Adversarial Generative Nets: Neural Network Attacks on State-of-the-Art Face Recognition

Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer
Carnegie Mellon University
{mahmoods, srutib, lbauer}@cmu.edu

Michael K. Reiter
University of North Carolina at Chapel Hill
reiter@cs.unc.edu

1.看起来不明显
2.对目前的防御，此攻击方法鲁棒性较高（主要是在观察角度方面鲁棒性高）
3.可拓展性很高，比如可以创造个通用的眼镜

***Abstract*—In this paper we show that misclassification attacks against face-recognition systems based on deep neural networks (DNNs) are more dangerous than previously demonstrated, even in contexts where the adversary can manipulate only her physical appearance (versus directly manipulating the image input to the DNN). Specifically, we show how to create eyeglasses that, when worn, can succeed in targeted (impersonation) or untargeted (dodging) attacks while improving on previous work in one or more of three facets: (i) inconspicuousness to onlooking observers, which we test through a user study; (ii) robustness of the attack against proposed defenses; and (iii) scalability in the sense of decoupling eyeglass creation from the subject who will wear them, i.e., by creating "universal" sets of eyeglasses that facilitate misclassification. Central to these improvements are *adversarial generative nets*, a method we propose to generate physically realizable attack artifacts (here, eyeglasses) automatically.**

I. 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 [25], [60]. 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 [59].

While this attack and almost all improvements of it (see Sec. II-A) operate by directly manipulating the image provided as input to the DNN, numerous scenarios preclude an attacker doing so (e.g., face recognition used in lieu of boarding passes at airports [58]). Our prior work, however, showed that attacks can be mounted *physically*, i.e., by creating an adversarial artifact that fools a DNN presented with an unmodified image containing it [56]. Specifically, this work showed how to physically render eyeglasses that a person can don to fool DNN-based face recognition. While disquieting under any circumstances, such physical attacks would be especially of concern if enhanced to have the following properties.

- An attack is *robust* if it succeeds despite changes to imaging conditions and/or the deployment of defenses. Robust attacks are desirable for the attacker when any of these factors is outside his control. Our prior work showed that attacks can be robust to minor changes in viewing angle but did not consider other types of robustness.
- An attack is *inconspicuous* if in the context of where and how the adversarial object is presented to the camera, it is unlikely to arouse the suspicions of human onlookers. Our prior work conjectured that attacks could be made inconspicuous but did not show how to do so.
- The attack is *scalable* if relatively few adversarial objects are sufficient to fool DNNs in many contexts. In contrast, an attack that requires a new adversarial object to be generated per subject (e.g., a new pair of eyeglasses, in the case of our prior work) is not scalable.

In this paper we advance physically realized attacks along all three of these dimensions. A key ingredient and novelty of our attacks is that unlike most attack algorithms, which produce a single input that may be misclassified by a machine-learning algorithm, our approach builds on recent work on generative adversarial networks (GANs) to develop a neural network whose outputs are physically realizable attack instances. We call such attack networks *adversarial generative nets (AGNs)*. We use AGNs that target two of the DNN-based based face-recognition algorithms that achieve human-level accuracy—VGG [49] and OpenFace [1]—to output eyeglasses that enable an attacker to either evade correct recognition or to impersonate a specific target inconspicuously. To confirm 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 [35].

攻击对象

We show empirically that such AGNs, trained in a white-box setting where they have access to the trained network being attacked (a typical assumption; see, e.g., [47]), can produce attacks that have several concerning characteristics beyond simple physical realizability. Specifically, we demonstrate that:

- AGNs can create eyeglasses that appear as inconspicuous to a human as normal eyeglasses, i.e., eyeglasses designed with no adversarial intent. We show through a user study that, for example, that the most inconspicuous 30% of glasses generated by AGNs are as inconspicuous as the top 50% of real eyeglasses.
- AGNs can produce attacks that could be deployed at scale. In particular, we demonstrate attacks that we call *universal* because they produce a few (ten or less) maliciously designed eyeglasses using which a large fraction

of the population—not just one attacker—can evade face recognition. Specifically, we show that approximately five AGN-generated eyeglasses are sufficient to allow about 90% of the population to evade detection.

- AGNs produce attacks that are robust against some defenses. In particular, we enhance the VGG and OpenFace DNNs by augmenting training with labeled "attack" images (cf. [34]) and a detector designed to determine if an input is an attack (cf. [39]). While these defenses, particularly the detector, are very effective against previous attacks [56] and AGNs that are not aware of them, AGNs aware of the defenses produce attack images with a similar success rate as to when no defense is deployed. We also show experimentally that the attacks produced by AGNs are robust to different angles of observation.

A notable property of AGNs is that a trained AGN can efficiently generate a large number of *diverse* adversarial examples. 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.

Taken together, these results dramatically advance the state-of-the-art in physically realized attacks on DNNs, specifically in the areas of inconspicuousness, robustness, and scalability.

This paper proceeds as follows. We first review related work (Sec. II). We then describe AGNs, our novel attack method (Sec. III) and how we instantiate them for generating attacks against specific DNNs (Sec. IV). Next, we evaluate the effectiveness of our AGNs, including in a digital environment, with physically realized attacks, and with a user study to examine conspicuousness (Sec. V). Finally, we discuss other potential applications for AGNs and conclude (Sec. VI).

II. Related Work

In this section, we present prior work on attacks targeting neural networks and proposed defenses. Then, we discuss physical-world attacks on machine-learning systems.

A. Fooling Neural Networks

Szegedy et al. showed how adversarial examples to fool DNNs can be systematically found [59]. Given an input x that is classified to $F(x)$ by the DNN, their 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, the perturbation can be found by formalizing the problem as an optimization, which can be solved using the L-BFGS solver [45].

More efficient algorithms were later proposed for finding even more imperceptible adversarial examples using different notions of imperceptibility [8], [13], [20], [26], [28], [40], [42], [47], [53], [54]. For example, Papernot et al.'s algorithm aims to minimize the number of pixels changed [47]. Carlini and Wagner experimented with different formulations of the optimization's objective function for finding adversarial examples [8]. 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. $Loss_{cw}$ is not directly defined 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:

$$Loss_{cw} = \max\{L(x+r)_c : c \neq c_t\} - L(x+r)_{c_t}$$

Minimizing $Loss_{cw}$ increases the probability of the target class, c_t, while decreasing the probability of other classes.

Perhaps closest to our work is the work of Baluja and Fischer [5]. They propose to train an auto-encoding neural network that takes an image as its input and outputs a perturbed version of the same image that would be misclassified as a target class. Follow-up research efforts that are concurrent to ours propose to train generative neural networks to create adversarially perturbed images (or just perturbations to be later added to benign images) that can lead to misclassification [51], [67]. These techniques require that the perturbation has a small norm and allow it to cover the entire image. In contrast, as we discuss in Sec. III–IV, the attacks that we propose cover a small portion of the image, and are designed to be realizable and to resemble eyeglass designs that can be found online.

Moosavi et al. showed how to find universal adversarial perturbations [41]. A universal perturbation is one that leads not just one image to be misclassified, but a large set of images. Universal perturbations improve our understanding of DNNs' limitations, as they show that adversarial examples often lie in fixed directions (in the image RGB space) with respect to their corresponding benign inputs. Differently from their work, we explore the existence of universal attacks that are both inconspicuous and constrained to a small region. Moreover, we show that it is possible to find such attacks with many fewer training examples than was used before.

Research suggests that adversarial examples are not a result of overfitting, as it would be unlikely for them to transfer between models (i.e., adversarial example against one model manage to fool another model with distinct architecture or training data) if overfitting took place [62]. A widely held conjecture attributes adversarial examples to the inflexibility of classification models [15], [20], [62]. This conjecture is supported by the success of attacks that approximate DNNs' classification boundaries by linear separators [20], [42].

B. Defending Neural Networks

Proposals to ameliorate DNN's susceptibility to adversarial examples follow two 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. Some techniques involve augmenting the training process with adversarial examples in the hope that the DNN will learn to classify them correctly [20], [29], [34], [59]. These techniques were found to increase the norm of the perturbation needed to achieve misclassification. However, it remains unclear whether the increase is sufficient to make the

adversarial examples noticeable to humans. A recent adversarial training method significantly enhanced the robustness of DNNs with small input size trained on small datasets [36]. Another recent defense achieved relatively high success via approximating the outputs achievable via perturbations with bounded max-norm (i.e., L_∞), and ensuring these outputs are classified correctly [31]. Unfortunately, both recent methods are limited to adversaries with limited max-norm perturbations, and they do not scale to large datasets with large input size. Training DNNs with soft labels (assigning a probability lower than one to the correct class) rather than with hard labels (assigning probability one to the correct class and zero to others) was also proposed as a defense [48], [62]. This defense is ineffective against Carlini and Wagner's attack [8].

The other line of work proposes techniques to detect adversarial examples (e.g., [16], [21], [38], [39], [63]). The main assumption of this line of work is that adversarial examples follow a different distribution than that of 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 [39]. 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 these detectors were originally evaluated on [7].

C. Physical Attacks on Machine-Learning Systems

Similarly to this work, our prior work proposed using eyeglasses to perform physically realizable dodging and impersonation against state-of-the-art DNNs for facial recognition [56]. Differently from our prior work, here we propose an algorithm that attempts to make the texture of the eyeglasses as close as possible to real designs found online, while still fooling the DNNs in a desired manner. Moreover, we run a user study to measure the inconspicuousness of the eyeglasses. We find that our algorithms can produce more robust and inconspicuous attacks (Sec. V-B and Sec. V-E). We also show our attacks to be scalable and robust in the face of defenses.

Another line of work attempts to achieve privacy from face-recognition systems by completely avoiding *face detection* [23], [65]. 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 could avoid the post processing of her face image by recognition systems. The proposed techniques are not inconspicuous: they either use excessive makeup [23] or attempt to blind the camera using light-emitting eyeglasses [65].

Kurakin et al. demonstrated that imperceptible adversarial examples manage to fool DNNs even when providing the DNNs with an image of the adversarial example printed on paper [33] . Differently than us, they created adversarial perturbations that covered the entire image they aimed to misclassify. Recently, Evtimov et al. showed that especially crafted patterns can mislead DNNs for street-sign recognition when printed and affixed to actual street signs [14]. Their proof-of-concepts raised attention to the possible limitations of algorithms used in self-driving vehicles.

The susceptibility to attacks of learning systems that operate on non-visual input was studied too. For instance, researchers showed that speech-recognition systems can be misled to interpret sounds that are incomprehensible to humans as actual commands [6], [66]. Other work found that electrocardiogram-based authentication can be bypassed via signal injection [11].

Some work did not focus on physical-domain attacks per se, but rather digital-domain attacks that satisfy certain constraints [9], [22], [57], [64]. For example, Xu et al. showed to automatically create malicious PDFs that cannot be detected using machine-learning algorithms [64]. The constraints tackled in the digital domain have different nature than the ones we tackle in the physical domain.

III. A Novel Attack Against DNNs

In this section, we describe a new algorithm to create inconspicuous attacks against DNNs. We begin by making our threat model explicit in Sec. III-A, and then provide needed background on Generative Adversarial Networks in Sec. III-B. Finally, we describe our attack framework in Sec. III-C.

A. Threat Model

We assume an adversary who gains access to an already trained face-recognition system to mount her attack. The adversary cannot poison the parameters of the face-recognition system by injecting mislabeled data or altering training data. Instead, she can only alter the inputs to be classified.

The attacker's goal is to present herself to the face-recognition system in such a way that she is misclassified as someone other than herself. We consider two variants of this attack. In *impersonation* (or *targeted*) attacks, the adversary attempts to be misclassified as a specific other person. In *dodging* (or *untargeted*) attacks, the adversary attempts to be misclassified as an arbitrary other person. Obviously, successful impersonation counts as successful dodging, as well.

We typically assume the adversary is operating under a *white-box* scenario. That is, the adversary knows the feature space (images in RGB representation, as is typical in DNNs for image classification) and the internals (architecture and parameters) of the system being attacked [47]. Studying robustness of DNNs under white-box assumptions is the standard in the literature (e.g., [8], [42]). Moreover, as shown in Sec. V-D and in prior work (e.g., [46]), black-box attacks can use white-box attacks on local substitute-models which they later transfer to the black-box. We note that gradient approximation techniques can also be used to generalize our proposed method to black-box settings (e.g., [17], [43]).

B. Generative Adversarial Networks

Our attacks build on Generative Adversarial Networks (GANs) [19] 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. A trained G maps samples from a distribution, Z, that we know how to sample from (e.g., $[-1,1]^d$) 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, the training procedure can be conceptualized as a game with two players, D and G, in which D is trained to emit 1 on real examples from the dataset and 0 on the generated samples, and G is trained to generate outputs that are (mis)classified as real by D. In practice, GAN's training proceeds iteratively, and alternates between updating the parameters of G and D via back-propagation. G is trained to minimize the following function:

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

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

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

$Gain_D$ is maximized when D emits 1 on real samples, and 0 otherwise.

Several GAN architectures and training procedures have been proposed to train GANs, including Deep Convolutional GANs [52], on which we build, and Wasserstein GANs [2].

C. Attack Framework

Except for a few attacks [5], [14], [51], [56], [67], 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 Sec. II-A). Differently from previous attacks, we propose to train neural networks to generate outputs that can be used to achieve desired evasions. That is, instead of iteratively tweaking benign inputs to become adversarial, we iteratively update the weights of neural networks so that the neural networks, when so adjusted, will generate outputs that lead to misclassification.

More specifically, we propose to train neural networks to generate images of eyeglasses that, when worn, cause dodging or impersonation. To achieve inconspicuousness, we require that the eyeglasses generated by these neural networks resemble real eyeglass designs. We call the neural networks we propose *Adversarial Generative Nets (AGNs)*. Similarly to GANs (see Sec. III-B), 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—in our case, neural networks designed to recognize faces.

Formally, three neural networks are involved in AGN training: 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]

$$Loss_G(Z,D) - \kappa \cdot \sum_{z \in Z} Loss_F(x + G(z)) \tag{1}$$

We define $Loss_G$ in the same manner as in Sec. III-B—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 that is maximized when training G (as $-Loss_F$ is minimized). The definition of $Loss_F$ depends on whether the attacker aims to achieve dodging or impersonation. For dodging, we use:

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

For impersonation we use:

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

By maximizing the $Loss_F$, for dodging, the probability of the correct class c_x decreases; while for impersonation, the probability of the target class c_t increases. We chose the aforementioned definition of $Loss_F$ because we empirically found that it causes the training of AGNs to converge faster than $Loss_{cw}$ or loss functions defined via cross entropy used in prior work [8], [56]. κ is a parameter that balances the two objectives of G; we discuss it further below.

As part of the training process, D is trained to maximize $Gain_D$, defined previously. By doing so, D adjusts its weights to G, helping G to generate more realistic examples. In contrast to D and G, $F(\cdot)$'s weights are unaltered during training (as the attacks should be generated to fool the same DNN during test time).

The algorithm for training AGNs is provided in Alg. 1. The algorithm takes as input a set of benign examples (X), a pre-initialized generator and discriminator, a neural network to be fooled, a dataset of real examples (which the generator's output should resemble; in our case this is a dataset of eyeglasses), a function that enables sampling from G's latent space (Z), the maximum number of training epochs (N_e), the size of mini-batches[2] s_b, and κ (a value between zero and one). The result of the training process is an *adversarial generator* that creates outputs (e.g., eyeglasses) that fool $F(\cdot)$. In each iteration of training, either D or G are updated using a subset of the data that is randomly chosen. D's weights are updated via gradient ascent to increase $Gain_D$. G's weights, in contrast, are updated via gradient descent to minimize the value of Eqn. 1. To balance the generator's two objectives, the derivatives from $Gain_D$ and $Loss_F$ are carefully joined. We do so by normalizing the two derivatives to have the the lower Euclidean norm of the two (line 20 in the algorithm),

[1] We slightly abuse the mathematical 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.

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

Algorithm 1: AGN training

Input : X, G, D, $F(\cdot)$, $dataset$, Z, N_e, s_b, $\kappa \in \{0, 1\}$
Output: Adversarial G

1 $e \leftarrow 0$;
2 **for** $e \leftarrow 1$ **to** N_e **do**
3 create $mini\text{-}batches$ of size s_b from $dataset$;
4 **for** $batch \in mini\text{-}batches$ **do**
5 $z \leftarrow s_b$ samples from Z;
6 $gen \leftarrow G(z)$;
7 $batch \leftarrow concat(gen, batch)$;
8 **if** *even iteration* **then** // update D
9 update D by backpropagating $\frac{\partial Gain_D}{\partial batch}$;
10 **end**
11 **else** // update G
12 **if** $F(\cdot)$ *fooled* **then**
13 return;
14 **end**
15 $d_1 \leftarrow -\frac{\partial Gain_D}{\partial gen}$;
16 $x \leftarrow s_b$ sample images from X;
17 $x \leftarrow x + gen$;
18 Compute forward pass $F(x)$;
19 $d_2 \leftarrow \frac{\partial Loss_F}{\partial gen}$;
20 $d_1, d_2 \leftarrow normalize(d_1, d_2)$;
21 $d \leftarrow \kappa \cdot d_1 + (1 - \kappa) \cdot d_2$;
22 update G via backpropagating d;
23 **end**
24 **end**
25 **end**

and then add them together while controlling, via setting κ, which of the two objectives gets more weight (line 21 in the algorithm). When κ is closer to zero, more weight is given to fooling $F(\cdot)$, and less weight is given to making the output of G realistic. Conversely, setting κ closer to one puts more weight on making G's output resemble real examples. Training ends when the maximum number of training epochs is reached, or when $F(\cdot)$ is fooled in the desired manner—in other words, when impersonation or dodging are achieved.

IV. AGNs that Fool Face Recognition

In this section, we describe how we trained AGNs to generate inconspicuous, adversarial eyeglasses that can be used to mislead state-of-the-art DNNs trained to recognize faces. To train the AGNs, we needed to: *1)* collect a dataset of real eyeglasses to be used in training; *2)* select the architecture of the generator and the discriminator, and instantiate their weights; *3)* train DNNs that could be used to evaluate the attacks; and *4)* set the parameters for the attack. In what follows, we discuss how we carried out each of these tasks.

A. 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 large variety of designs, we searched for "eyeglasses" and synonyms thereof (e.g., "glasses," "eyewear"), sometimes modified by an adjective. The adjectives we used mainly included 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.

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

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).

The images we collected were not of only eyeglasses; e.g., we found images of cups, vases, and logos of eyeglass brands. Such images hinder the training process. Moreover, the images also contained eyeglasses worn by models and images of eyeglasses over dark backgrounds. We found these images hard to model using generative nets. Therefore, we trained a classifier to help us detect and keep only images of eyeglasses over white backgrounds and that are not worn by models. Using 250 hand-labeled images, we trained a classifier and tuned it to have 100% precision and 65% recall. After applying it on all the images in the dataset, 8,340 images of eyeglasses remained. Manually examining a subset of these images revealed no false positives.

Using the raw images from this dataset, it was possible to train a generator that can emit eyeglasses of different patterns, shapes, and orientations. Unfortunately, variations in shape and orientation made such eyeglasses difficult to efficiently and reasonably align to face images while running Alg. 1. Therefore, we preprocessed the images in the dataset and transferred the patterns from their frames to a fixed shape that can be easily aligned to face images. A silhouette of the shape we used is shown in Fig. 1. We then trained the generators 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 [12]. Fig. 2 shows examples of the texture synthesis results. Since the texture synthesis process is nondeterministic, we repeated it twice per image, getting a slightly different result each time. 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 was trained to emit images

Fig. 3: Examples of eyeglasses emitted by the generator (left) and similar eyeglasses from the training set (right).

of eyeglasses that are *printable*. In particular, we needed to ensure that the colors of the eyeglasses were within the range of colors our commodity printer (see Sec. V-B) 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. We then captured a picture of that image and computed the convex hull of all the RGB triplets found in the captured image. The convex hull was used as an approximation to the printer's color gamut. 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.

B. Pretraining the Generator and the Discriminator

When training GANs, it is desirable for the generator to emit sharp, real-looking, diverse images. Emitting only a small set of images would indicate the generator's function does not approximate the underlying distribution of data 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. In particular, the Deep Convolutional GAN architecture is known for its ability to train generators that can emit sharp, real-looking images [52].

We then explored a variety of different possibilities for the generator's latent space, the output dimensionality, and the number of weights in 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 containing 64×176 pixels produced the best-looking, diverse results. The final architecture of G and D is reported in Table I.

To ensure that our 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 Alg. 1. [4] Moreover, we used Salimans et al.'s recommendation and trained D on *soft labels* [55]. Specifically, we trained D to emit 0 on samples originating from the generator, and 0.9 (instead of 1) on real examples. Fig. 3 presents a couple of eyeglasses that the generator emitted by the end of training.

C. DNNs Used 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 [49]. 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 [25]. 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]. The main design consideration of OpenFace is to provide high accuracy DNN with low training and prediction times such that the DNN can be deployed on mobile and IoT devices. Hence, the DNN is relatively compact—it contains 3.74 million parameters. Despite the low number of parameters, the original DNN achieves near-human accuracy on the LFW benchmark (92.92%).

We trained one small and one large face-recognition DNN for each of the two architectures. Since we wanted to experiment with physically realizable dodging and impersonation, we trained the DNNs to recognize a mix of subjects available locally to us and celebrities for whom it was possible to 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 [32]: 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, respectively. 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 an input and produces a highly discriminative face descriptor (i.e., vector representation of the face) of 4096 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.

The architectures of the neural networks trained for each of VGG10 and VGG143 are provided in Table I. Essentially, they consist of fully-connected layers (i.e., linear separators) connected to a so-called *softmax* layer that turns the linear separators' outputs into probabilities. We trained the networks using the standard technique of minimizing cross-entropy loss [18]. 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.

[4] For training, we used the Adam optimizer [30] 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.

Model	*Architecture*
G (generator)	$FC(25 \times 7040) \rightarrow RB \rightarrow Reshape(200, 4, 11) \rightarrow Deconv \rightarrow RB \rightarrow Deconv \rightarrow RB \rightarrow Deconv \rightarrow RB \rightarrow Deconv \rightarrow tanh$
D (discriminator)	$Conv \rightarrow LReLu \rightarrow Conv \rightarrow LrB \rightarrow Conv \rightarrow LrB \rightarrow Conv \rightarrow LrB \rightarrow Flatten \rightarrow FC(7040 \times 1) \rightarrow Sigmoid$
VGG10	$FC(4096 \times 10) \rightarrow softmax$
VGG143	$FC(4096 \times 143) \rightarrow softmax$
OF10	$FC(128 \times 12) \rightarrow tanh \rightarrow FC(12 \times 10) \rightarrow softmax$
OF143	$FC(128 \times 286) \rightarrow tanh \rightarrow FC(286 \times 10) \rightarrow softmax$
Detector	$Conv \rightarrow RB \rightarrow MP \rightarrow Conv \rightarrow RB \rightarrow Conv \rightarrow RB \rightarrow FC(196 \times 2) \rightarrow softmax$

TABLE I: Architectures of the neural-networks used in this work. For the OpenFace and VGG DNNs, we only report the layers we added for the base, feature extraction, DNNs. $Conv$ refers to convolution, $Deconv$ to deconvolutional (a.k.a, transposed convolution), FC to fully-connected layer, $Flatten$ to vectorization of matrices, LrB to batchnorm followed by a leaky rectified-linear layer, $LReLu$ to leaky rectified-linear layer, MP to max-pooling layer, RB to batchnorm followed by a rectified-linear layer, and $tanh$ to hyperbolic tanget. All convolutions and de-convolutions in G and D use 5×5 filters, and have strides and paddings of two. The detector's convolutions use 3×3 filters, have strides of two, and padding of one. The detector's MP layer has a 2×2 window size and a stride of two.

Model	*acc.*	*SR naive dodge*	*SR naive 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%

TABLE II: Performance of the face-recognition DNNs. We report the accuracy (how often the correct class is assigned the highest probability), the success rate (SR) of naïve dodging (how likely are naïve attackers to be arbitrarily misclassified) and impersonation (how likely are naïve attackers to be misclassified as random targets), the threshold to balance correct and false classifications, the true-positive rate (TPR; how often the correct class is assigned a probability higher than the threshold), and the false-positive rate (FPR; how often a wrong class is assigned a probability higher than the threshold). The relatively low TPR for OF143 is unsurprising, since the network's design prioritized compactness.

An initial evaluation of VGG10 and VGG143 showed high performance. To further increase our confidence 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% and <0.01% chance for VGG10 and VGG143, respectively. However, we found that dodging might succeed with non-negligible chance—naïve attackers are likely to succeed in 7.81% and 26.87% of their dodging attempts against VGG10 and VGG143, respectively (possibly because the training samples do not show certain subjects wearing eyeglasses). To make the DNNs more robust, we augmented their training data, following adversarial training techniques [34]. For each image initially used in training we added two variants with generated eyeglasses attached. We found no improvement by augmenting a larger number of images. We also followed Kurakin et al.'s advice and included 50% raw training images and 50% augmented images in each mini-batch during training [34].

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 lower than 0.01%. Finally, to maintain a high level of security, it is important to minimize the DNN's false positives [27]. 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 on the top probability emitted by VGG10 and VGG143 such that their accuracies remained as high as 100% and 98%, while the false-positive rates of both DNNs remained 0%. The performance of the DNNs and the thresholds we set are reported in Table II.

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, whereas the descriptors of different people's images are far. In contrast to VGG, the descriptors of OpenFace lie on a unit sphere.

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

To improve the robustness of the neural networks, we increased their depth by prepending a fully-connected layer followed by a hyperbolic-tangent (*tanh*) layer (see Table I). This architecture was chosen as it performed the best out of different ones we experimented with. We also increased the number of images we augmented in training to 10 and 100 (per image in the training set) for OF10 and OF143, respectively. The number of images augmented was selected such that increasing it did not result in improved 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.

The performance of the networks is reported in Table II. 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 DNNs we trained for face recognition, we trained a DNN to detect attacks that target the VGG networks following the proposal of Metzen et al. [39]. We chose this detector because it was found to be one of the most effective detectors against imperceptible adversarial examples [39], [7]. Moreover, we trained a detector for the VGG DNNs only as no architecture was proposed for the detector to detect attacks against OpenFace-like architectures. To mount a successful attack when a detector is deployed it becomes necessary to simultaneously fool the detector and the (face-recognition) DNN.

We used the architecture proposed by Metzen et al. (see Table I). To achieve the 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 [49]. 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 [30].Training parameters were set to standard values (learning rate $= 1\text{e}{-4}$, $\beta_1 = 0.99$, $\beta_2 = 0.999$). At the end of training, we evaluated the detector using a set of 20 subjects who were not used in training, finding that it had 100% recall and 100% precision (i.e., it did not err).

D. Implementation Details

Setting parameters The Adam optimizer was used to update the weights of D and G when running Alg. 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 $5\text{e}{-5}$ gave the best tradeoff between success in fooling the DNNs, inconspicuousness, and the algorithm's run time. The number of epochs was limited to at most one, as we found that the results the algorithm returned when running longer were not inconspicuous.

Libraries used The majority of our work was implemented in MatConvNet, a MATLAB toolbox for convolutional neural networks [61]. The OpenFace DNN was translated from the original implementation in Torch to MatConvNet.

System details We ran the experiments on a Linux machine equipped with two Nvidia Titan X GPUs (12GB of memory each), Intel i5-6400 CPU, and 24GB of memory.

V. Evaluating AGNs

We extensively evaluated the attacks proposed in this work. In Sec. V-A 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 Sec. V-B that these attacks can also be successfully mounted in the physical domain. In Sec. V-C, we demonstrate universal dodging, i.e., using a small set of subjects and images to generate a few eyeglasses that can be used by other subjects to evade recognition. We show in Sec. V-D that our attacks transfer well between models. In Sec. V-E we demonstrate that AGNs can generate eyeglasses that are inconspicuous to human participants in a user study. Finally, in Sec. V-F we show how AGNs can fool a digit-recognition DNN.

A. 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 for classification, since the inputs are not subject to the noise added by physically realizing the attack and capturing the image with a camera. Therefore, it is important to first see whether the attacker can successfully fool the DNNs in the digital domain, as failure in the digital domain implies certain 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 ten subjects for VGG10 and OF10. For each combination of subject and DNN we used a single image of the subject to create dodging or impersonation attacks. In impersonation attacks, the targets were chosen at random. To compute the uncertainty in our estimation of success, we repeated each attack using three different images of the attacker.

To test whether a detector can be used to prevent attacks, we selected additional sets of 10 and 20 subjects (with three images per subject) for VGG10 and VGG143, respectively. 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 Eqn. 1 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.

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

Model	# attackers	Without detector Dodging suc. rate	Without detector Impers. suc. rate	With detector Dodging suc. rate	With detector Impers. suc. rate
VGG10	10	100±0%	100±0%	100±0%	100±0%
VGG143	20	100±0%	88±5%	100±0%	90±4%
OF10	10	100±0%	100±0%	-	-
OF143	20	100±0%	90±4%	-	-

TABLE III: Results of attacks in the digital environment. In each attack we used three images of the subject to be misclassified. We report the the mean success rate of attacks and the standard error, when fooling the facial-recognition DNNs with and without the detector trained to detect attacks against the VGG networks (see Sec. IV-C). To the left, we show the DNNs attacked, and the number of attackers. For impersonation, the targets were picked at random.

Fig. 4: An example of digital dodging. Left: An image of actor Owen Wilson, correctly classified by VGG143 with probability 1.00. Right: Dodging against VGG143 using AGN's output (probability assigned to the correct class: < 0.01).

same criteria to measure success when the detector was used, but we also required that the detector assigned a probability higher than 0.5 that the input was non-adversarial.

Experiment results Table III summarizes the results of the digital-environment experiments. All dodging attempts succeeded; Fig. 4 shows an example. As with dodging, all impersonation attempts 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, but the success rates were still high.

Using a detector did not prevent the attacks: success rates for dodging and impersonation were similar to when a detector was not used. However, as detailed in Sec. V-E, using a detector reduces the inconspicuousness of attacks.

We tested whether attackers can accomplish higher success by using eyeglasses of different shapes. To this end, we trained additional AGNs to generate eyeglasses of six new shapes and tested whether attackers would impersonate more successfully against VGG143 and OF143. We found that three of the new shapes achieve comparable performance to the one in Fig. 1, but they do not improve the overall success when combined. While the new shapes did not enhance the likelihood of successful evasion, they could still be useful for enhancing the inconspicuousness of attacks.

Fig. 5: A 3d-model of the eyeglasses we 3d-printed.

B. 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, and distance from the camera (among others) may dramatically change the concrete values of pixels. Thus, attacks need to be robust against changes in the imaging environment in order to succeed. We took two additional measures to make the attacks more robust.

First, to train adversarial generators that emit images of eyeglasses that lead 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 Alg. 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 against changes in pose, we trained the adversarial generator to minimize $Loss_F$ when the eyeglasses were well aligned to the face's orientation. To determine how to align the eyeglasses to the attacker's face, we created and printed a 3d model of the eyeglasses whose (front) frames have the same shape as the eyeglasses emitted by the generator.[5] A rendering of the 3d-model of the eyeglasses we printed is shown in Fig. 5. We added tracking markers—specifically positioned green dots—to the 3d-printed eyeglass frames. The attacker wore the eyeglasses when capturing training data for the generator. We then used the markers to find a projective alignment, θ_x, to align the eyeglasses emitted by the generator to the attacker's pose in each one of the collected images. Subsequently, the generator was trained to minimize $Loss_F\big(x + \theta_x(G(z))\big)$ for different images of the attacker ($x \in X$).

Experiment setup To evaluate the physically realized attacks, three subjects from our research team acted as attackers: S_A (the 1^{st} author), a Middle-Eastern male in the mid-20s, S_B (the 3^{rd} author), a white male in the early 40s, and S_C (the 2^{nd} author), a South-Asian female in the mid-20s. Each subject attempted both dodging and impersonation attacks against each of the four DNNs (which were trained to recognize them, among others).

To train the adversarial generators and create the attacks, we collected 45 images of each attacker (the set X in Alg. 1) while he or she stood a fixed distance from the camera, kept a neutral expression, and moved his head up-down, left-right, and in a circle. Each generator was trained for at most one epoch, and was stopped earlier if the generator reached a point

[5]We used the code in https://github.com/caretdashcaret/pince-nez to turn the silhouette into a 3d model, and created the temples in AutoCAD.

where it could emit eyeglasses that, in the case of dodging, led the mean probability of the correct class to fall below 0.005, or, in the case of impersonation, led the mean probability of the target class to exceed 0.99. As explained earlier, when collecting the generator's training data, the attackers wore 3d-printed eyeglasses that were later used to align the generator's output to the attackers' pose.

To physically realize the attacks, we printed selected eyeglasses emitted 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 several eyeglasses that can lead to successful impersonation or dodging, we (digitally) sampled 48 outputs and kept the top 25% (i.e., a set of 12 images of eyeglasses) that were most successful for dodging or impersonation (the mean probability of the correct class was the lowest, or the mean probability of the target class was the highest, respectively). We then manually selected eyeglasses to print out and test, until dodging or impersonation were successful. We chose to sample 48 outputs and to keep the top 25% as we found these numbers to result in a collection that was diverse, and yet small enough to manually inspect and test.

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 each 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, whereas for impersonation we measured success by the fraction of frames that were classified as the target. We picked the target of impersonation at random per attack. 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] to RGB=[255,255,255]). In such cases, we attempted to impersonate another (randomly picked) target.

In our collection, we ensured that the attackers covered a range of poses. We used a state-of-the-art tool [4] to measure the head poses of the attackers in the images (i.e., pitch, yaw, and roll angles). On average, the attackers covered 13.01° in the pitch (up-down) direction, 17.11° in the yaw (left-right) direction, and 4.42° in the roll (diagonal) direction. This is comparable to the mean difference in head pose between pairs of images randomly picked from the PubFig dataset (11.64° in pitch, 15.01° in yaw, and 6.51° in roll).

As a baseline to compare to, we repeated the dodging and impersonation attempts using our prior algorithm [56].

The data used for training and evaluating the physically realized attacks were collected from the same room at our university using a Canon T4i camera. This room has a ceiling light but no windows on exterior walls. This limited the effects of lighting, which are difficult to model realistically [10].

Experiment results Most dodging attempts succeeded with 100% of video frames misclassified. Even in the worst case 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 Sec. IV-C.

For impersonation, for each DNN, one to four subjects had to be targeted before impersonation succeeded. For successful impersonations, an average of 69.70% of the video frames were classified as the target. In half of these attempts, >40% of frames were misclassified with high confidence (again using the thresholds discussed in Sec. IV-C). This suggests that even a conservatively tuned system would be likely be fooled by some attacks.

The results are summarized in Table IV and examples of eyeglasses are shown in Fig. 6.

Openface vs. VGG Interestingly, despite the OpenFace DNNs having many fewer parameters than the VGG DNNs, fooling them appears to be more difficult. For example, dodging against OpenFace was less successful on average than against VGG (94.54% vs. 98.45% of video frames misclassified on average, respectively), and impersonation against OpenFace required more attempts at picking targets (2 picks on average) until success than VGG (1.67 picks). Moreover, as we discuss later, attacks against the OpenFace network were less inconspicuous than attacks against the VGG networks. Thus, building a robust DNN (i.e., one that is difficult to mislead) using an OpenFace-like architecture may be a more promising future direction than doing so using a VGG-like architecture.

Comparison to previous work We found that physical-domain evasion attempts using AGNs were significantly more successful than attempts using our previous algorithm [56]. The mean success rate of dodging attempts was 45% higher when using AGNs compared to prior work (96.50% compared to 66.57%; a paired t-test shows that the difference is statistically significant with $p = 0.03$). The difference in the success rates of impersonation attempts was even larger. The mean success rate of impersonation attempts was 127% higher when using AGNs compared to prior work (69.70% compared to 30.66%; 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 our previous algorithm [56].

Effect of pose and other factors Last, we built a mixed-effects logistic regression model [50] to analyze how different factors, and especially the head pose, affect the success of physical-domain attacks. In the model, the dependent variable was whether an image was misclassified, and the independent

Fig. 6: Eyeglasses used in physically realized attacks by S_B to dodge against OF143 (left) and by S_A to impersonate S_D against VGG10 (right) (see Table IV).

DNN	Subject	Dodging results SR	Dodging results $E(p(\text{subject}))$	Dodging results SR [56]	Impersonation results Target	Impersonation results Attempts	Impersonation results SR	Impersonation results HC	Impersonation results $E(p(\text{target}))$	Impersonation results SR [56]
VGG10	S_A	100%	0.06	0%	S_D	1	100%	74%	0.93	0%
	S_B	97%	0.09	98%	Milla Jovovich	2	91%	6%	0.70	63%
	S_C	96%	0.10	100%	Brad Pitt	1	100%	94%	0.98	100%
VGG143	S_A	98%	0.17	0%	Alicia Keys	2	89%	41%	0.73	0%
	S_B	100%	<0.01	87%	Ashton Kutcher	2	28%	0%	0.22	0%
	S_C	100%	0.03	82%	Daniel Radcliffe	2	3%	0%	0.04	0%
OF10	S_A	81%	0.40	0%	Brad Pitt	2	28%	23%	0.25	0%
	S_B	100%	0.01	100%	Brad Pitt	2	65%	55%	0.58	43%
	S_C	100%	0.01	100%	S_D	1	98%	95%	0.83	67%
OF143	S_A	100%	0.09	36%	Carson Daly	1	60%	0%	0.41	0%
	S_B	86%	0.12	97%	Aaron Eckhart	4	99%	41%	0.83	92%
	S_C	100%	<0.01	99%	Eva Mendes	2	77%	39%	0.67	3%

TABLE IV: Summary of physical realizability experiments. The first two columns show the attacked DNNs and the attackers. For dodging, we report the success rate (SR; percentage of misclassified video frames), the probability of being classified as the correct class by the DNN (smaller is better), and the success rate of Sharif et al.'s attack [56]. For impersonation, we report the target, the number of targets we attempted until succeeding, the success rate of the attack (SR; 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 beteween the true and the false positive rate), the mean probability assigned to the target after classifying the adversarial images (higher is better), and success rate of Sharif et al.'s attack [56]. S_D is an Asian female in the early 20s. Non-adversarial images of the attackers were assigned to the correct class with a mean probability above 0.93.

variables accounted for the pitch (up-down), yaw (left-right), and roll (tilt) angles of the head in the image (measured via the University of Cambridge tool [4]), 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 [56]), the architecture of the DNN attacked (VGG or OpenFace), and the size of the DNN (10 or 143 subjects). To train the model, we used all the images we collected to test our attacks in the physical domain.

The coefficients learned from the model shed light on how angles affect the success of attacks. We found that movements in the pitch and yaw directions significantly affected the success, while movements in the roll direction had no significant effect: Each degree of deviation in the pitch direction away from $0°$, increases the likelihood of success by 1.05 times ($p < 0.01$), on average. Thus, an attacker who faces the camera with a pitch degree of $\pm 12°$ is about two times more likely to succeed than when directly facing the camera. In contrast, each degree of deviation in the yaw direction away from $0°$ decrease the attacker's likelihood of success by 0.97 times ($p = 0.05$), on average. As a result, an attacker whose yaw angle is $\pm 10°$ is about 0.80 times as likely to succeed as when directly facing the camera. These results highlight the attack's robustness, as its success was not drastically harmed by increasing by head movements, and even improved when moving along the pitch direction. The results also suggest that to better resist attacks, operators should position the camera at face level and capture images with a slight pitch angle.

C. Universal Dodging Attacks

We next show that it is possible to create small number of adversarial eyeglasses that will lead to successful dodging for the majority of subjects, even if images of those subjects were not used in training the adversarial generator.

Algorithm 2: Universal attacks (given many subjects)

Input : X, G, D, $F(\cdot)$, *dataset*, Z, N_e, s_b, κ, s_c
Output: *Gens* `// a set of generators`

```
Gens ← {};
clusters ← clusters of size s_c via k-means++;
for cluster ∈ clusters do
    G ← Alg1(cluster, G, D, F, dataset, Z, N_e, s_b, κ);
    Gens ← Gens ∪ {G};
end
```

We created the universal attacks by training the generator in Alg. 1 on a set of images of different people. Consequently, the generator learned to emit eyeglasses that led a set of 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 Alg. 2, which uses Alg. 1 to train several adversarial generators, one per cluster of similar subjects. Alg. 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 (4096-dimensional features for VGG, and 128-dimensional features for OpenFace; see Sec. IV-C). The result was a set of generators, which, combined, generate eyeglasses that led to the misclassification of a large fraction of subjects, yet the eyeglasses seemed 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 eyeglasses for similar subjects to dodge than for vastly different subjects.

Experiment setup We tested the universal attacks against VGG143 and OF143 only, as the other DNNs are not suitable for evaluating universality due to being trained on too few

subjects. To train and evaluate the generators, we selected two images for each one 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 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 were the highest among all the images available to us.

We evaluated the success of universal dodging attacks by computing the fraction of subjects from the test set whose images were misclassified using eyeglasses emitted by the generators. 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 generators. To increase confidence in the result, we averaged the success rate after repeating the process five times (each time selecting a random set of subjects for training). When using $\geq$ 50 subjects for the universal attacks, we used Alg. 2 and set the cluster size, s_c, to 10.

Additionally, we explored how the number of picked eyeglasses affected the success of the attack. We did so by generating a large number of eyeglasses—100 eyeglasses from each trained generator, or set of generators, seemed to capture the majority of different patterns they could emit—and finding the subsets that led the largest fraction of images in the test set to be misclassified. This is essentially a *maximum coverage problem*, which is NP-hard [24]. Therefore, we used a procedure based on submodular function maximization to select the set of eyeglasses [44]. The algorithm starts from selecting the eyeglasses that maximize the success rate (i.e., they led to the misclassification of the largest fraction of images from the test set). It then proceeds iteratively, increasing the set of eyeglasses by picking the pair that leads to the misclassification of the largest fraction of test images that have not yet been misclassified. This algorithm provides a $(1 - \frac{1}{e})$-approximation of the optimal success rate [44].

Experiment results Fig. 7 summarizes the results of the experiments for VGG143 and OF143. 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 different 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.

D. Transferability of Dodging Attacks

In this section, we explore whether dodging against DNNs of one architecture leads to successful dodging against DNNs of a different architecture. Indeed, the transferability of attacks has been shown to be effective in fooling models to which adversaries do not have access (e.g., [46]). 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.

Using the data from Sec. V-A, we first tested whether dodging in the digital environment successfully transferred between architectures (see Table V). We found that attacks against the OpenFace architecture successfully fooled the VGG architecture in only a limited number of attempts (10–12%). In contrast, dodging against VGG led to successful dodging against OpenFace in at least 63.33% of attempts.

Universal attacks seemed to transfer between architectures with similar success. Using attacks created with 100 subjects and 10 eyeglasses from Sec. V-C, we found that about 81.54% ($\pm$2.82% standard deviation) of attacks transferred from VGG143 to OF143, and 26.15% ($\pm$3.94% standard deviation) transferred in the other direction.

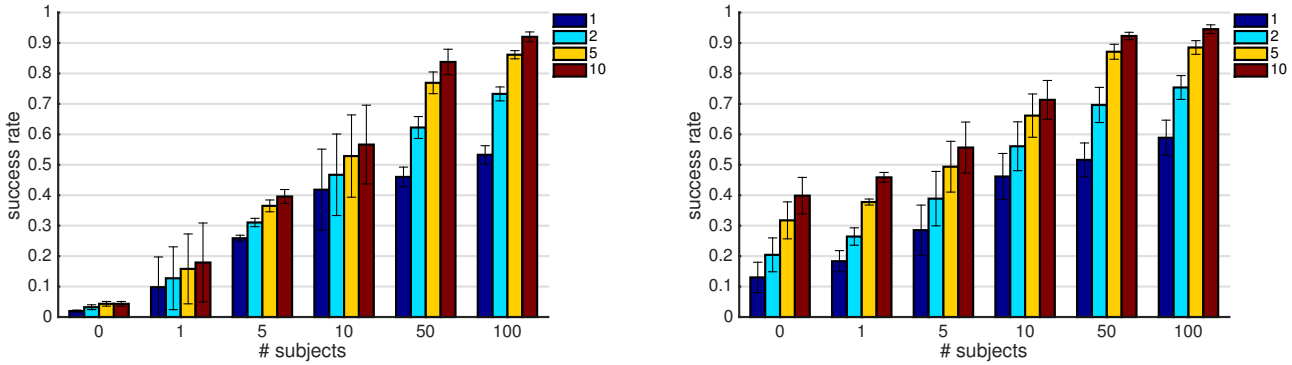
As opposed to dodging in the digital environment, physically realized dodging attacks do not transfer well (see Table VI). Using the frames that successfully dodged from Sec. V-B, we found that none of the video frames that successfully dodged against the OpenFace DNNs managed to dodge against the VGG DNNs, and only a small percentage (7–18%) of the video frames that dodged against VGG, successfully dodged against OpenFace.

E. A User Study to Measure Inconspicuousness

Methodology To evaluate inconspicuousness of eyeglasses generated by AGNs for the different attacks 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 ten "real" eyeglasses and ten attack eyeglasses in random order. All eyeglasses were the same shape, and varied only in their coloring. The "real" eyeglasses were from images used for pretraining the AGNs (see Sec. IV-A). The attack eyeglasses were either generated by our AGNs or examples of eyeglasses from our previous work [56].

The eyeglasses that we showed to study participants were not photo-realistically or three-dimensionally rendered. This applies to all the eyeglasses, both the "real" ones and the ones produced as attacks. Consequently, we consider attack glasses to be found inconspicuous by participants not if they were uniformly rated as real (which even "real" glasses are not, particularly when attack glasses are inconspicuous), but, rather, if the rate at which participants deemed eyeglasses as real does not differ significantly regardless of whether they were "real" eyeglasses or attack eyeglasses.

We recruited 353 participants in the U.S. through the Prolific crowdsourcing service (https://prolific.ac). Their ages ranged



(a) (b)

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 that were misclassified (i.e., achieving successful dodging). 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 one, two, five, or ten eyeglasses. For example, the rightmost bar in Fig. 7b 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, Alg. 1 was used to create the attacks. For 50 and 100 subjects, Alg. 2 was used.

From \ To	VGG10	OF10
VGG10	-	63.33%
OF10	10.00%	-

From \ To	VGG143	OF143
VGG143	-	88.33%
OF143	11.67%	-

TABLE V: Transferability of dodging in the digital domain. Each table shows how likely it is for a generator used for dodging against one network (rows) to succeed against another network (columns).

From \ To	VGG10	OF10
VGG10	-	17.97%
OF10	0.00%	-

From \ To	VGG143	OF143
VGG143	-	7.23%
OF143	0.00%	-

TABLE VI: Transferability of dodging in the physical domain. 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).

from 18 to 77, with a median of 29. 44% of participants specified being female and 55% male (the remainder chose other or did not answer). The majority (60%) held an associate's degree or higher. Our study took $\sim$ 13 minutes to complete and participants were compensated $2.50. The study design was approved by our institution's ethics review board.

We measured the inconspicuousness of a set of eyeglasses by the fraction of time participants marked those eyeglasses as real. 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 [37], 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 one group being marked as real divided by the odds of eyeglasses in the other group being marked as real.

Results Overall, eyeglasses created by AGNs are more realistic than those created by previous attacks [56] ($\times$1.33 odds ratio, i.e., selected by participants as realistic 1.33$\times$ as

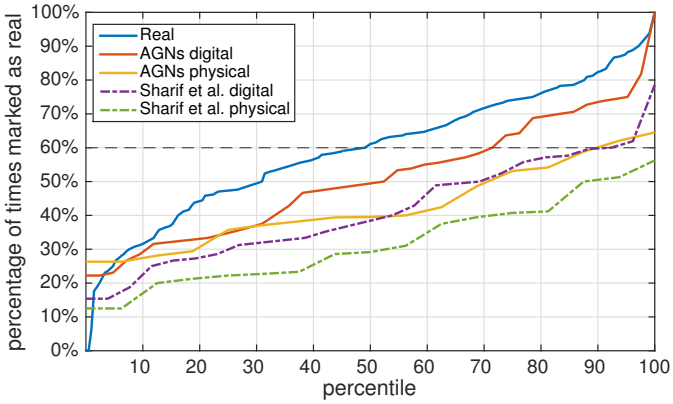


Fig. 8: The percentage of times in which eyeglasses from different sets were marked as real. The horizontal 60% line is highlighted to mark that the top half of "real" eyeglasses were marked as real at least 60% of the time.

Comparison (group 1 vs. group 2)		*Odds ratio*	*p-val*
Real (60%)	AGNs (47%)	1.68	<0.01
AGNs (47%)	Sharif et al. [56] (41%)	1.33	<0.01
AGNs:			
digital (51%)	physical (43%)	1.36	0.05
all digital (51%)	digital w. detector (42%)	1.46	<0.01
digital dodging (56%)	universal dodging (40%)	1.90	<0.01
physical w. VGG (49%)	physical w. OpenFace (37%)	1.62	0.03

TABLE VII: Relative realism of selected pairs of sets of eyeglasses. For each two sets compared, we report in parentheses the fraction of eyeglasses per group that were marked as real by the study participants, the odds ratios between the groups, and the p-value of the χ^2 test of independence. E.g., odds ratio of 1.68 means that the odds of eyeglasses being selected as real are 1.68 times higher if they belong to the first group than if they belong to the second.

often). "Real" glasses were more realistic than AGN-generated ones ($\times 1.68$ odds ratio).

Perhaps most indicative of inconspicuousness in practice is that many AGN-generated eyeglasses were as realistic as "real" eyeglasses. The most inconspicuous 30% of eyeglasses emitted by AGNs for digital-environment attacks were on average deemed equally 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 10% most inconspicuous were still marked as real at least 60% of the time (against to the top 50% of "real" eyeglasses).

Other results match intuition: the more difficult the attack, the bigger the impact on conspicuousness. E.g., digital attack glasses that do not try to fool a detector are less conspicuous then ones that fool a detector ($\times 1.46$ odds ratio), and individual dodging is less conspicuous than universal dodging ($\times 1.90$ odds ratio). Other comparisons and the percentage of time participants marked different glasses as real are shown in Table VII and Fig. 8.

F. Applying AGNs Against Digit Recognition

We next show that AGNs can apply to domains besides face recognition. Specifically, we use AGNs to fool a state-of-the-art DNN for detecting digits trained on the MNIST dataset [35], which contains 70,000 28x28-pixel images of digits.

Experiment setup First, we trained a DNN for digit recognition. We used the architecture and training code of Carlini and Wagner [8]. We trained the DNN on the first 55,000 digits in the dataset 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 28x28-pixel images of digits. We again used the Deep Convolutional GAN architecture [52]. Starting from the pretrained GANs, we trained AGNs using a variant of Alg. 1 to produce a generator that can emit images of digits that simultaneously fool the discriminator to be real and are misclassified by the digit-recognition DNN.

Fig. 9: An illustrations 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.

Experiment results After training, the generator was able to generate arbitrarily many adversarial examples of digits that appear comprehensible to human observers, but are misclassified by the digit-recognition DNN. As a test, we generated 5,004 adversarial examples to fool the DNN (Fig. 9 illustrates samples). The adversarial examples were produced by first generating 600,000 images using the adversarial generators (60,000 per generator). Out of all samples, 8.34% of samples that were misclassified by the DNN were kept. Out of these, only the digits that were likely to be comprehensible by humans were kept: the filtration process 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.

Unlike traditional attacks on digit recognition (e.g., [8]), these attack images are not explicitly designed for minimal deviation from specific benign inputs; rather, their advantage is that they can be substantially different from images in the training set. 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 set.

VI. Discussion and Conclusion

Face recognition is a compelling application of DNNs, but one for which the adversary's ability to dodge recognition (e.g., to evade detection at a checkpoint) or to impersonate another (e.g., to gain unauthorized access) can have significant consequences. In this paper we contributed a methodology based on *adversarial generative nets* (AGNs) to produce physical eyeglasses that, when worn, lead a DNN-based face-recognition system to misclassify the subject. Moreover, this methodology enables the construction of eyeglasses that are more inconspicuous (Sec. V-E) and robust (Sec. V-A, V-D) than previous approaches, and in a way that scales so that a small "universal" set of eyeglass variants enables attacks by many, including previously unseen, subjects (Sec. V-C).

Differently from most attacks that directly tweak benign samples to create adversarial examples (e.g., [59]), AGNs are optimized by training a neural network to fool targeted DNNs. Thus, they constitute a natural alternative for evaluating robustness of DNNs when other attacks fail. A related advantage of AGNs over other attack methods (e.g., [20], [59]) is that AGNs can generate a diverse set of adversarial examples, which can be useful for evaluating the robustness of models as well as for adversarial training [34].

AGNs are also general; as DNNs are applied to problems other than face recognition, we conjecture that AGNs could similarly be applied to generate physically realizable examples to mislead them. We demonstrate this by training AGNs to fool classifiers that recognize handwritten digits (Sec. V-F).

Defenses against our attacks is an obvious priority for future work. In the meantime, and since face-recognition systems are already seeing application in air-travel security [58], we have taken steps to notify the Transportation Security Administration of our findings and to suggest that they require that subjects remove physical artifacts (glasses, hats, and possibly jewelry) prior to face recognition being performed.

Acknowledgments

We would like to thank Cara Bloom for providing comments on an early draft of this work. This work was supported in part by a gift from NVIDIA, gifts from NATO and Lockheed Martin through Carnegie Mellon CyLab, and via a CyLab Presidential Fellowship.

References

[1] B. Amos, B. Ludwiczuk, and M. Satyanarayanan. OpenFace: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science, 2016.

[2] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. In *ICML*, 2017.

[3] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proc. SODA*, 2007.

[4] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Openface: an open source facial behavior analysis toolkit. In *Proc. WACV*, 2016.

[5] S. Baluja and I. Fischer. Adversarial transformation networks: Learning to generate adversarial examples. In *Proc. AAAI*, 2018. To appear.

[6] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou. Hidden voice commands. In *Proc. USENIX Security*, 2016.

[7] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proc. AISec*, 2017.

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *Proc. IEEE S&P*, 2017.

[9] Y. Chen, Y. Nadji, A. Kountouras, F. Monrose, R. Perdisci, M. Antonakakis, and N. Vasiloglou. Practical attacks against graph-based clustering. In *Proc. CCS*, 2017.

[10] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proc. SIGGRAPH*, 2008.

[11] S. Eberz, N. Paoletti, M. Roeschlin, M. Kwiatkowska, I. Martinovic, and A. Patané. Broken hearted: How to attack ecg biometrics. In *Proc. NDSS*, 2017.

[12] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *Proc. ICCV*, 1999.

[13] L. Engstrom, D. Tsipras, L. Schmidt, and A. Madry. A rotation and a translation suffice: Fooling CNNs with simple transformations. *arXiv preprint arXiv:1712.02779*, 2017.

[14] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, and D. Song. Robust physical-world attacks on machine learning models. *arXiv preprint arXiv:1707.08945*, 2017.

[15] A. Fawzi, S.-M. Moosavi-Dezfooli, and P. Frossard. Robustness of classifiers: from adversarial to random noise. In *Proc. NIPS*, 2016.

[16] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[17] M. Fredrikson, S. Jha, and T. Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proc. CCS*, 2015.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. NIPS*, 2014.

[20] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[21] K. Grosse, P. Manoharan, N. Papernot, M. Backes, and P. McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[22] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel. Adversarial perturbations against deep neural networks for malware classification. In *Proc. ESORICS*, 2017.

[23] A. Harvey. CV Dazzle: Camouflage from face detection. Master's thesis, New York University, 2010. Available at: http://cvdazzle.com.

[24] D. S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Co., Boston, MA, USA, 1997.

[25] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[26] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvári. Learning with a strong adversary. *CoRR, abs/1511.03034*, 2015.

[27] L. Introna and H. Nissenbaum. Facial recognition technology: A survey of policy and implementation issues. Technical report, Center for Catastrophe Preparedness and Response, New York University, 2009.

[28] C. Kanbak, S.-M. Moosavi-Dezfooli, and P. Frossard. Geometric robustness of deep networks: analysis and improvement. *arXiv preprint arXiv:1711.09115*, 2017.

[29] A. Kantchelian, J. Tygar, and A. D. Joseph. Evasion and hardening of tree ensemble classifiers. In *Proc. ICML*, 2016.

[30] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[31] J. Z. Kolter and E. Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017.

[32] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Proc. ICCV*, 2009.

[33] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[34] A. Kurakin, I. Goodfellow, and S. Bengio. Adversarial machine learning at scale. In *ICLR*, 2017.

[35] Y. LeCun, C. Cortes, and C. J. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[36] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[37] M. L. McHugh. The chi-square test of independence. *Biochemia Medica*, 23(2):143–149, 2013.

[38] D. Meng and H. Chen. Magnet: a two-pronged defense against adversarial examples. In *Proc. CCS*, 2017.

[39] J. H. Metzen, T. Genewein, V. Fischer, and B. Bischoff. On detecting adversarial perturbations. In *ICLR*, 2017.

[40] T. Miyato, S.-i. Maeda, M. Koyama, K. Nakae, and S. Ishii. Distributional smoothing with virtual adversarial training. In *ICLR*, 2016.

[41] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proc. CVPR*, 2017.

[42] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proc. CVPR*, 2016.

[43] N. Narodytska and S. Kasiviswanathan. Simple black-box adversarial attacks on deep neural networks. In *Proc. CVPRW*, 2017.

[44] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functionsI. *Mathematical Programming*, 14(1):265–294, 1978.

[45] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782, 1980.

[46] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami. Practical black-box attacks against machine learning. In *Proc. AsiaCCS*, 2017.

[47] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. In *Proc. IEEE Euro S&P*, 2016.

[48] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proc. IEEE S&P*, 2016.

[49] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proc. BMVC*, 2015.

[50] J. Pinheiro, D. Bates, S. DebRoy, D. Sarkar, and R Core Team. *nlme: Linear and Nonlinear Mixed Effects Models*, 2015. R package version 3.1-122.

[51] O. Poursaeed, I. Katsman, B. Gao, and S. Belongie. Generative adversarial perturbations. *arXiv preprint arXiv:1712.02328*, 2017.

[52] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016.

[53] A. Rozsa, E. M. Rudd, and T. E. Boult. Adversarial diversity and hard positive generation. In *Proc. CVPR DeepVision Workshop*, 2016.

[54] S. Sabour, Y. Cao, F. Faghri, and D. J. Fleet. Adversarial manipulation of deep representations. In *ICLR*, 2016.

[55] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Proc. NIPS*, 2016.

[56] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proc. CCS*, 2016.

[57] N. Srndic and P. Laskov. Practical evasion of a learning-based classifier: A case study. In *Proc. IEEE S&P*, 2014.

[58] Y. Steinbuch. JetBlue ditching boarding passes for facial recognition. *New York Post*, May 31 2017.

[59] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[60] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proc. CVPR*, 2014.

[61] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for MATLAB. In *Proc. ACMMM*, 2015.

[62] D. Warde-Farley and I. Goodfellow. Adversarial perturbations of deep neural networks. *Advanced Structured Prediction*, 2016.

[63] W. Xu, D. Evans, and Y. Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proc. NDSS*, 2018. To appear.

[64] W. Xu, Y. Qi, and D. Evans. Automatically evading classifiers. In *Proc. NDSS*, 2016.

[65] T. Yamada, S. Gohshi, and I. Echizen. Privacy visor: Method based on light absorbing and reflecting properties for preventing face image detection. In *Proc. SMC*, 2013.

[66] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu. Dolphinatack: Inaudible voice commands. In *Proc. CCS*, 2017.

[67] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. Under submission to ICLR 2018.