, finding the perturbation can be formalized as an optimization to be solved by the L-BFGS solver [56]. Differently from the minimal perturbations proposed by Szegedy et al., Biggio et al. focused on finding perturbations that would significantly increase the machine-learning algorithm's confidence in the target class [10]. Attacks akin to Biggio et al.'s are more suitable for the security domain, where one may be interested in assessing the security of algorithms or systems under worst-case attacks [9, 24].

More efficient algorithms were later proposed for finding adversarial examples that were even more imperceptible (using different notions of imperceptibility) or were misclassified with even higher confidence [13, 19, 27, 33, 35, 51, 52, 57, 63, 64]. For example, Papernot et al.'s algorithm aims to minimize the number of pixels changed [57]. Carlini and Wagner experimented with different formulations of the optimization's objective function for finding adversarial examples [13]. They found that minimizing a weighted sum of the perturbation's norm and a particular classification loss-function, $Loss_{cw}$, helps achieve more imperceptible attacks. They defined $Loss_{cw}$ not directly over the probabilities emitted by $F(\cdot)$ but rather over the *logits*, $L(\cdot)$. The logits are usually the output of the one-before-last layer of DNNs, and higher logits for a class imply higher probability assigned to it by the DNN. Roughly speaking, $Loss_{cw}$ was defined as follows:

$$Loss_{cw} = \max\{L_c(x + r) : c \neq c_t\} - L_{c_t}(x + r),$$

where $L_c(\cdot)$ is the logit for class c. Minimizing $Loss_{cw}$ increases the probability of the target class, c_t, and decreases the probability of others.

Perhaps closest to our work is the work of Baluja and Fischer [8]. They propose to train an auto-encoding neural network that takes an image as input and outputs a perturbed version of the same image that would be misclassified. Follow-up research efforts concurrent to ours propose to train generative neural networks to create adversarially perturbed images that lead to misclassification [61, 77, 82]. These attacks require only that the perturbations have a small norm, and allow perturbations to cover the entire image. In contrast, as we discuss in Sections 3–4, we propose attacks that must satisfy stricter constraints (e.g., cover only a small, specific portion of the image) and multiple objectives (e.g., generate eyeglasses that both lead to misclassification and look realistic).

Moosavi et al. showed how to find universal adversarial perturbations, which lead not just one image to be misclassified, but a large set of images [53]. Universal perturbations improve our understanding of DNNs' limitations, as they show that adversarial examples often lie in fixed directions (in the images' RGB space) with respect to their corresponding benign inputs. Differently from that work, we explore universal attacks that are both inconspicuous and constrained to a small region.

Some work has explored digital-domain attacks that satisfy certain objectives [15, 29, 70, 78]. For example, Xu et al. showed to automatically create malicious PDFs that cannot be detected using machine-learning algorithms [78]. These methods use customized algorithms to achieve very precise objectives (e.g., create a valid PDF output). Our work instead focuses on a general method for meeting objectives that might be imprecise.

Research suggests that adversarial examples are not a result of overfitting, as in that case adversarial examples would be unlikely to transfer between models (i.e., to fool models with different architecture or training data than the ones the adversarial examples were crafted for) [76]. A widely held conjecture attributes adversarial examples to the inflexibility of classification models [21, 27, 67, 76]. This conjecture is supported by the success of attacks that approximate DNNs' classification boundaries by linear separators [27, 52].

Physical Attacks on Machine Learning. Kurakin et al. demonstrated that imperceptible adversarial examples can fool DNNs even if the input to the DNN is an image of the adversarial example printed on paper [42]. Differently than us, they created adversarial perturbations that covered the entire image they aimed to misclassify. Recently, Evtimov et al. showed that specially crafted patterns printed and affixed to street signs can mislead DNNs for street-sign recognition [20]. Unlike the work described in this article, they specified the printability and inconspicuousness objectives in an ad-hoc fashion.

Our prior work proposed using eyeglasses to perform physically realizable dodging and impersonation against state-of-the-art DNNs for facial recognition [68]. The problem of finding adversarial eyeglass patterns was formalized as an optimization problem with multiple ad hoc objectives to increase the likelihood that (1) face recognition can be fooled even when the attacker's pose changes slightly; (2) transitions between neighboring pixels on the eyeglasses are smooth; and (3) the eyeglasses' colors can be realized using an off-the-shelf printer. Unlike that prior work, the approach described in this article is a general framework for generating adversarial inputs, in this context instantiated to generate eyeglass patterns as similar as possible to real designs, while still fooling the DNNs in a desired manner. Moreover, unlike prior work, we evaluate the inconspicuousness of the eyeglasses using a user study. We find that the new algorithm can produce more robust and inconspicuous attacks (Section 5.2 and Section 5.5). We also show the attacks produced by the method we describe here to be scalable as well as robust in the face of defenses.

Another line of work attempts to achieve privacy from face-recognition systems by completely avoiding *face detection* [31, 79]. Essentially, face detection finds sub-windows in images that contain faces, which are later sent for processing by face-recognition systems. Consequently, by evading detection, one avoids the post processing of her face image by recognition systems. The proposed techniques are not inconspicuous: they either use excessive makeup [31] or attempt to blind the camera using light-emitting eyeglasses [79].

The susceptibility to attacks of learning systems that operate on non-visual input has also been studied [14, 16, 17, 81]. For instance, researchers showed that speech-recognition systems can be misled to interpret sounds unintelligible to humans as actual commands [14].

Defending Neural Networks. Proposals to ameliorate DNN's susceptibility to adversarial examples follow three main directions. One line of work proposes techniques for training DNNs that would correctly classify adversarial inputs or would not be susceptible to small perturbations. Such techniques involve augmenting training with adversarial examples in the hope that the DNN will learn to classify them correctly [27, 36, 37, 41, 73]. These techniques were found to increase the norms of the perturbations needed to achieve misclassification. However, it remains unclear whether the increase is sufficient to make adversarial examples noticeable to humans. A recent adversarial training method significantly enhanced the robustness of DNNs by training using examples generated with the Projected Gradient Sign (*PGD*) attack, which is conjectured to be the strongest attack using local derivative information about DNNs [45]. Two subsequent defenses achieved relatively high success by approximating the outputs achievable via certain norm-bounded perturbations and then ensuring these outputs are classified correctly [39, 50]. Unfortunately, the recent defenses [36, 39, 45, 50] are limited to specific types of perturbations (e.g., ones bounded in L_∞), similarly to their predecessors [27, 37, 41, 73].

A second line of work proposes techniques to detect adversarial examples (e.g., References [22, 28, 48, 49]). The main assumption of this line of work is that adversarial examples follow a different distribution than benign inputs and hence can be detected via statistical techniques. For instance, Metzen et al. propose to train a neural network to detect adversarial examples [49]. The detector would take its input from an intermediate layer of a DNN and decide whether the input is adversarial. It was recently shown that this detector, as well as others, can be evaded using different attack techniques than the ones on which these detectors were originally evaluated [12].

A third line of work suggests to transform the DNNs' inputs to sanitize adversarial examples and lead them to be correctly classified, while keeping the DNNs' original training procedures intact [30, 44, 48, 66, 69]. The transformations aim to obfuscate the gradients on which attacks often rely. In certain cases the defenses even rely on undifferentiable transformations (e.g., JPEG compression) to prevent the back propagation of gradients (e.g., Reference [30]). Unfortunately, researchers have shown that it is possible to circumvent such defenses, sometimes by vanilla attacks [4] and other times by more advanced means (e.g., by approximating the input-transformation functions using smooth and differentiable functions) [5].

3 A NOVEL ATTACK AGAINST DNNS

In this section, we describe a new algorithm to attack DNNs. We define our threat model in Section 3.1, discuss the challenges posed by vaguely specified objectives in Section 3.2, provide background on Generative Adversarial Networks in Section 3.3, and describe the attack framework in Section 3.4.

3.1 Threat Model

We assume an adversary who gains access to an already trained DNN (e.g., one trained for face recognition). The adversary cannot poison the parameters of the DNN by injecting mislabeled data or altering training data. Instead, she can only alter the inputs to be classified.

The attacker's goal is to trick the DNN into misclassifying adversarial inputs. We consider two variants of this attack. In *targeted* attacks, the adversary attempts to trick the DNN to misclassify the input as a specific class (e.g., to *impersonate* another subject enrolled in a face-recognition system). In *untargeted* attacks, the adversary attempts to trick the DNN to misclassify the input as an arbitrary class (e.g., to *dodge* recognition by a face-recognition system).

The framework we propose supports attacks that seek to satisfy a variety of objectives, such as maximizing the DNN's confidence in the target class in impersonation attacks and crafting perturbations that are inconspicuous. Maximizing the confidence in the target class is especially important in scenarios where strict criteria may be used in an attempt to ensure security—for instance, scenarios when the confidence must be above a threshold, as is used to to prevent false positives in face-recognition systems [34]. While inconspicuousness may not be necessary in certain scenarios (e.g., unlocking a mobile device via face recognition), attacks that are not inconspicuous could easily be ruled out in some safety-critical scenarios (e.g., when human operators monitor face-recognition systems at airports [72]).

We assume a *white-box* scenario: The adversary knows the feature space (images in RGB representation, as is typical in DNNs for image classification) and the architecture and parameters of the system being attacked. Studying robustness of DNNs under such assumptions is standard in the literature (e.g., References [13, 52]). Moreover, as shown in Section 5.4 and in prior work (e.g., Reference [58]), black-box attacks can be built from white-box attacks on local substitute-models. Gradient approximation techniques could also be used to generalize our proposed method to black-box settings (e.g., References [23, 54]).

3.2 Vaguely Specified Objectives

In practice, certain objectives, such as inconspicuousness, may elude precise specification. In early stages of our work, while attempting to produce eyeglasses to fool face recognition, we attempted multiple ad-hoc approaches to enhance the inconspicuousness of the eyeglasses, with limited success. For instance, starting from solid-colored eyeglasses in either of the RGB or HSV color spaces, we experimented with algorithms that would gradually adjust the colors until evasion was achieved, while fixing one or more of the color channels. We also attempted to use Compositional Pattern-Producing Neural Networks [71] combined with an evolutionary algorithm to produce eyeglasses with symmetric or repetitive patterns. These approaches had limited success both at capturing inconspicuousness (e.g., real eyeglasses do not necessarily have symmetric patterns) and at evasion or failed completely.

Other approaches to improve inconspicuousness such as ensuring that the transitions between neighboring pixels are smooth [68] or limiting the extent to which pixels are perturbed (see the *CCS16* and $\widehat{CCS16}$ attacks in Section 5) had some success at evasion but resulted in patterns that our user-study participants deemed as distinguishable from real eyeglasses' patterns (see Section 5.5). Therefore, instead of pursuing such ad hoc approaches to formalize properties that may be insufficient or unnecessary for inconspicuousness, in this work we achieve inconspicuousness via a general framework that models inconspicuous eyeglasses based on many examples thereof, while simultaneously achieving additional objectives, such as evasion.

3.3 Generative Adversarial Networks

Our attacks build on Generative Adversarial Networks (GANs) [25] to create accessories (specifically, eyeglasses) that closely resemble real ones. GANs provide a framework to train a neural network, termed the *generator* (G), to generate data that belongs to a distribution (close to the real one) that underlies a target dataset. G maps samples from a distribution, Z, that we know how to sample from (such as $[-1, 1]^d$, i.e., d-dimensional vectors of reals between -1 and 1) to samples from the target distribution.

To train G, another neural network, called the discriminator (D), is used. D's objective is to discriminate between real and generated samples. Thus, training can be conceptualized as a game with two players, D and G, in which D is trained to emit 1 on real examples and 0 on generated samples, and G is trained to generate outputs that are (mis)classified as real by D. In practice, training proceeds iteratively and alternates between updating the parameters of G and D via back-propagation. G is trained to minimize the following function:

$$Loss_G(Z, D) = \sum_{z \in Z} \lg(1 - D(G(z))). \tag{1}$$

$Loss_G$ is minimized when G misleads D (i.e., $D(G(z))$ is 1). D is trained to maximize the following function:

$$Gain_D(G, Z, data) = \sum_{x \in data} \lg(D(x)) + \sum_{z \in Z} \lg(1 - D(G(z))). \tag{2}$$

$Gain_D$ is maximized when D emits 1 on real samples and 0 on all others.

Several GAN architectures and training methods have been proposed, including Wasserstein GANs [2] and Deep Convolutional GANs [62], on which we build.

3.4 Attack Framework

Except for a few exceptions [8, 20, 61, 68, 82], in traditional evasion attacks against DNNs the attacker directly alters benign inputs to maximize or minimize a pre-defined function related to the desired misclassification (see Section 2). Differently from previous attacks, we propose to train neural networks to generate outputs that can be used to achieve desired evasions (among other objectives), instead of iteratively tweaking benign inputs to become adversarial.

More specifically, we propose to train neural networks to generate images of artifacts (e.g., eyeglasses) that would lead to misclassification. We require that the artifacts generated by these neural networks resemble a reference set of artifacts (e.g., real eyeglass designs), as a means to satisfy an objective that is hard to specify precisely (e.g., inconspicuousness). We call the neural networks we propose adversarial generative nets (*AGNs*). Similarly to GANs, AGNs are adversarially trained against a discriminator to learn how to generate realistic images. Differently from GANs, AGNs are also trained to generate (adversarial) outputs that can mislead given neural networks (e.g., neural networks designed to recognize faces).

Formally, three neural networks comprise an AGN: a generator, G; a discriminator, D; and a pre-trained DNN whose classification function is denoted by $F(\cdot)$. When given an input x to the DNN, G is trained to generate outputs that fool $F(\cdot)$ and are inconspicuous by minimizing[1]

$$Loss_G(Z, D) - \kappa \cdot \sum_{z \in Z} Loss_F(x + G(z)). \tag{3}$$

We define $Loss_G$ in the same manner as in Equation (1); minimizing it aims to generate real-looking (i.e., inconspicuous) outputs that mislead D. $Loss_F$ is a loss function defined over the DNN's classification function that is maximized when training G (as $-Loss_F$ is minimized). The definition of $Loss_F$ depends on whether the attacker aims to achieve an untargeted misclassification or a targeted one. For untargeted attacks, we use:

$$Loss_F(x + G(z)) = \sum_{i \neq x} F_{c_i}(x + G(z)) - F_{c_x}(x + G(z)),$$

while for targeted attacks we use:

$$Loss_F(x + G(z)) = F_{c_t}(x + G(z)) - \sum_{i \neq t} F_{c_i}(x + G(z)),$$

where $F_c(\cdot)$ is the DNN's output for class c (i.e., the estimated probability of class c in case of a *softmax* activation in the last layer). By maximizing $Loss_F$, for untargeted attacks, the probability of the correct class c_x decreases; for targeted attacks, the probability of the target class c_t increases. We chose this definition of $Loss_F$, because we empirically found that it causes AGNs to converge faster than $Loss_{cw}$ or loss functions defined via cross entropy, as used in prior work [13, 68]. κ is a parameter that balances the two objectives of G; we discuss it further below.

As part of the training process, D's weights are updated to maximize $Gain_D$, defined in Equation (2), to tell apart realistic and generated samples. In contrast to D and G, $F(\cdot)$'s weights are unaltered during training (as attacks should fool the same DNN at test time).

The algorithm for training AGNs is provided in Algorithm 1. The algorithm takes as input a set of benign examples (X), a pre-initialized generator and discriminator, a neural network to be

[1]We slightly abuse notation by writing $x + r$ to denote an image x that is modified by a perturbation r. In practice, we use a mask and set the values of x within the masked region to the exact values of r.

ALGORITHM 1: AGN training

```
Input: $X$, $G$, $D$, $F(\cdot)$, $dataset$, $Z$, $N_e$, $s_b$, $\kappa \in \{0,1\}$
Output: Adversarial $G$
for $e \leftarrow 1$ to $N_e$ do
    create $mini\text{-}batches$ of size $s_b$ from $dataset$;
    for $batch \in mini\text{-}batches$ do
        $z \leftarrow s_b$ samples from $Z$;
        $gen \leftarrow G(z)$;
        $batch \leftarrow concat(gen, batch)$;
        if even iteration then // update $D$
            update $D$ by backpropagating $\frac{\partial Gain_D}{\partial batch}$;
        else // update $G$
            if $F(\cdot)$ fooled then return $G$;
            $d_1 \leftarrow -\frac{\partial Gain_D}{\partial gen}$;
            $x \leftarrow s_b$ sample images from $X$;
            $x \leftarrow x + gen$;
            Compute forward pass $F(x)$;
            $d_2 \leftarrow \frac{\partial Loss_F}{\partial gen}$;
            $d_1, d_2 \leftarrow normalize(d_1, d_2)$;
            $d \leftarrow \kappa \cdot d_1 + (1-\kappa) \cdot d_2$;
            update $G$ via backpropagating $d$;
```

fooled, a dataset of real examples (which the generator's output should resemble; in our case this is a dataset of eyeglasses), a function for sampling from G's latent space (Z), the maximum number of training epochs (N_e), the mini-batch[2] size s_b, and $\kappa \in [0,1]$. The result of the training process is an *adversarial generator* that creates outputs (e.g., eyeglasses) that fool $F(\cdot)$. In each training iteration, either D or G is updated using a subset of the data chosen at random. D's weights are updated via gradient ascent to increase $Gain_D$; G's weights are updated via gradient descent to minimize Equation (3). To balance the generator's two objectives, the gradients from $Gain_D$ and $Loss_F$ are normalized to the lower Euclidean norm of the two and then combined into a weighted average controlled by κ. When κ is closer to zero, more weight is given to fooling $F(\cdot)$ and less to making the output of G realistic. Conversely, setting κ closer to one places more weight on increasing the resemblance between G's output and real examples. Training ends when the maximum number of training epochs is reached, or when $F(\cdot)$ is fooled, i.e., when impersonation or dodging is achieved.

4 AGNS THAT FOOL FACE RECOGNITION

We next describe how we trained AGNs to generate inconspicuous, adversarial eyeglasses that can mislead state-of-the-art DNNs trained to recognize faces. To do so, we (1) collect a dataset of real eyeglasses, (2) select the architecture of the generator and the discriminator and instantiate their weights, (3) train DNNs that can evaluate the attacks, and (4) set the parameters for the attacks.

[2]Mini-batch: A subset of samples from the dataset used to approximate the gradients and compute updates in an iteration of the algorithm.

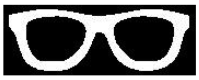


Fig. 1. A silhouette of the eyeglasses we use.

Fig. 2. Examples of raw images of eyeglasses that we collected (left) and their synthesis results (right).

4.1 Collecting a Dataset of Eyeglasses

A dataset of real eyeglass-frame designs is necessary to train the generator to create real-looking attacks. We collected such a dataset using Google's search API.[3] To collect a variety of designs, we searched for "eyeglasses" and synonyms (e.g., "glasses," "eyewear"), sometimes modified by an adjective, including colors (e.g., "brown," "blue"), trends (e.g., "geek," "tortoise shell"), and brands (e.g., "Ralph Lauren," "Prada"). In total, we made 430 unique API queries and collected 26,520 images.

The images we collected were not of only eyeglasses; e.g., we found images of cups, vases, and logos of eyeglass brands. Some images were of eyeglasses worn by models or on complex backgrounds. Such images would hinder the training process. Hence, we trained a classifier to detect and keep only images of eyeglasses over white backgrounds and not worn by models. Using 250 hand-labeled images, we trained a classifier that identified such images with 100% precision and 65% recall. After applying it to all the images in the dataset, 8,340 images remained. Manually examining a subset of these images revealed no false positives.

Using images from this dataset, we could train a generator that can emit eyeglasses of different patterns, shapes, and orientations. However, variations in shape and orientation made such eyeglasses difficult to efficiently align to face images while running Algorithm 1. Therefore, we preprocessed the images in the dataset and transferred the patterns from their frames to a fixed shape (a silhouette of the shape is shown in Figure 1), which we could then easily align to face images. We then trained the generator to emit images of eyeglasses with this particular shape, but with different colors and textures. To transfer the colors and textures of eyeglasses to a fixed shape, we thresholded the images to detect the areas of the frames. (Recall that the backgrounds of the images were white.) We then used Efros and Leung's texture-synthesis technique to synthesize the texture from the frames onto the fixed shape [18]. Figure 2 shows examples. Since the texture synthesis process is nondeterministic, we repeated it twice per image. At the end of this process, we had 16,680 images for training.

Since physical realizability is a requirement for our attacks, it was important that the generator emitted images of eyeglasses that are *printable*. In particular, the colors of the eyeglasses needed to be within the range our commodity printer (see Section 5.2) could print. Therefore, we mapped the colors of the eyeglass frames in the dataset into the color gamut of our printer. To model the color gamut, we printed an image containing all 2^{24} combinations of RGB triplets, captured a picture of that image, and computed the convex hull of all the RGB triplets in the captured image. To make an image of eyeglasses printable, we mapped each RGB triplet in the image to the closest RGB triplet found within the convex hull.

[3]https://developers.google.com/custom-search/.

4.2 Pretraining the Generator and the Discriminator

When training GANs, it is desirable for the generator to emit sharp, realistic, diverse images. Emitting only a small set of images would indicate the generator's function does not approximate the underlying distribution well. To achieve these goals, and to enable efficient training, we chose the Deep Convolutional GAN, a minimalistic architecture with a small number of parameters [62]. In particular, this architecture is known for its ability to train generators that can emit sharp, realistic images.

We then explored a variety of options for the generator's latent space and output dimensionality, as well as the number of weights in both G and D (via adjusting the depth of filters). We eventually found that a latent space of $[-1, 1]^{25}$ (i.e., 25-dimensional vectors of real numbers between -1 and 1) and output images of 64×176 pixels produced the best-looking, diverse results. The final architectures of G and D are reported in Figure 3.

To ensure that attacks converged quickly, we initialized G and D to a state in which the generator can already produce real-looking images of eyeglasses. To do so, we pretrained G and D for 200 epochs and stored them to initialize later runs of Algorithm 1.[4] Moreover, we used Salimans et al.'s recommendation and trained D on *soft labels* [65]. Specifically, we trained D to emit 0 on samples originating from the generator and 0.9 (instead of 1) on real examples. Figure 4 presents a couple of eyeglasses emitted by the pretrained generator.

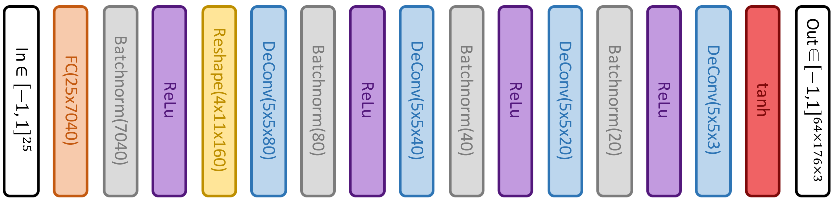
4.3 DNNs for Face Recognition

We evaluated our attacks against four DNNs of two architectures. Two of the DNNs were built on the *Visual Geometry Group (VGG)* neural network [59]. The original VGG DNN exhibited state-of-the-art results on the Labeled Faces in the Wild (LFW) benchmark, with 98.95% accuracy for face verification [32]. The VGG architecture contains a large number of weights (the original DNN contains about 268.52 million parameters). The other two DNNs were built on the OpenFace neural network, which uses the Google FaceNet architecture [1]. OpenFace's main design consideration is to provide high accuracy with low training and prediction times so that the DNN can be deployed on mobile and IoT devices. Hence, the DNN is relatively compact, with 3.74 million parameters, but nevertheless achieves near-human accuracy on the LFW benchmark (92.92%).

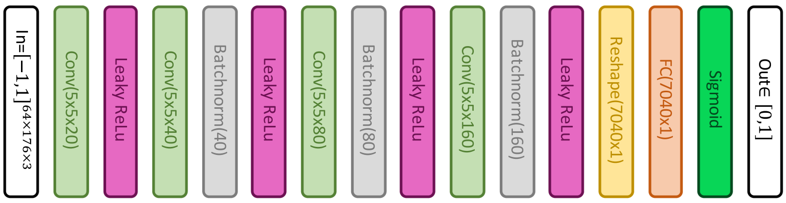
We trained one small and one large face-recognition DNN for each architecture. Since we wanted to experiment with physically realizable dodging and impersonation, we trained the DNNs to recognize a mix of subjects available to us locally and celebrities of whom we could acquire images for training. The small DNNs were trained to recognize five subjects from our research group (three females and two males) and five celebrities from the PubFig dataset [40]: Aaron Eckhart, Brad Pitt, Clive Owen, Drew Barrymore, and Milla Jovovich. We call the small DNN of the VGG and OpenFace architectures VGG10 and OF10. The large DNNs, termed VGG143 and OF143, were trained to recognize 143 subjects. Three of the subjects were members of our group, and 140 were celebrities with images in PubFig's evaluation set. In training, we used about 40 images per subject.

Training the VGG Networks. The original VGG network takes a 224×224 aligned face image as input and produces a highly discriminative face descriptor (i.e., vector representation of the face) of 4,096 dimensions. Two descriptors of images of the same person are designed to be closer to each other in Euclidean space than two descriptors of different people's images. We used the descriptors to train two simple neural networks that map face descriptors to probabilities over the set of identities. In this manner, the original VGG network effectively acted as a feature extractor.

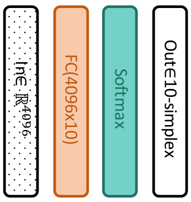
[4]For training, we used the Adam optimizer [38] and set the learning rate to 2e−4, the mini-batch size to 260, β_1 to 0.5, and β_2 to 0.999.



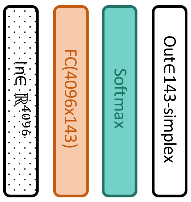
(a) *G* (generator).



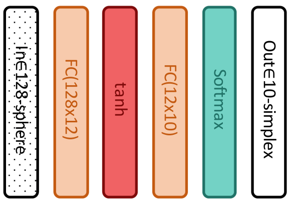
(b) *D* (discriminator).



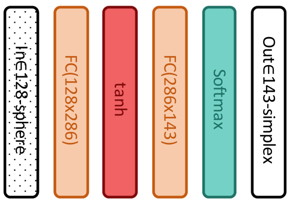
(c) VGG10.



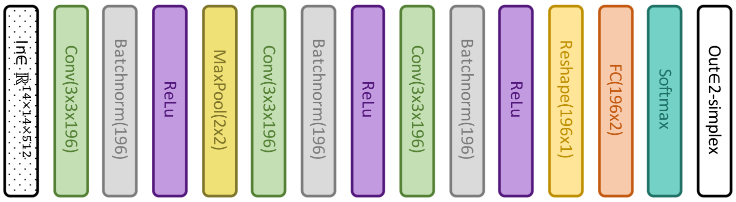
(d) VGG143.



(e) OF10.



(f) OF143.



(g) Detector

Fig. 3. Architectures of the neural networks used in this work. Inputs that are intermediate (i.e., received from feature-extraction DNNs) have dotted backgrounds. *Deconv* refers to transposed convolution and *FC* to fully connected layer. *N*-simplex refers to the set of probability vectors of *N* dimensions, and the 128-sphere denotes the set of real 128-dimensional vectors lying on the Euclidean unit sphere. All convolutions and deconvolutions in *G* and *D* have strides and paddings of two. The detector's convolutions have strides of two and padding of one. The detector's max-pooling layer has a stride of two.

Fig. 4. Examples of eyeglasses emitted by the generator (left) and similar eyeglasses from the training set (right).

Table 1. Performance of the Face-recognition DNNs

Model	*acc.*	*SR naïve dodge*	*SR naïve impers.*	*thresh.*	*TPR*	*FPR*
VGG10	100%	3%	0%	0.92	100%	0%
VGG143	98%	5%	0%	0.82	98%	0%
OF10	100%	14%	1%	0.55	100%	0%
OF143	86%	22%	<1%	0.91	59%	2%

We report the accuracy, the success rate (SR) of naïve dodging and impersonation (likelihood of naïve attackers to be misclassified arbitrarily or as *a priori* chosen targets), the threshold to balance correct and false classifications, the true-positive rate (TPR; how often the correct class is assigned a probability above the threshold), and the false-positive rate (FPR; how often a wrong class is assigned a probability above the threshold).

This is a standard training approach, termed *transfer learning*, which is commonly used to train high-performing DNNs using ones that have already been trained to perform a related task [80].

The architectures of the VGG-derived neural networks are provided in Figure 3. They consist of fully connected layers (i.e., linear separators) connected to a *softmax* layer that turns the linear separators' outputs into probabilities. We trained the networks using the standard technique of minimizing cross-entropy loss [26]. After training, we connected the trained neural networks to the original VGG network to construct end-to-end DNNs that map face images to identities.

An initial evaluation of VGG10 and VGG143 showed high performance. To verify that the DNNs cannot be easily misled, we tested them against naïve attacks by attaching eyeglasses emitted by the pretrained (non-adversarial) generator to test images. We found that impersonations of randomly picked targets are unlikely—they occur with 0.79% chance for VGG10 and <0.01% for VGG143. However, we found that dodging would succeed with non-negligible chance: 7.81% of the time against VGG10 and 26.87% against VGG143. We speculated that this was because the training samples for some subjects never included eyeglasses. To make the DNNs more robust, we augmented their training data following adversarial training techniques [41]: For each image initially used in training, we added two variants with generated eyeglasses attached. We also experimented with using more variants but found no additional improvement. Also following Kurakin et al., we included 50% raw training images and 50% augmented images in each mini-batch during training [41].

Evaluating VGG10 and VGG143 on held-out test sets after training, we found that they achieved 100% and 98% accuracy, respectively. In addition, the success of naïve dodging was at most 4.60% and that of impersonation was below 0.01%. Finally, to maintain a high level of security, it is important to minimize the DNNs' false positives [34]. One way to do so is by setting a criteria on the DNNs' output to decide when it should be accepted. We were able to find thresholds for the probabilities emitted by VGG10 and VGG143 such that their accuracies remained 100% and 98%, while the false-positive rates of both DNNs were 0%. The performance of the DNNs is reported in Table 1.

Training the OpenFace Networks. The original OpenFace network takes a 96×96 aligned face image as input and outputs a face descriptor of 128 dimensions. Similar to the VGG networks, the descriptors of images of the same person are close in Euclidean space, while the descriptors of different people's images are far. Unlike VGG, the OpenFace descriptors lie on a unit sphere.

We again used transfer learning to train the OpenFace networks. We first attempted to train neural networks that map the OpenFace descriptors to identities using architectures similar to the ones used for the VGG DNNs. We found these neural networks to achieve competitive accuracies. Similarly to the VGG DNNs, they were also vulnerable to naïve dodging attempts, but unlike the VGG DNNs, straightforward data augmentation did not improve their robustness. We believe this may stem from limitations of classifying data on a sphere using linear separators.

To improve the robustness of the DNNs, we increased their depth by prepending a fully connected layer followed by a hyperbolic-tangent (*tanh*) layer (see Figure 3). This architecture was chosen as it performed the best of different ones we experimented with. We also increased the number of images we augmented in training to 10 (per image in the training set) for OF10 and to 100 for OF143. The number of images augmented was selected such that increasing it did not further improve robustness against naïve attacks. Similarly to the VGG networks, we trained with about 40 images per subject, and included 50% raw images and 50% augmented images in training mini-batches.

We report the performance of the networks in Table 1. OF10 achieved 100% accuracy, while OF143 achieved 85.50% accuracy (comparable to Amos et al.'s finding [1]). The OpenFace DNNs were more vulnerable to naïve attacks than the VGG DNNs. For instance, OF10 failed against 14.10% of the naïve dodging attempts and 1.36% of the naïve impersonation attempts. We believe that the lower accuracy and higher susceptibility of the OpenFace DNNs compared to the VGG DNNs may stem from the limited capacity of the OpenFace network induced by the small number of parameters.

Training an Attack Detector. In addition to the face-recognition DNNs, we trained a DNN to detect attacks that target the VGG networks following the proposal of Metzen et al. [49]. We chose this detector because it was found to be one of the most effective detectors against imperceptible adversarial examples [12, 49]. We focused on VGG DNNs because no detector architecture was proposed for detecting attacks against OpenFace-like architectures. To mount a successful attack when a detector is deployed, it is necessary to simultaneously fool the detector and the face-recognition DNN.

We used the architecture proposed by Metzen et al. (see Figure 3). For best performance, we attached the detector after the fourth max-pooling layer of the VGG network. To train the detector, we used 170 subjects from the original dataset used by Parkhi et al. for training the VGG network [59]. For each subject we used 20 images for training. For each training image, we created a corresponding adversarial image that evades recognition. We trained the detector for 20 epochs using the Adam optimizer with the training parameters set to standard values (learning rate $= 1e-4$, $\beta_1 = 0.99$, $\beta_2 = 0.999$) [38]. At the end of training, we evaluated the detector on 20 subjects who were not used in training, finding that it had 100% recall and 100% precision.

4.4 Implementation Details

We used the Adam optimizer to update the weights of D and G when running Algorithm 1. As in pretraining, β_1 and β_2 were set to 0.5 and 0.999. We ran grid search to set κ and the learning rate and found that a $\kappa = 0.25$ and a learning rate of 5e−5 gave the best tradeoff between success in fooling the DNNs, inconspicuousness, and the algorithm's runtime. The number of epochs was

Table 2. Combinations of Domains, Attack Types, and Objectives Considered for Evaluation

	Domain		*Type*		*Objectives*							
						Robustness vs.						
Section	*Dig.*	*Phys.*	*Untarg.*	*Targ.*	*Inconspicuous.*	*Aug.*	*Detect.*	*Print.*	*Pose*	*Lum.*	*Universal.*	*Transfer.*
5.1	✓		✓		✓	✓		✓				
	✓		✓		✓	✓	✓	✓				
	✓			✓	✓	✓		✓				
	✓			✓	✓	✓	✓	✓				
5.2		✓	✓		✓	✓		✓	✓	✓		
		✓		✓	✓	✓		✓	✓	✓		
5.3	✓		✓		✓	✓		✓			✓	
5.4	✓		✓		✓	✓		✓				✓
	✓		✓		✓	✓		✓			✓	✓
		✓	✓		✓	✓		✓	✓	✓		✓

For each experimental section, we mark the combinations of domains (digital or physical) in which the attack was tested, the attack types (untargeted or targeted) tested, and the objectives of the attack, chosen from inconspicuousness, robustness (against training-data augmentation, detection, printing noise, pose changes, and luminance changes), universality, and transferability.
Note that while we did not explicitly design the attacks to transfer between architectures, we found that they transfer relatively well; see Section 5.4.

limited to at most one, as we found that the results the algorithm returned when running longer were not inconspicuous.

The majority of our work was implemented in MatConvNet, a MATLAB toolbox for convolutional neural networks [75]. The OpenFace DNN was translated from the original implementation in Torch to MatConvNet. We released the implementation online: https://github.com/mahmoods01/agns.

5 EVALUATING AGNS

We extensively evaluated AGNs as an attack method. In Section 5.1 we show that AGNs reliably generate successful dodging and impersonation attacks in a digital environment, even when a detector is used to prevent them. We show in Section 5.2 that these attacks can also be successful in the physical domain. In Section 5.3, we demonstrate universal dodging, i.e., generating a small number of eyeglasses that many subjects can use to evade recognition. We test in Section 5.4 how well our attacks transfer between models. Table 2 summarizes the combinations of domains the attacks were performed in, the attack types considered, and what objectives were aimed or tested for in each experiment. While more combinations exist, we attempted to experiment with the most interesting combinations under computational and manpower constraints. (For example, attacks in the physical domain require significant manual effort, and universal attacks require testing with more subjects than is feasible to test with in the physical domain.) In Section 5.5 we demonstrate that AGNs can generate eyeglasses that are inconspicuous to human participants in a user study. Finally, in Section 5.6 we show that AGNs are applicable to areas other than face recognition (specifically, by fooling a digit-recognition DNN).

5.1 Attacks in the Digital Domain

In contrast to physically realized attacks, an attacker in the digital domain can exactly control the input she provides to DNNs, since the inputs are not subject to noise added by physically realizing the attack or capturing the image with a camera. Therefore, our first step is to verify that the attacker can successfully fool the DNNs in the digital domain, as failure in the digital domain implies failure in the physical domain.

Experiment Setup. To evaluate the attacks in the digital domain, we selected a set of subjects for each DNN from the subjects the DNNs were trained on: 20 subjects selected at random for VGG143 and OF143 and all 10 subjects for VGG10 and OF10. In impersonation attacks, the targets were chosen at random. To compute the uncertainty in our estimation of success, we repeated each attack three times, each time using a different image of the attacker.

As baselines for comparison, we evaluated three additional attacks using the same setup. The first attack, denoted *CCS16*, is our proposed attack from prior work [68]. The *CCS16* attack iteratively modifies the colors of the eyeglasses until evasion is achieved. We ran the attack up to 300 iterations, starting from solid colors, and clipping the colors of the eyeglasses to the range $[0, 1]$ after each iteration to ensure that they lie in a valid range. The second attack is the *PGD* attack of Madry et al. [45], additionally constrained to perturb only the area covered by the eyeglasses. Specifically, we started from eyeglasses with solid colors and iteratively perturbed them for up to 100 iterations, while clipping the perturbations to have max-norm (i.e., L_∞-norm) of at most 0.12. We picked 100 and 0.12 as the maximum number of iterations and max-norm threshold, respectively, as these parameters led to the most powerful attack in prior work [45]. The *PGD* attack can be seen as a special case of the *CCS16* attack, as the two attacks follow approximately the same approach except that *PGD* focuses on a narrower search space by clipping perturbations more aggressively to decrease their perceptibility. As we show below, the success rate of *PGD* was significantly lower than of *CCS16*, suggesting that the clipping *PGD* performs may be too aggressive for this application. Therefore, we evaluated a third attack, denoted $\widehat{CCS16}$, in which we clipped perturbations, but did so less aggressively than *PGD*. In $\widehat{CCS16}$, we set the number of iterations to 300, as in the *CCS16* attack, and clipped perturbations to have max-norm of at most 0.47. We selected 0.47 as the max-norm threshold as it is the lowest threshold that led to success rates at fooling face recognition that are comparable to *CCS16*. In other words, this threshold gives the best chance of achieving inconspicuousness without substantially sacrificing the success rates. To maximize the success rates of the three attacks, we optimized only for the evasion objectives (defined via categorical cross-entropy), and ignored other objectives that are necessary to physically realize the attacks (specifically, generating colors that can be printed and ensuring smooth transitions between neighboring pixels [68]).

To test whether a detector would prevent attacks based on AGNs, we selected all ten subjects for VGG10 and 20 random subjects for VGG143, each with three images per subject. We then tested whether dodging and impersonation can be achieved while simultaneously evading the detector. To fool the detector along with the face-recognition DNNs, we slightly modified the objective from Equation (3) to optimize the adversarial generator such that the detector's loss is increased. In particular, the loss function we used was the difference between the probabilities of the correct class (either "adversarial" or "non-adversarial" input) and the incorrect class. As the vanilla *PGD*, *CCS16*, and $\widehat{CCS16}$ attacks either failed to achieve success rates comparable to AGNs or produced more conspicuous eyeglasses (Section 5.5), we did not extend them to fool the detector.

We measured the *success rate* of attacks via two metrics. For dodging, we measured the percentage of attacks in which the generator emitted eyeglasses that (1) led the image to be misclassified (i.e., the most probable class was not the attacker) and (2) kept the probability of the correct class below 0.01 (much lower than the thresholds set for accepting any of the DNNs' classifications; see Table 1). For impersonation, we considered an attack successful if the attacker's image was classified as the target with probability exceeding 0.92, the highest threshold used by any of the DNNs.

Table 3. Results of Attacks in the Digital Environment

	Without detector								*With detector*	
	Dodging				*Impersonation*				*Dodging*	*Impers.*
Model	*AGNs*	*PGD*	$\widehat{CCS16}$	*CCS16*	*AGNs*	*PGD*	$\widehat{CCS16}$	*CCS16*	*AGNs*	*AGNs*
VGG10	100±0%	37±10%	100±0%	100±0%	100±0%	10±5%	100±0%	100±0%	100±0%	100±0%
VGG143	100±0%	53±9%	100±0%	100±0%	88±5%	3±2%	82±5%	98±2%	100±0%	90±4%
OF10	100±0%	43±11%	100±0%	100±0%	100±0%	13±7%	87±9%	100±0%	—	—
OF143	100±0%	85±7%	100±0%	100±0%	90±4%	11±4%	78±6%	88±4%	—	—

We report the the mean success rate of attacks and the standard error when fooling the facial-recognition DNNs. To the right, we report the mean success rates and standard errors of dodging and impersonation using AGNs when simultaneously fooling facial-recognition DNNs and a detector (see Section 4.3).

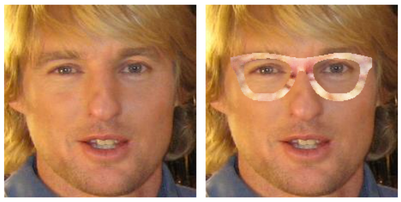


Fig. 5. An example of digital dodging. Left: An image of actor Owen Wilson (from the PubFig dataset [40]), correctly classified by VGG143 with probability 1.00. Right: Dodging against VGG143 using AGN's output (probability assigned to the correct class <0.01).

When using the detector, we also required that the detector deemed the input non-adversarial with probability higher than 0.5.

Experiment Results. Table 3 summarizes the results of the digital-environment experiments. All dodging attempts using AGNs succeeded; Figure 5 shows an example. As with dodging, all impersonation attempts using AGNs against the small DNNs (VGG10 and OF10) succeeded. A few attempts against the larger DNNs failed, suggesting that inconspicuous impersonation attacks may be more challenging when the DNN recognizes many subjects, although attacks succeeded at least 88% of the time.

Differently from AGNs, the success of *PGD* was limited—its success rates at dodging ranged from 37% to 85%, and its success rates at impersonation were below 13%. This suggests that the clipping performed by *PGD* may be too aggressive. The *CCS16* and $\widehat{CCS16}$ attacks achieved success rates comparable to AGNs at both dodging and impersonation. Unlike for AGNs, in this set of experiments the high success rates of the *CCS16* and $\widehat{CCS16}$ attacks were achieved by only attempting to dodge or impersonate (i.e., fool the classifier), while ignoring other objectives, such as generating colors that can be printed (previously achieved by minimizing the so-called non-printability score [68]). For physically realized attacks, evaluated in Section 5.2, satisfying these additional objectives is necessary for the *CCS16* and $\widehat{CCS16}$ attacks to succeed; there, however, measurements suggest that AGNs capture the inconspicuousness and realizability objectives more effectively.

Using a detector did not thwart the AGN attacks: Success rates for dodging and impersonation were similar to when a detector was not used. However, using a detector reduced the inconspicuousness of attacks (see Section 5.5).

We further tested whether attackers can be more successful by using eyeglasses of different shapes: We trained AGNs to generate eyeglasses of six new shapes and tested them against VGG143 and OF143. Three of the new shapes achieved comparable performance to the original shape (shown in Figure 5), but the overall success rates did not improve. Nevertheless, it would be useful to explore whether using variety of eyeglass shapes can enhance the inconspicuousness of attacks in practice.

5.2 Attacks in the Physical Domain

Attackers in the physical domain do not have complete control over the DNN's input: Slight changes in the attacker's pose, expression, distance from the camera, and illumination may dramatically change the concrete values of pixels. Practical attacks need to be robust against such changes. We took three additional measures to make the attacks more robust.

First, to train adversarial generators that emit images of eyeglasses that lead to more than one of the attacker's images to be misclassified, we used multiple images of the attacker in training the generator. Namely, we set X in Algorithm 1 to be a collection of the attacker's images. As a result, the generators learned to maximize $Loss_F$ for different images of the attacker.

Second, to make the attacks robust to changes in pose, we trained the adversarial generator to minimize $Loss_F$ over multiple images of the attacker wearing the eyeglasses. To align the eyeglasses to the attacker's face, we created and printed a three-dimensional (3D) model of eyeglasses with frames that have the same silhouette as the 2D eyeglasses emitted by the generator (using code from GitHub [11]). We added tracking markers—specifically positioned green dots—to the 3D-printed eyeglasses. The attacker wore the eyeglasses when capturing training data for the generator. We then used the markers to find a projective alignment, θ_x, of the eyeglasses emitted by the generator to the attacker's pose in each image. The generator was subsequently trained to minimize $Loss_F\left(x + \theta_x(G(z))\right)$ for different images of the attacker ($x \in X$).

Third, to achieve robustness to varying illumination conditions, we modeled how light intensity (luminance) affects eyeglasses and incorporated the models in AGN training. Specifically, we used the Polynomial Texture Maps approach [46] to estimate degree-3 polynomials that map eyeglasses' RGB values under baseline luminance to values under a specific luminance. In the forward pass of Algorithm 1, before digitally attaching eyeglasses to an attacker's image of certain luminance, we mapped the eyeglasses' colors to match the image's luminance. In the backward pass, the errors were back-propagated through the polynomials before being back-propagated through the generator to adjust its weights. In this way, the texture-map polynomials enabled us to digitally estimate the effect of lighting on the eyeglasses.

Experiment Setup. To evaluate the physically realized attacks, three subjects from our team acted as attackers: S_A (the first author), a Middle-Eastern male in his mid-20s; S_B (the third author), a white male in his 40s; and S_C (the second author), a South-Asian female in her mid-20s. Each subject attempted both dodging and impersonation against each of the four DNNs (which were trained to recognize them, among others). The data used for training and evaluating the physically realized attacks were collected from a room on Carnegie Mellon University's Pittsburgh campus using a Canon T4i camera. To control the illumination more accurately, we selected a room with a ceiling light but no windows on exterior walls.

In a first set of experiments, we evaluated the attacks under varied poses. To train the adversarial generators, we collected 45 images of each attacker (the set X in Algorithm 1) while he or she stood a fixed distance from the camera, kept a neutral expression, and moved his or her head up-down, left-right, and in a circle. Each generator was trained for at most one epoch, and training stopped earlier if the generator could emit eyeglasses that, for dodging, led the mean probability of the correct class to fall below 0.005, or, for impersonation, led the mean probability of the target class to exceed 0.990. For impersonation, we picked the target at random per attack.

To physically realize the attacks, we printed selected eyeglass patterns created by the generator on Epson Ultra Premium Glossy paper, using a commodity Epson XP-830 printer, and affixed them to the 3D-printed eyeglasses. Since each generator can emit a diverse set of eyeglasses, we (digitally) sampled 48 outputs (qualitatively, this amount seemed to capture the majority of patterns that the generators could emit) and kept the most successful one for dodging or impersonation in the digital environment (i.e., the one that led to the lowest mean probability assigned the attacker or the highest mean probability assigned to the target, respectively).

We evaluated the attacks by collecting videos of the attackers wearing the 3D-printed eyeglasses with the adversarial patterns affixed to their front. Again, the attackers were asked to stand a fixed distance from the camera, keep a neutral expression, and move their heads up-down, left-right, and in a circle. We extracted every third frame from each video. This resulted in 75 frames, on average, per attack. We then classified the extracted images using the DNNs targeted by the attacks. For dodging, we measured success by the fraction of frames that were classified as anybody but the attacker, and for impersonation by the fraction of frames that were classified as the target. In some cases, impersonation failed—mainly due to the generated eyeglasses not being realizable, as many of the pixels had extreme values (close to RGB = [0,0,0] or RGB = [1,1,1]). In such cases, we attempted to impersonate another (randomly picked) target.

We measured the head poses (i.e., pitch, yaw, and roll angles) of the attackers in training images using a state-of-the-art tool [7]. On average, head poses covered 13.01° of pitch (up-down direction), 17.11° of yaw (left-right direction), and 4.42° of roll (diagonal direction). This is similar to the mean difference in head pose between pairs of images randomly picked from the PubFig dataset (11.64° of pitch, 15.01° of yaw, and 6.51° of roll).

As a baseline to compare to, we repeated the dodging and impersonation attempts using our prior attack [68], referred to by *CCS16*. Unlike experiments in the digital domain, we did not evaluate variants of the *CCS16* attack where additional clipping is performed, as our experience in the digital domain showed that clipping harms the success rate of attacks and fails to improve their inconspicuousness (see Section 5.5).

In a second set of experiments, we evaluated the effects of changes to luminance. To this end, we placed a lamp (with a 150W incandescent light bulb) about 45° to the left of the attacker, and used a dimmer to vary the overall illuminance between $\sim 110lx$ and $\sim 850lx$ (comparable to difference between a dim corridor and a bright chain store interior [6]). We crafted the attacks by training the generator on 20 images of the attacker collected over five equally spaced luminance levels. In training the generator, we used the polynomial texture models as discussed above. For impersonation, we used the same targets as in the first set of experiments. We implemented the eyeglasses following the same procedure as before, then collected 40 video frames per attack, split evenly among the five luminance levels. In these experiments, the attackers again stood a fixed distance from the camera but did not vary their pose. For this set of experiments, we do not compare with our previous algorithm, as it was not designed to achieve robustness to changing luminance, and informal experiments showed that it performed poorly when varying the luminance levels.

Experiment Results. Table 4 summarizes our results and Figure 6 shows examples of attacks in the physical environment.

In the first set of experiments we varied the attackers' pose. Most dodging attempts succeeded with all video frames misclassified. Even in the worst attempt, 81% of video frames were misclassified. Overall, the mean probability assigned to the correct class was at most 0.40, much below the thresholds discussed in Section 4.3. For impersonation, one to four subjects had to be targeted before impersonations succeeded, with an average of 68% of video frames (mis)classified as the targets in successful impersonations. In two thirds of these attempts, >20% of frames were

Table 4. Summary of Physical Realizability Experiments

		Dodging results				Impersonation results						
DNN	*Subject*	*AGNs*	p(sub.)	*CCS16*	*AGNs-L*	*Target*	*Attempts*	*AGNs*	*HC*	p(tar.)	*CCS16*	*AGNs-L*
VGG10	S_A	100%	0.06	0%	93%	S_D	1	100%	74%	0.93	0%	80%
	S_B	97%	0.09	98%	100%	Milla Jovovich	2	88%	0%	0.70	63%	100%
	S_C	96%	0.10	100%	100%	Brad Pitt	1	100%	96%	0.98	100%	100%
VGG143	S_A	98%	0.17	0%	100%	Alicia Keys	2	89%	41%	0.73	0%	70%
	S_B	100%	<0.01	87%	100%	Ashton Kutcher	2	28%	0%	0.22	0%	0%
	S_C	100%	0.03	82%	100%	Daniel Radcliffe	2	3%	0%	0.04	0%	16%
OF10	S_A	81%	0.40	0%	83%	Brad Pitt	2	28%	23%	0.25	0%	50%
	S_B	100%	0.01	100%	100%	Brad Pitt	2	65%	55%	0.58	43%	100%
	S_C	100%	0.01	100%	100%	S_D	1	98%	95%	0.83	67%	38%
OF143	S_A	100%	0.09	36%	75%	Carson Daly	1	60%	0%	0.41	0%	73%
	S_B	97%	0.05	97%	100%	Aaron Eckhart	4	99%	25%	0.83	92%	100%
	S_C	100%	<0.01	99%	100%	Eva Mendes	2	53%	39%	0.67	3%	9%

For dodging, we report the success rate of AGNs (percentage of misclassified video frames), the mean probability assigned to the correct class (lower is better), the success rate of the *CCS16* attack [68], and the success rate of AGNs under luminance levels higher than the baseline luminance level (AGNs-L). For impersonation, we report the target (S_D is a member of our group, an Asian female in the early 20s), the number of targets attempted until succeeding, the success rate of AGNs (percentage of video frames classified as the target), the fraction of frames classified as the target with high confidence (HC; above the threshold which strikes a good balance between the true and the false positive rate), the mean probability assigned to the target (higher is better), the success rate of the *CCS16* attack [68], and the success rate under varied luminance levels excluding the baseline level (AGNs-L). Non-adversarial images of the attackers were assigned to the correct class.

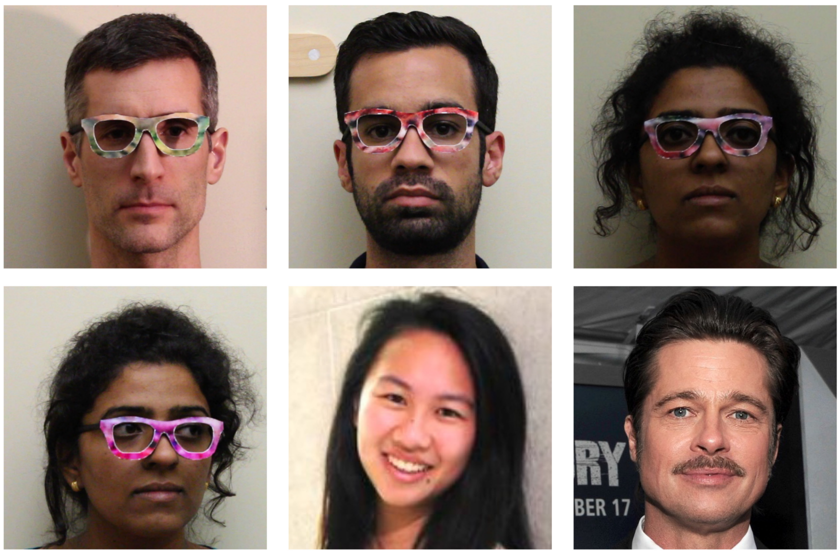


Fig. 6. Examples of physically realized attacks. (a) S_B (top) and S_C (bottom) dodging against OF143. (b) S_A impersonating S_D against VGG10. (c) S_C impersonating actor Brad Pitt (by Marvin Lynchard / CC BY 2.0 / cropped from https://goo.gl/Qnhe2X) against VGG10.

misclassified with high confidence (again, using the thresholds from Section 4.3). This suggests that even a conservatively tuned system would likely be fooled by some attacks.

We found that physical-domain evasion attempts using AGNs were significantly more successful than attempts using the *CCS16* algorithm. The mean success rate of dodging attempts was 45% higher when using AGNs compared to prior work (97% vs. 67%; a paired t-test shows that the difference is statistically significant with $p = 0.03$). The difference in success rates for impersonation was even larger. The mean success rate of impersonation attempts was 126% higher using AGNs compared to prior work (70% vs. 31%; paired t-test shows that the difference is statistically significant with $p < 0.01$). Given these results, we believe that AGNs provide a better approach to test the robustness of DNNs against physical-domain attacks than the *CCS16* algorithm.

The second set of experiments shows that AGNs can generate attacks that are robust to changes in luminance. On average, the dodging success rate was 96% (with most attempts achieving 100% success rate), and the impersonation success rate was 61%. Both are comparable to success rates under changing pose and fixed luminance. To evaluate the importance of modeling luminance to achieve robustness, we measured the success rate of S_A dodging against the four DNNs *without* modeling luminance effects. This caused the average success rate of attacks to drop from 88% (the average success rate of S_A at dodging when modeling luminance) to 40% (marginally significant according to a t-test, with $p = 0.06$). This suggests that modeling the effect of luminance when training AGNs is essential to achieve robustness to luminance changes.

Last, we built a mixed-effects logistic regression model [60] to analyze how different factors, and especially head pose and luminance, affect the success of physical-domain attacks. In the model, the dependent variable was whether an image was misclassified, and the independent variables accounted for the absolute value of pitch (up-down), yaw (left-right), and roll (tilt) angles of the head in the image (measured with Baltrušaitis et al.'s tool [7]); the luminance level (normalized to a [0,4] range); how close are the colors of the eyeglasses printed for the attack to colors that can be produced by our printer (measured via the *non-printability score* defined in our prior work [68], and normalized to a [0,1] range); the architecture of the DNN attacked (VGG or OpenFace); and the size of the DNN (10 or 143 subjects). The model also accounted for the interaction between angles and architecture, as well as the luminance and architecture.

To train the model, we used all the images we collected to test the attack in the physical domain. The model's R^2 is 0.70 (i.e., it explains 70% of the variance in the data), indicating a good fit. The parameter estimates are shown in Table 5. Luminance is not a statistically significant factor—i.e., the DNNs were equally likely to misclassify the images under the different luminance levels we considered. In contrast, the face's pose has a significant effect on misclassification. For the VGG networks, each degree of pitch or yaw away from 0° reduced the likelihood of success by 0.94, on average. Thus, an attacker who faced the camera at a pitch or yaw of ±10° was about 0.53 times less likely to succeed than when directly facing the camera. Differently from the VGG networks, for the OpenFace networks each degree of pitch away from 0° increased the likelihood of success by 1.12, on average. Thus, an attacker facing the camera at a pitch of ±10° was about 3.10 times more likely to succeed than when directly facing the camera. Overall, these results highlight the attacks' robustness to changes in luminance, as well as to small changes in pose away from frontal.

5.3 Universal Dodging Attacks

We next show that a small number of adversarial eyeglasses can allow successful dodging for the majority of subjects, even when images of those subjects are not used in training the adversarial generator.

We created the universal attacks by training the generator in Algorithm 1 on a set of images of different people. Consequently, the generator learned to emit eyeglasses that caused multiple

Table 5. Parameter Estimates for the Logistic Regression Model

Factor	lg(*odds*)	*odds*	*p-value*
(intercept)	<0.01	1.00	0.96
is.143.subjects.dnn	−0.48	0.62	<0.01
is.openface.dnn	6.34	568.78	<0.01
abs(pitch)	−0.06	0.94	<0.01
abs(yaw)	−0.06	0.94	<0.01
abs(roll)	0.01	1.01	0.80
luminance	0.04	1.04	0.12
non-printability	−1.09	0.34	<0.01
is.openface.dnn:luminance	0.38	1.48	0.14
is.openface.dnn:abs(pitch)	0.18	1.19	<0.01
is.openface.dnn:abs(yaw)	−0.08	0.92	0.06
is.openface.dnn:abs(roll)	−0.62	0.54	<0.01

Statistically significant factors are in bold.

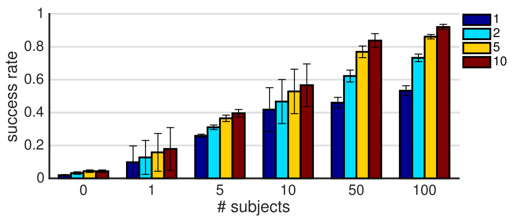
ALGORITHM 2: Universal attacks (given many subjects)

```
Input: $X, G, D, F(\cdot), dataset, Z, N_e, s_b, \kappa, s_c$
Output: $Gens$ // a set of generators
1 $Gens \leftarrow \{\}$;
2 $clusters \leftarrow$ clusters of size $s_c$ via k-means++;
3 for $cluster \in clusters$ do
4     $G \leftarrow Alg1(cluster, G, D, F, dataset, Z, N_e, s_b, \kappa)$;
5     $Gens \leftarrow Gens \cup \{G\}$;
6 return $Gens$;
```

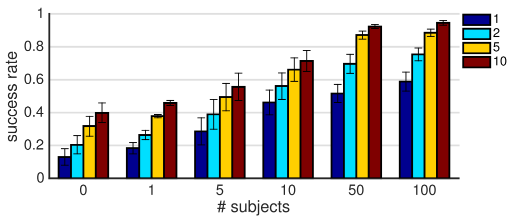
people's images to be misclassified, not only one person's. We found that when the number of subjects was large, the generator started emitting conspicuous patterns that did not resemble real eyeglasses. For such cases, we used Algorithm 2, which builds on Algorithm 1 to train several adversarial generators, one per cluster of similar subjects. Algorithm 2 uses *k-means++* [3] to create clusters of size s_c. Clustering was performed in Euclidian space using the features extracted from the base DNNs (4,096-dimensional features for VGG, and 128-dimensional features for OpenFace; see Section 4.3). The result was a set of generators that create eyeglasses that, cumulatively, (1) led to the misclassification of a large fraction of subjects and (2) appeared more inconspicuous (as judged by members of our team) than when training on all subjects combined. The key insight behind the algorithm is that it may be easier to find inconspicuous universal adversarial eyeglasses for similar subjects than for vastly different subjects.

Experiment Setup. We tested the universal attacks against VGG143 and OF143 only, as the other DNNs were trained with too few subjects to make meaningful conclusions. To train and evaluate the generators, we selected two images for each of the subjects the DNNs were trained on—one image for training and one image for testing. To make dodging more challenging, we selected the two images that were classified correctly with the highest confidence by the two networks. Specifically, we selected images such that the product of the probabilities both DNNs assigned to the correct class was the highest among all the available images.

To explore how the number of subjects used to create the universal attacks affected performance, we varied the number of (randomly picked) subjects with whose images we trained the adversarial



(a) Universal dodging against VGG143.



(b) Universal dodging against OF143.

Fig. 7. Universal dodging against VGG143 and OF143. The x-axis shows the number of subjects used to train the adversarial generators. When the number of subjects is zero, a non-adversarial generator was used. The y-axis shows the mean fraction of images misclassified (i.e., the dodging success rate). The whiskers on the bars show the standard deviation of the success rate, computed by repeating each experiment five times, each time with a different set of randomly picked subjects. The color of the bars denotes the number of eyeglasses used, as shown in the legend. We evaluated each attack using 1, 2, 5, or 10 eyeglasses. For example, the rightmost bar in Figure 7(b) indicates that an AGN trained with images of 100 subjects will generate eyeglasses such that 10 pairs of eyeglasses will allow approximately 94% of subjects to evade recognition. For ≤10 subjects, Algorithm 1 was used to create the attacks. For 50 and 100 subjects, Algorithm 2 was used.

generators. We averaged the success rate after repeating the process five times (each time selecting a random set of subjects for training). When using ≥50 subjects for the universal attacks, we used Algorithm 2 and set the cluster size to 10.

Additionally, we explored how the number of adversarial eyeglasses affected the success of the attack. We did so by generating 100 eyeglasses from each trained generator or set of generators and identifying the subsets (of varying size) that led the largest fraction of images in the test set to be misclassified. Finding the optimal subsets is NP-hard, and so we used an algorithm that provides a $(1 - \frac{1}{e})$-approximation of the optimal success rate [55].

Experiment Results. Figure 7 summarizes the results. Universal attacks are indeed possible: generators trained to achieve dodging using a subset of subjects produced eyeglasses that led to dodging when added to images of subjects not used in training. The effectiveness of dodging depends chiefly on the number of subjects used in training and, secondarily, the number of eyeglasses generated. In particular, training a generator (set) on 100 subjects and using it to create 10 eyeglasses was sufficient to allow 92% of remaining subjects to dodge against VGG143 and 94% of remaining subjects to dodge against OF143. Even training on five subjects and generating five eyeglasses was sufficient to allow more than 50% of the remaining users to dodge against either network. OF143 was particularly more susceptible to universal attacks than VGG143 when a small number of subjects was used for training, likely due to its overall lower accuracy.

5.4 Transferability of Dodging Attacks

Transferability of attacks has been shown to be effective in fooling models to which adversaries do not have access (e.g., Reference [58]). In our case, although this is not an explicit goal of our attacks, attackers with access to one DNN but not another may attempt to rely on transferability to dodge against the second DNN. In this section, we explore whether dodging against DNNs of one architecture leads to successful dodging against DNNs of a different architecture.

Using the data from Section 5.1, we first tested whether dodging in the digital environment successfully transferred between architectures (see Table 6). We found that attacks against the OpenFace architecture successfully fooled the VGG architecture in only a limited number of

Table 6. Transferability of Dodging in the Digital Domain

To From	VGG10	OF10
VGG10	—	63.33%
OF10	10.00%	—

To From	VGG143	OF143
VGG143	—	88.33%
OF143	11.67%	—

Each table shows how likely it is for a generator used for dodging against one network (rows) to succeed against another network (columns).

Table 7. Transferability of Dodging in the Physical Domain

To From	VGG10	OF10
VGG10	—	43.84%
OF10	27.77%	—

To From	VGG143	OF143
VGG143	—	51.78%
OF143	19.86%	—

We classified the frames from the physically realized attacks using DNNs different from the ones for which the attacks were crafted. Each table shows how likely it is for frames that successfully dodged against one network (rows) to succeed against another network (columns).

attempts (10–12%). In contrast, dodging against VGG led to successful dodging against OpenFace in at least 63% of attempts.

Universal attacks seemed to transfer between architectures with similar success. Using attacks created with 100 subjects and 10 eyeglasses from Section 5.3, we found that 82% (±3% standard deviation) of attacks transferred from VGG143 to OF143, and 26% (±4% standard deviation) transferred in the other direction.

The transferability of dodging attacks in the physical environment between architectures followed a similar trend (see Table 7). Successful attacks transferred less successfully from the OpenFace networks to the VGG networks (20–28%) than in the other direction (44–52%).

5.5 A Study to Measure Inconspicuousness

Methodology. To evaluate inconspicuousness of eyeglasses generated by AGNs we carried out an online user study. Participants were told that we were developing an algorithm for designing eyeglass patterns, shown a set of eyeglasses, and asked to label each pair as either algorithmically generated or real. Each participant saw 15 "real" and 15 attack eyeglasses in random order. All eyeglasses were the same shape and varied only in their coloring. The "real" eyeglasses were ones used for pretraining the AGNs (see Section 4.1). The attack eyeglasses were generated using either AGNs, the *CCS16* attack, or the $\widehat{\mathit{CCS16}}$ attack.

Neither "real" nor attack eyeglasses shown to participants were photo-realistically or three-dimensionally rendered. So, we consider attack glasses to have been inconspicuous to participants not if they were uniformly rated as real (which even "real" glasses were not, particularly when attack glasses were inconspicuous), but rather if the rate at which participants deemed them as real does not differ significantly regardless of whether they are "real" eyeglasses or attack eyeglasses.

Given two sets of eyeglasses (e.g., a set of attack glasses and a set of "real" glasses), we tested whether one is more inconspicuous via the χ^2 test of independence [47], and conservatively corrected for multiple comparisons using the Bonferroni correction. We compared the magnitude of differences using the odds-ratio measure: The odds of eyeglasses in the first group being marked as real divided by the odds of eyeglasses in the second group being marked as real. The higher (respectively, lower) the odds ratios are from 1, the higher (respectively, lower) was the likelihood

Table 8. Relative Realism of Selected Sets of Eyeglasses

Comparison (group 1 vs. group 2)		*Odds ratio*	*p-value*
Real (61%)	All AGNs (47%)	1.71	<0.01
AGNs digital (49%)	*CCS16* digital [68] (31%)	2.19	<0.01
AGNs digital (49%)	$\widehat{\mathit{CCS16}}$ digital (30%)	2.24	<0.01
AGNs physical (45%)	*CCS16* physical [68] (34%)	1.59	<0.01
AGNs:			
digital (49%)	physical (45%)	1.19	0.26
digital (49%)	digital with detector (43%)	1.28	0.02
digital dodging (52%)	universal dodging (38%)	1.80	<0.01

For each two sets compared, we report in parentheses the fraction of eyeglasses per set that were marked as real by study participants, the odds ratios between the groups, and the p-value of the χ^2 test of independence; e.g., odds ratio of 1.71 means that eyeglasses are ×1.71 as likely to be selected as real if they are in the first set than if they are in the second.

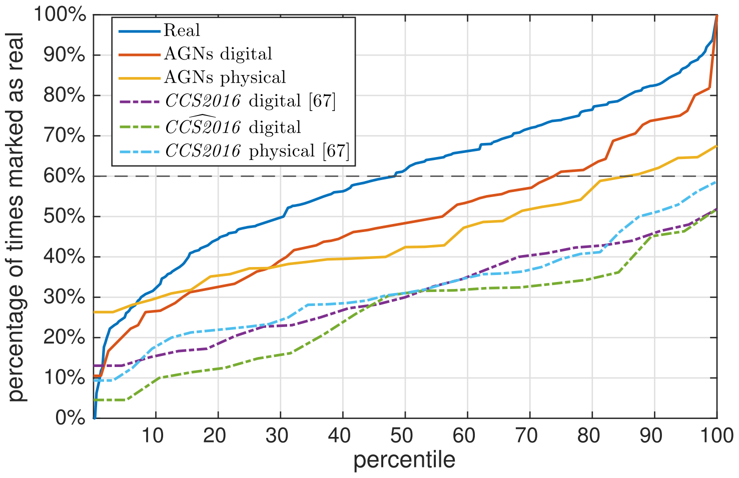


Fig. 8. The percentage of times in which eyeglasses from different sets were marked as real. The horizontal 60% line is highlighted to mark that the top half of "real" eyeglasses were marked as real at least 60% of the time.

that eyeglasses from the first group were selected as real compared to eyeglasses from the second group.

We recruited 301 participants in the U.S. through the Prolific crowdsourcing service.[5] Their ages ranged from 18 to 73, with a median of 29. 51% of participants specified being female and 48% male (1% chose other or did not answer). Our study took 3 minutes to complete on average and participants were compensated $1.50. The study design was approved by Carnegie Mellon University's ethics review board.

Results. Table 8 and Figure 8 show comparisons between various groups of eyeglasses, as well as the percentage of time participants marked different eyeglasses as real. "Real" eyeglasses were more realistic than AGN-generated ones (×1.71 odds ratio). This is expected, given the additional

[5] https://prolific.ac.

objectives that attack eyeglasses are required to achieve. However, AGNs were superior to other attacks. Both for digital and physical attacks, eyeglasses created by AGNs were more realistic than those created by our previous *CCS16* attack [68] (×2.19 and ×1.59 odds ratio, respectively). Even limiting the max-norm of the perturbations did not help—AGNs generated eyeglasses that were more likely to be selected as real than the $\widehat{\mathit{CCS16}}$ attack (×2.24 odds ratio).

Perhaps most indicative of inconspicuousness in practice is that many AGN-generated eyeglasses were as realistic as "real" eyeglasses. The most inconspicuous 26% of eyeglasses emitted by AGNs for digital-environment attacks were on average deemed as real as the most inconspicuous 50% of "real" eyeglasses; in each case participants marked these eyeglasses as real >60% of the time. Physical attacks led to less inconspicuous eyeglasses; however, the 14% most inconspicuous were still marked as real at least 60% of the time (i.e., as real as the top 50% of "real" eyeglasses).

Other results match intuition—the more difficult the attack, the bigger the impact on conspicuousness. Digital attack glasses that do not try to fool a detector are less conspicuous than ones that fool a detector (×1.28 odds ratio), and individual dodging is less conspicuous than universal dodging (×1.80 odds ratio). Digital attack glasses had higher odds of being selected as real than physical attack glasses (×1.19 odds ratio), but the differences were not statistically significant ($p\text{-}value = 0.26$).

5.6 AGNs against Digit Recognition

We next show that AGNs can be used in domains besides face recognition. Specifically, we use AGNs to fool a state-of-the-art DNN for recognizing digits, trained on the MNIST dataset [43], which contains 70,000 28×28-pixel images of digits.

Experiment Setup. First, we trained a DNN for digit recognition using the architecture and training code of Carlini and Wagner [13]. We trained the DNN on 55,000 digits and used 5,000 for validation during training time. The trained DNN achieved 99.48% accuracy on the test set of 10,000 digits. Next, we pretrained 10 GANs to generate digits, one for each digit. Each generator was trained to map inputs randomly sampled from $[-1, 1]^{25}$ to 28×28-pixel images of digits. We again used the Deep Convolutional GAN architecture [62]. Starting from the pretrained GANs, we trained AGNs using a variant of Algorithm 1 to produce generators that emit images of digits that simultaneously fool the discriminator to be real and are misclassified by the digit-recognition DNN.

Unlike prior attacks, which typically attempted to minimally perturb specific benign inputs to cause misclassification (e.g., References [13, 19, 27, 57, 61, 77]), the attack we propose does not assume that a benign input is provided, nor does it attempt to produce an attack image minimally different from a benign image. Hence, a comparison with prior attacks would not be meaningful.

Experiment Results. The AGNs were able to output arbitrarily many adversarial examples that appear comprehensible to human observers, but are misclassified by the digit-recognition DNN (examples are shown in Figure 9). As a test, we generated 5,004 adversarial examples that all get misclassified by the digit-recognition DNN. The adversarial examples were produced by first generating 600,000 images using the adversarial generators (60,000 per generator). Out of all samples, the ones that were misclassified by the DNN (8.34% of samples) were kept. Out of these, only the digits that were likely to be comprehensible by humans were kept: the automatic filtering process to identify these involved computing the product of the discriminator's output (i.e., how realistic the images were deemed by the discriminator) and the probability assigned by the digit-recognition DNN to the correct class and keeping the 10% of digits with the highest product.

Differently from traditional attacks on digit recognition (e.g., Reference [13]), these attack images are not explicitly designed for minimal deviation from specific benign inputs; rather, their

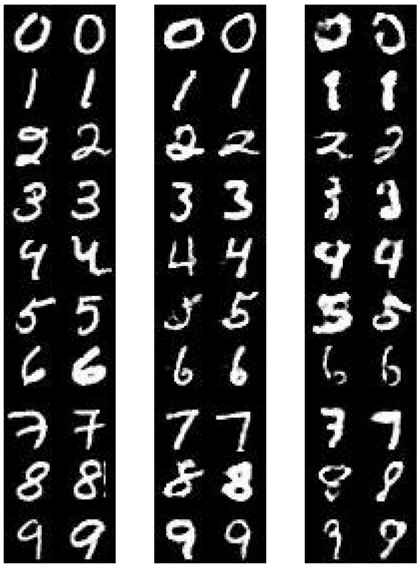


Fig. 9. An illustration of attacks generated via AGNs. Left: A random sample of digits from MNIST. Middle: Digits generated by the pretrained generator. Right: Digits generated via AGNs that are misclassified by the digit-recognition DNN.

advantage is that they can be substantially different (e.g., in Euclidean distance) from training images. We measured the diversity of images by computing the mean Euclidean distance between pairs of digits of the same type; for attack images, the mean distance was 8.34, while for the training set it was 9.25.

A potential way AGNs can be useful in this domain is adversarial training. For instance, by augmenting the training set with the 5,004 samples, one can extend it by almost 10%. This approach can also be useful for visualizing inputs that would be misclassified by a DNN, but are otherwise not available in the training or testing sets.

6 DISCUSSION AND CONCLUSION

In this article, we contributed a methodology that we call AGNsto generate adversarial examples to fool DNN-based classifiers while meeting additional objectives. We focused on objectives imposed by the need to physically realize artifacts that, when captured in an image, result in misclassification of the image. Using the physical realization of eyeglass frames to fool face recognition as our driving example, we demonstrated the use of AGNs to improve robustness to changes in imaging conditions (lighting, angle, etc.) and even to specific defenses; inconspicuousness to human onlookers; and scalability in terms of the number of adversarial objects (eyeglasses) needed to fool DNNs in different contexts. AGNs generated adversarial examples that improved upon prior work in all of these dimensions, and did so using a general methodology.

Our work highlights a number of features of AGNs. They are flexible in their ability to accommodate a range of objectives, including ones that elude precise specification, such as inconspicuousness. In principle, given an objective that can be described through a set of examples, AGNs can be trained to emit adversarial examples that satisfy this objective. Additionally, AGNs are general in being applicable to various domains, which we demonstrated by training AGNs to fool classifiers for face and (handwritten) digit recognition. We expect that they would generalize to other applications, as well. For example, one may consider using AGNs to fool DNNs for street-sign recognition by training the generator to emit adversarial examples that resemble street-sign images collected from the internet.

One advantage of AGNs over other attack methods (e.g., References [27, 73]) is that they can generate multiple, diverse, adversarial examples for a given benign sample. A diverse set of adversarial examples can be useful for evaluating the robustness of models. Moreover, such adversarial examples may be used to defend against attacks (e.g., by incorporating them in adversarial training [41]).

ACKNOWLEDGMENT

The authors thank Cara Bloom for her comments on an early draft of this article.

REFERENCES

[1] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. 2016. *OpenFace: A General-purpose Face Recognition Library with Mobile Applications*. Technical Report. CMU-CS-16-118, CMU School of Computer Science.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein GAN. In *Proceedings of the International Conference on Machine Learning (ICML'17)*.

[3] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'07)*.

[4] Anish Athalye and Nicholas Carlini. 2018. On the robustness of the CVPR 2018 white-box adversarial example defenses. *arXiv:1804.03286* (2018).

[5] Anish Athalye, Nicholas Carlini, and David Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the International Conference on Machine Learning (ICML'18)*.

[6] Autodesk. [n.d.]. Measuring light levels. Retrieved from https://goo.gl/hkBWbZ.

[7] Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency. 2016. Openface: An open source facial behavior analysis toolkit. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV'16)*. DOI: https://doi.org/10.1109/WACV.2016.7477553

[8] Shumeet Baluja and Ian Fischer. 2018. Learning to attack: Adversarial transformation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[9] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recogn.* 84 (2018), 317–331. DOI: https://doi.org/10.1016/j.patcog.2018.07.023

[10] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013. Evasion attacks against machine learning at test time. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'13)*. DOI: https://doi.org/10.1007/978-3-642-40994-3_25

[11] CaretDashCaret. [n.d.]. 3D printable frames from eyeglasses SVGs. Retrieved from https://github.com/caretdashcaret/pince-nez.

[12] Nicholas Carlini and David Wagner. 2017. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec'17)*. DOI: https://doi.org/10.1145/3128572.3140444

[13] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'17)*. DOI: https://doi.org/10.1109/SP.2017.49

[14] Nicholas Carlini, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, and Wenchao Zhou. 2016. Hidden voice commands. In *Proceedings of the USENIX Security Symposium*.

[15] Yizheng Chen, Yacin Nadji, Athanasios Kountouras, Fabian Monrose, Roberto Perdisci, Manos Antonakakis, and Nikolaos Vasiloglou. 2017. Practical attacks against graph-based clustering. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. DOI: https://doi.org/10.1145/3133956.3134083

[16] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. 2017. Houdini: Fooling deep structured prediction models. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'17)*.

[17] Simon Eberz, Nicola Paoletti, Marc Roeschlin, Marta Kwiatkowska, I. Martinovic, and A. Patané. 2017. Broken hearted: How to attack ECG biometrics. In *Proceedings of the ISOC Annual Network and Distributed System Security Symposium (NDSS'17)*.

[18] Alexei A. Efros and Thomas K. Leung. 1999. Texture synthesis by non-parametric sampling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'99)*. DOI: https://doi.org/10.1109/ICCV.1999.790383

[19] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. 2017. A rotation and a translation suffice: Fooling CNNs with simple transformations. In *Proceedings of the NeurIPS Machine Learning and Computer Security Workshop*.

[20] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. 2018. Robust physical-world attacks on machine learning models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. DOI: https://doi.org/10.1109/CVPR.2018.00175

[21] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. 2016. Robustness of classifiers: From adversarial to random noise. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'16)*.

[22] Reuben Feinman, Ryan R. Curtin, Saurabh Shintre, and Andrew B. Gardner. 2017. Detecting adversarial samples from artifacts. *arXiv:1703.00410* (2017).

[23] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'15)*. DOI: https://doi.org/10.1145/2810103.2813677

[24] Justin Gilmer, Ryan P. Adams, Ian Goodfellow, David Andersen, and George E. Dahl. 2018. Motivating the rules of the game for adversarial example research. *arXiv:1807.06732* (2018).

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'14)*.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

[27] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR'15)*.

[28] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv:1702.06280* (2017).

[29] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. 2017. Adversarial perturbations against deep neural networks for malware classification. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS'17)*.

[30] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. 2018. Countering adversarial images using input transformations. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

[31] Adam Harvey. 2010. *CV Dazzle: Camouflage from face detection*. Master's thesis. New York University. Retreieved from http://cvdazzle.com.

[32] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Technical Report 07-49. University of Massachusetts, Amherst.

[33] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. 2015. Learning with a strong adversary. *arXiv:1511.03034* (2015).

[34] Lucas Introna and Helen Nissenbaum. 2009. *Facial Recognition Technology: A Survey of Policy and Implementation Issues*. Technical Report. Center for Catastrophe Preparedness and Response, New York University.

[35] Can Kanbak, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. 2018. Geometric robustness of deep networks: Analysis and improvement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. DOI: https://doi.org/10.1109/CVPR.2018.00467

[36] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. 2018. Adversarial logit pairing. *arXiv:1803.06373* (2018).

[37] Alex Kantchelian, J. D. Tygar, and Anthony D. Joseph. 2016. Evasion and hardening of tree ensemble classifiers. In *Proceedings of the International Conference on Machine Learning (ICML'16)*.

[38] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR'15)*.

[39] J. Zico Kolter and Eric Wong. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the International Conference on Machine Learning (ICML'18)*.

[40] Neeraj Kumar, Alexander C. Berg, Peter N. Belhumeur, and Shree K. Nayar. 2009. Attribute and simile classifiers for face verification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'09)*. DOI: https://doi.org/10.1109/ICCV.2009.5459250

[41] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

[42] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *Proceedings of the International Conference on Learning Representations Workshop (ICLRW'17)*.

[43] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. [n.d.]. The MNIST database of handwritten digits. Retrieved from http://yann.lecun.com/exdb/mnist/.

[44] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Jun Zhu, and Xiaolin Hu. 2018. Defense against adversarial attacks using high-level representation guided denoiser. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. DOI: https://doi.org/10.1109/CVPR.2018.00191

[45] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

[46] Tom Malzbender, Dan Gelb, and Hans Wolters. 2001. Polynomial texture maps. In *Proceedings of the ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'01)*. DOI: https://doi.org/10.1145/383259.383320

[47] Mary L. McHugh. 2013. The Chi-square test of independence. *Biochem. Med.* 23, 2 (2013), 143–149.

[48] Dongyu Meng and Hao Chen. 2017. MagNet: A two-pronged defense against adversarial examples. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. DOI: https://doi.org/10.1145/3133956.3134057

[49] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. In *Proceedings of the International Conference on Learning Representations (ICLR'17)*.

[50] Matthew Mirman, Timon Gehr, and Martin Vechev. 2018. Differentiable abstract interpretation for provably robust neural networks. In *Proceedings of the International Conference on Machine Learning (ICML'18)*.

[51] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. 2016. Distributional smoothing with virtual adversarial training. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[52] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. DOI: https://doi.org/10.1109/CVPR.2016.282

[53] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*.

[54] Nina Narodytska and Shiva Kasiviswanathan. 2017. Simple black-box adversarial attacks on deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'17)*. DOI: https://doi.org/10.1109/CVPRW.2017.172

[55] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. 1978. An analysis of approximations for maximizing submodular set functions. *Math. Program.* 14, 1 (1978), 265–294. DOI: https://doi.org/10.1007/BF01588971

[56] Jorge Nocedal. 1980. Updating quasi-Newton matrices with limited storage. *Math. Comp.* 35, 151 (1980), 773–782. DOI: https://doi.org/10.1090/S0025-5718-1980-0572855-7

[57] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'16)*. DOI: https://doi.org/10.1109/EuroSP.2016.36

[58] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM Symposium on Information, Computer and Communications Security (AsiaCCS'17)*. DOI: https://doi.org/10.1145/3052973.3053009

[59] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. 2015. Deep face recognition. In *Proceedings of the British Machine Vision Conference (BMVC'15)*. DOI: https://doi.org/10.5244/C.29.41

[60] Jose Pinheiro, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, and R Core Team. 2015. *nlme: Linear and Nonlinear Mixed Effects Models.* R package version 3.1–122.

[61] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. 2018. Generative adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. DOI: https://doi.org/10.1109/CVPR.2018.00465

[62] Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[63] Andras Rozsa, Ethan M. Rudd, and Terrance E. Boult. 2016. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'16)*. DOI: https://doi.org/10.1109/CVPRW.2016.58

[64] Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J. Fleet. 2016. Adversarial manipulation of deep representations. In *Proceedings of the International Conference on Learning Representations (ICLR'16)*.

[65] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved techniques for training gans. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'16)*.

[66] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

[67] Adi Shamir, Itay Safran, Eyal Ronen, and Orr Dunkelman. 2019. A simple explanation for the existence of adversarial examples with small hamming distance. *arXiv:1901.10861* (2019).

[68] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*. DOI: https://doi.org/10.1145/2976749.2978392

[69] Vignesh Srinivasan, Arturo Marban, Klaus-Robert Müller, Wojciech Samek, and Shinichi Nakajima. 2018. Counterstrike: Defending deep learning architectures against adversarial samples by Langevin dynamics with supervised denoising autoencoder. *arXiv:1805.12017* (2018).

[70] Nedim Srndic and Pavel Laskov. 2014. Practical evasion of a learning-based classifier: A case study. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P'14)*. DOI: https://doi.org/10.1109/SP.2014.20

[71] K. O. Stanley. 2007. Compositional pattern producing networks: A novel abstraction of development. *Gen. Program. Evolv. Mach.* 8, 2 (2007), 131–162. DOI: https://doi.org/10.1007/s10710-007-9028-8

[72] Yaron Steinbuch. 2017. JetBlue ditching boarding passes for facial recognition. *New York Post* (May 31 2017).

[73] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR'14)*.

[74] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. 2014. DeepFace: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*. DOI: https://doi.org/10.1109/CVPR.2014.220

[75] A. Vedaldi and K. Lenc. 2015. MatConvNet—Convolutional neural networks for MATLAB. In *Proceedings of the Annual ACM Conference on Multimedia (MM'15)*. DOI: https://doi.org/10.1145/2733373.2807412

[76] David Warde-Farley and Ian Goodfellow. 2016. Adversarial perturbations of deep neural networks. *Perturbations, Optimization, and Statistics*, Tamir Hazan, George Papandreou, and Daniel Tarlow (Eds.). MIT Press, 1–32.

[77] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. 2018. Generating adversarial examples with adversarial networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'18)*. DOI: https://doi.org/10.24963/ijcai.2018/543

[78] Weilin Xu, Yanjun Qi, and David Evans. 2016. Automatically evading classifiers. In *Proceedings of the ISOC Annual Network and Distributed System Security Symposium (NDSS'16)*.

[79] Takayuki Yamada, Seiichi Gohshi, and Isao Echizen. 2013. Privacy Visor: Method based on light absorbing and reflecting properties for preventing face image detection. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'13)*. DOI: https://doi.org/10.1109/SMC.2013.271

[80] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'14)*.

[81] Guoming Zhang, Chen Yan, Xiaoyu Ji, Taimin Zhang, Tianchen Zhang, and Wenyuan Xu. 2017. DolphinAttack: Inaudible voice commands. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*. DOI: https://doi.org/10.1145/3133956.3134052

[82] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*.

Received April 2018; revised October 2018; accepted March 2019