Neurocomputing 499 (2022) 11–22

Contents lists available at ScienceDirect

Neurocomputing

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

Enhancing real-world adversarial patches through 3D modeling of complex target scenes ☆

Yael Mathov *, Lior Rokach, Yuval Elovici

Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Beer-Sheva 8410501, Israel

ARTICLE INFO

Article history:
Received 8 August 2021
Revised 28 February 2022
Accepted 6 May 2022
Available online 11 May 2022
Communicated by Zidong Wang

2020 MSC:
68T07
68T27
68T45

Keywords:
Adversarial example
Adversarial learning
3D modeling

ABSTRACT

Adversarial examples have proven to be a concerning threat to deep learning models, particularly in the image domain. While many studies have examined adversarial examples in the real world, most of them relied on 2D photos of the attack scene. As a result, the attacks proposed may have limited effectiveness when implemented in realistic environments with 3D objects or varied conditions. Some studies on adversarial learning have used 3D objects, however in many cases, other researchers are unable to replicate the real-world evaluation process. In this study, we present a framework that uses 3D modeling to craft adversarial patches for an existing real-world scene. Our approach uses a 3D digital approximation of the scene to simulate the real world. With the ability to add and manipulate any element in the digital scene, our framework enables the attacker to improve the adversarial patch's impact in real-world settings. We use the framework to create a patch for an everyday scene and evaluate its performance using a novel evaluation process that ensures that our results are reproducible in both the digital space and the real world. Our evaluation results show that the framework can generate adversarial patches that are robust to different settings in the real world.

1. Introduction

Advances in deep learning models have contributed to the development of solutions for challenges previously considered unsolvable. However, a concerning vulnerability in those models was discovered: An imperceptible perturbation to a legitimate input sample creates an adversarial example that causes the model to output an incorrect prediction with high confidence [1]. Although adversarial examples were initially observed in the digital space [2,3], they were later demonstrated in the real world [4], making the threat even more dangerous. While several studies have presented various use cases for real-world adversarial perturbations in the image domain [5–7], they all utilized a similar methodology: First, one or more 2D photos of the target scene are used to craft an adversarial perturbation that can be digitally added to the entire image or applied as a patch to a portion of an image (known as an adversarial patch). Then, the perturbation is recreated in the real world (i.e., printed) and placed in the scene. Finally, photos of the scene with the adversarial perturbation are fed to the neural network for evaluation.

Although that methodology has shown promising results, such 2D image-based methods do not accurately represent the 3D real world (as shown in Fig. 1). The main challenge stems from the fact that while both the photos of the scene and the adversarial perturbation are flat, a real-world scene is not. A flat patch must be placed on a flat surface (e.g., a stop sign [6]) and must always face the camera. If the adversarial patch is placed on a curved surface, like a traffic cone, parts of the patch will be hidden, and the attack may fail. Furthermore, crafting the adversarial perturbation based on photos of the scene limits the attacker's ability to model real-world properties as part of the attack. Changes to the environmental settings, such as lighting, must be manually added to the real-world scene, and only then can the attacker use them as part of the attack. In addition, such attacks can only be implemented when the attacker controls the entire target scene, which is not a realistic assumption.

Several studies focused on crafting adversarial perturbations for 3D objects. A prominent study [8] presented the expectation over transformation (EOT) framework for crafting adversarial perturbations that are robust to random transformations (e.g., rotation, translation); the framework relies on the attacker's ability to

☆ This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

* Corresponding author.
E-mail addresses: yaelmath@post.bgu.ac.il (Y. Mathov), liorrk@bgu.ac.il (L. Rokach), elovici@bgu.ac.il (Y. Elovici).

https://doi.org/10.1016/j.neucom.2022.05.031
0925-2312/

Fig. 1. Some of the challenges that should be addressed when crafting an adversarial patch for a real-world scene: 1) The patch is placed on a curved surface. 2) The patch is partially hidden by the object it is placed on. 3) The patch should have the same lighting as the rest of the scene. 4) The patch is hidden by an object that is placed in front of it. 5) The objects casting shadows on one another. 6) The scene includes an object that affects multiple objects (e.g., an object casts a shadow on more than one object). 7) The scene may have different environmental conditions (i.e., point yellow light). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

model those transformations as part of the attack process. Athalye et al. [8] used EOT to perturb the texture of a digital 3D object, which was later printed using a color 3D printer. Although successful, the attack targeted a single object, as opposed to the complex scenes that are more common in the real world. As Fig. 1 shows, a realistic scene contains environmental characteristics (e.g., ambient lighting) and different objects that can affect one another (e.g., hide, shadow). Failing to consider those attributes can impair an attack's performance. Additionally, EOT relies on a specific implementation (limiting the ability to use external modeling and rendering tools) and a 3D color printer, a piece of equipment that is not accessible to many people. Both issues complicate the attack and make it less feasible.

In this paper, we present a framework that utilizes 3D computer graphic methods to craft adversarial patches that can be added to an existing real-world scene. Our approach allows the attacker to target complex realistic scenes with multiple objects, have varied environmental characteristics, etc. The framework uses a digital 3D replica of the target scene to simulate the real world, thus allowing the attacker to assess the limitations of the patch and improve it without risking detection. Since the attacker controls the digital replica, he/she can add, change, and remove different elements and as a result, can create an adversarial patch that is more robust to real-world transformations. The framework is designed to allow the integration of external modeling and rendering tools, which gives the attacker the flexibility to implement the attack using the tools he/she prefers. We implement the framework using open-source tools, and thus the attack is also accessible to an attacker with a limited budget.

Additionally, we propose an evaluation process that is specifically designed so that the experiment can be performed multiple times and replicated in future studies. The two-step evaluation process is used in several experiments to evaluate our attack in both the digital space and the real world. We also examine whether the use of a digital replica of the scene to create and improve an adversarial patch can simulate the patch's performance in the real world. To do so, we use the framework to create adversarial patches of multiple target classes and use the evaluation process to compare their performance in the digital space and the real world. Our results show that using the digital replica to evaluate the adversarial patch can provide useful information about the patch's performance in the real world. Furthermore, in realistic settings, unexpected changes may occur in the target scene. Therefore, we examine the patch's robustness to changes in the scene that were not modeled when the original digital replica was produced. To do so, we evaluate how changing or adding new objects to the scene affects the patch's ability to fool the target neural network. Finally, by publishing our evaluation setup and code [9], researchers can reproduce our results and improve upon them.

The main contributions of this study are:

- We present a framework for crafting and improving adversarial patches for an existing real-world scene in the risk-free environment of the digital space.
- We demonstrate how the framework can be used to craft low-budget adversarial patches using free, open-source, and common 3D modeling tools.
- We present an evaluation process that enables reproducible experiments in the digital space and the real world.

2. Background

When discovered by Szegedy et al. [1], adversarial perturbations were considered a minor bug, but that changed with the development of advanced adversarial attacks and the growing number of failed attempts to defend against them [2,3,10]. Adversarial perturbations' success at fooling neural networks led to increased interest in implementing physical attacks [4,11]. While the first real-world studies performed were unable to reproduce the results obtained in the digital space, these studies set the stage for research on novel forms of adversarial perturbations. One example is adversarial patches, which are small shapes, which, when added to a specific part of an image, have been shown to fool object classifiers [6,12] and object detectors [7,13]. Adversarial patches can also be printed and placed in a real-world scene, but their performance is limited. In all of these cases, the adversarial patches were crafted using a set of 2D photos of the target scene but were used in a 3D space; as a result, the flat patch produced would be unable to address the real-world challenges presented in Fig. 1. Such challenges accelerated the development of methods aimed at improving the robustness of adversarial perturbations in real-world settings.

The use of the EOT framework, presented by Athalye et al. [8], to craft perturbations that are robust to specific transformations offered a solution. This framework builds a set of images by transforming the original sample, using parameters that were randomly sampled from the transformation function's distribution; the resulting set of images is then used to craft a robust adversarial perturbation. For example, to craft an adversarial example that is robust to rotation, the training set includes samples of the original image rotated at different angles. To demonstrate the framework's abilities, EOT was used to perturb the texture of a digital 3D object, which was later printed in the real world using a color 3D printer. The study focused on perturbing the texture of a single digital object and thus did not consider the challenges of a complex realistic scene (presented in Fig. 1). The EOT framework was demonstrated on a premodeled digital 3D object that was later printed in the real world, however an attacker is more likely to target an existing real-world scene. Moreover, the implementation of the rendering process and use of a color 3D printer complicate the attack, making it less accessible to inexperienced attackers.

While some studies examined how the representation of 3D data affects the model's robustness to adversarial perturbations [14], others suggested methods for perturbing the structure (mesh) or texture of 3D objects [15,16]. However, these methods focused on manipulating a single 3D object thus failing to consider the unique characteristics of a complex real-world scene. Zeng et al. [17] proposed a more realistic approach which perturbs different elements of a digital 3D scene. The output was used to gain more insight about attacks in the real world. Although using 3D objects to create realistic adversarial examples seems promising, the studies mentioned above used premade digital objects, and did not target an existing complex real-world scene or examine the attack outside of the digital space.

3. Suggested approach

3.1. Assumptions and threat model

In this study, we assume that a user is operating an automated object recognition system to monitor a specific scene. The system films the scene, sends the photos to a deep learning-based object classifier, and provides the output label to the user. The attacker wants to hide a specific object (e.g., a weapon or a specific face) from the system's deep learning model. In order to accomplish this, the attacker plans to add a sticker to the scene that obfuscates the object. By doing so, photos of the scene, which serve as input to the object classifier, will result in a false prediction by the learning model.

We assume the attacker has complete knowledge of the target neural network (e.g., parameters, architecture) but has no knowledge on the other system components (e.g., camera). The attacker also has physical access to the actual scene, allowing him/her to add the patch in the real world. Moreover, the attacker can examine the real-world scene to create a 3D digital replica of it, such that the neural network classifies the rendered images of the replica as the original class. The minimum hardware requirement for this task is a NVIDIA GTX 1060 GPU [18], but even the best available graphic processor is affordable. As we show in this study, our approach is effective even when the attacker has limited experience in 3D modeling. Therefore, we can assume that the attacker has the knowledge and resources to perform this attack.

We note that while this study demonstrates an attack under white-box settings, our framework can use black-box attack methods to craft an adversarial patch under more restrictive settings.

3.2. Framework overview

In this section, we describe the proposed framework for crafting an adversarial patch for an existing real-world scene. Fig. 2 presents the framework's six steps: modeling, rendering, combining, crafting, evaluation, and application. The first five steps are performed in the digital space, while in the sixth step, the adversarial patch is transferred from the digital space to the real world.

First, the attacker uses 3D modeling techniques to create a 3D digital replication of the real-world scene and adds an empty digital patch object to the replica (in the following steps, the texture of the empty object is perturbed to craft the adversarial patch). Second, the digital replica is rendered into 2D images, such that each image is a rendering of the scene under real-world transformations (e.g., rotation, a change in the lighting). The rendering process results in two outputs: a background image of the scene (without the patch) and the properties of the patch (all pixels that include the patch). Third, the attacker uses a differentiable method to combine the background and patch properties into one image to create a set of images of the scene. This way, the attacker can use a wide variety of external rendering tools without implementing a differentiable rendering process or approximating the gradients of a non-differentiable rendering. Fourth, the images are then used to perturb the patch's texture, using any method of crafting an adversarial perturbation. Fifth, the attacker adds the adversarial patch to the scene's digital replica, renders images of the scene, and feeds them to the neural network. As a result, the attacker can examine the patch's effect on the scene and improve it if needed. Finally, the attacker prints the final patch on a sticker and adds it to the real-world scene.

3.2.1. Modeling a digital replica of the real-world scene

The first step is to create a 3D digital replica that approximates the real-world target scene. To build this replica, the attacker can choose any tool (e.g., Blender, Maya) or resources (e.g., use free 3D objects, buy 3D designs, use a 3D scanner). Then the attacker adds an empty 3D object to the digital scene to serve as the adversarial patch in the replica. In this step, the attacker should consider the following elements: the objects in the scene, the ambient characteristics, and the adversarial patch. First, the replica is built from digital 3D objects that represent the objects in the real-world scene. Then, the scene's ambient characteristics (e.g., light sources, smoke) are added to the replica to improve the replica's similarity to the real world. Finally, the digital patch object is added to the scene according to its expected location in real life. Decisions regarding the location, shape, and size of the patch should be made based on the attacker's goals and the camera's expected location.

The modeling stage is affected by the expertise and resources of the attacker. Our findings show that a successful attack can be launched by roughly approximating the target scene, however when more realistic replications are modeled, the success rate will likely increase. This finding demonstrates a trade-off between the attacker's effort and the success rate, which can be used to balance the goals and capabilities of the attacker.

3.2.2. Rendering 2D images with realistic transformations

The EOT process uses a collection of images with different transformations ("views") to craft a single perturbation that fools the target neural network for all views. Unlike past studies that could only use EOT with a limited number of transformation functions, our framework's design allows the attacker to transform any of the scene's properties, including the scene's objects (one or more), ambient characteristics, camera view, and more. As presented by Athalye et al. [8], flexibility in choosing the transformations results in a better adversarial perturbation. Therefore, in this step, the attacker aims to create a collection of views, such that each view represents the digital replica under a different set of transformations.

We define $T = \{T_1, \ldots, T_k\}$, such that each T_i $(1 \leqslant i \leqslant k)$ is a distribution of transformation functions t_i on the digital replica (i.e., the scene). For example, a transformation function that rotates the scene through an angle θ around the x-axis can be sampled from $U(30, 90)$; hence, $\theta \in [30, 90]$. Similarly, we define C as the distribution for the transformation functions on the digital replica's camera (e.g., the camera's position). For each set of transformation functions sampled from T and C, the rendering process R applies the transformations to the 3D digital replica S with patch texture P and outputs a 2D image of the 3D scene with the t transformations from viewpoint c (i.e., a view). In this step, the attacker samples transformation functions multiple times to build a collection of views X:

$$X = \mathbb{E}_{t\sim T, c\sim C}[R(S, P, t, c)] \tag{1}$$

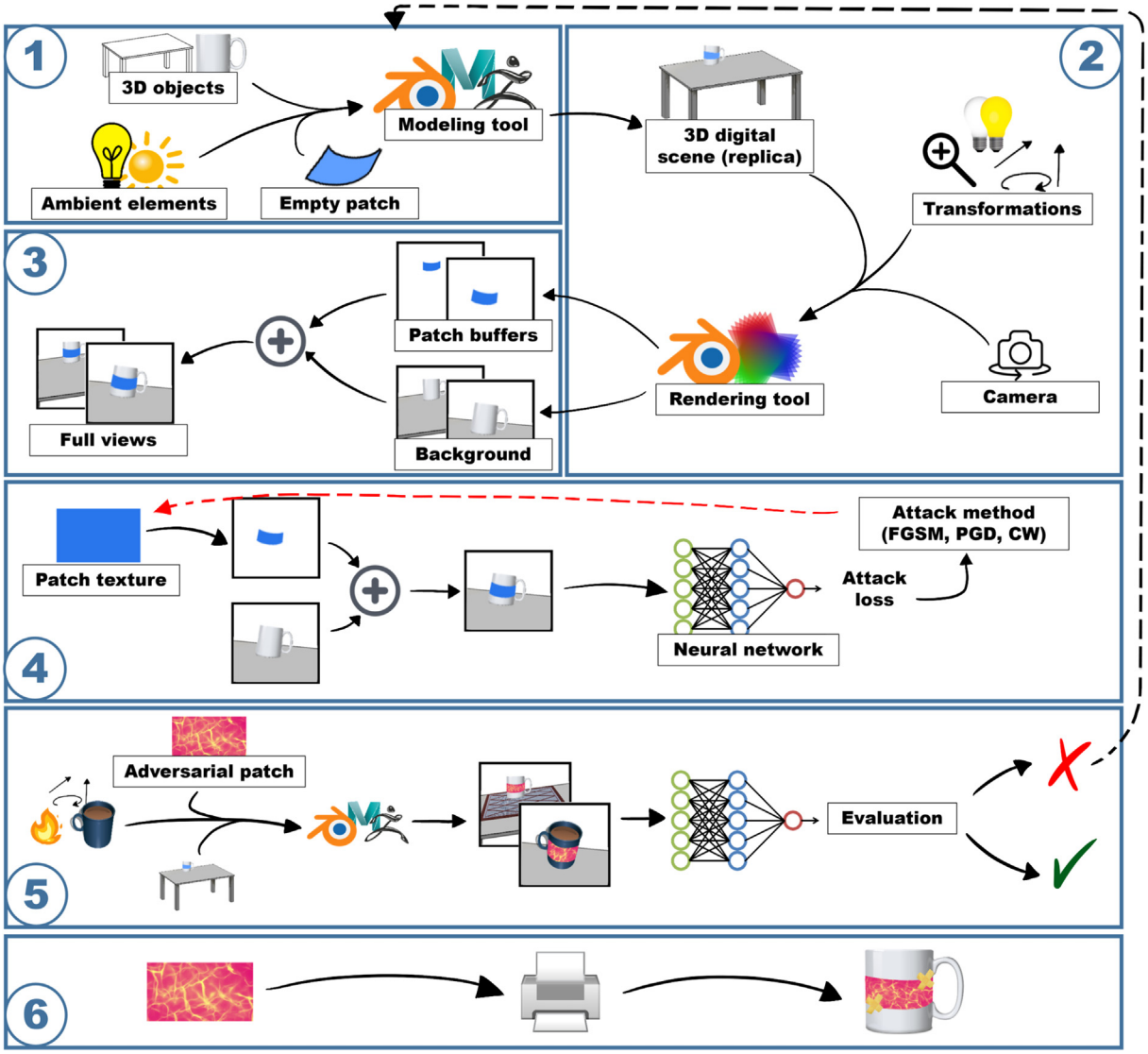


Fig. 2. The framework's steps: (1) Modeling: Use 3D objects, ambient elements, and an empty patch object to model a digital replication of the real-world scene. (2) Rendering: Use the 3D replica, scene, and camera transformation to render 2D images of the scene with the transformations (views). For each view, the output is the background and the patch property buffers. (3) Combining: Use a differentiable method to combine the two types of buffers into full images of each view. (4) Crafting: Use any attack method to perturb the texture of the adversarial patch. (5) Examining: Add the adversarial patch to the 3D replica and render images of the scene with different changes (e.g., change the mug). If the adversarial patch does not fool the neural network, go back to step 1 and improve the attack. (6) Applying: Print the chosen adversarial patch and apply it to the physical scene.

3.2.3. Combining the rendering output buffers

In the next step (i.e., crafting), the set of views is used to perturb the patch's texture P to create the adversarial patch. As shown in previous studies, crafting the adversarial perturbation is usually done by solving an optimization problem by calculating or approximating the attack's loss gradient concerning the patch. Because our framework allows the use of external rendering tools, an attacker can use a non-differentiable rendering process to craft X. As a result, the attacker cannot calculate $\frac{\partial X}{\partial P}$ thus preventing him/her from using gradient-based methods to craft adversarial perturbations. Past studies overcame this issue by implementing the rendering process as a differentiable part of the attack [8,15] or by approximating the loss gradient [16,17]. However, these solutions prevent the attacker from using external tools and increase the knowledge required to implement an attack.

Therefore, we suggest modifying the output of the rendering process, which is a simple configuration change that can be done by almost any rendering tool. Then, instead of outputting a single image, the result of the rendering process is buffers of two types: the background and the patch's properties. The background buffer is an image of the scene without the patch, and the patch's properties form a set of buffers with information that allows the patch to be added to the background in a realistic manner; combining the two types of buffer creates an image of the scene. Hence, the rendering step results in a set $X = \{(b_i, p_i)\}_{i=1}^{n}$, where for sample $x_i = (b_i, p_i) \in X$, b_i is the scene background, and p_i are the patch properties. Additionally, for a sample x_i, let $B(P, b_i, p_i)$ be a differentiable method that combines P, b_i, and p_i into an image of the rendered scene. Given the result of the rendering step X, the attacker builds a set of views $\widetilde{X}$ that are differentiable by P:

$$\widetilde{X} = \{B(P, b_i, p_i) : \forall (b_i, p_i) \in X\} \tag{2}$$

An example of combining buffers into a scene image is presented in Fig. 3. The background buffer b_i is an image of the scene without the patch, and the two patch property buffers p_i are the patch's texture map and lighting. Each pixel in the texture map

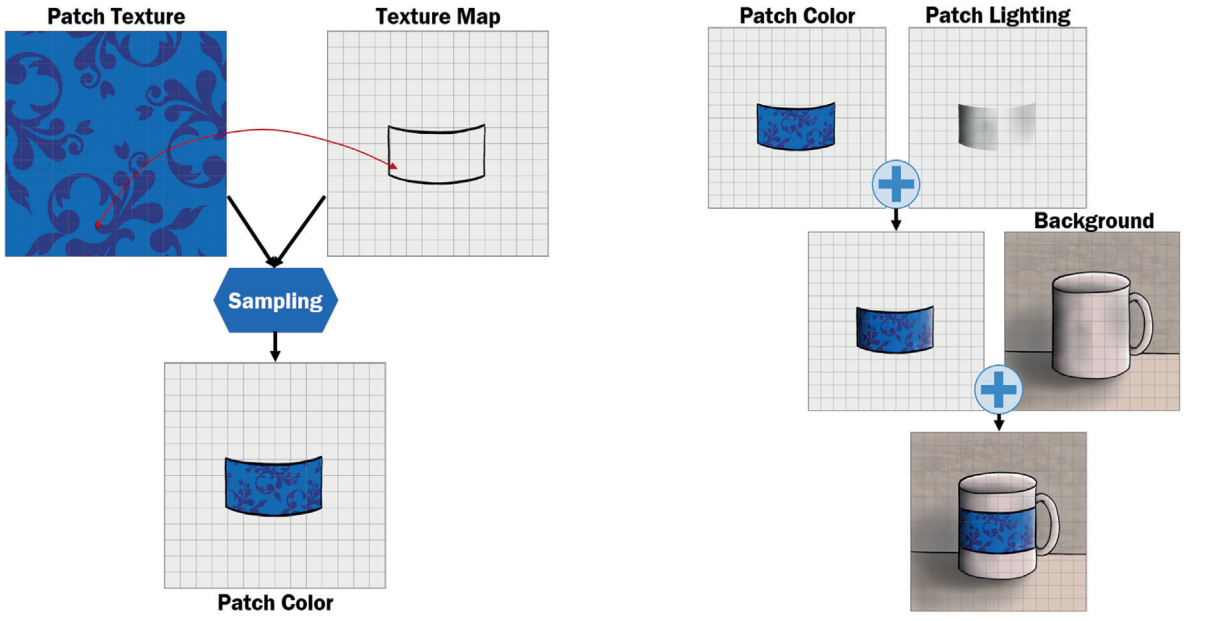


Fig. 3. Creating the images of the scene from the patch property and background buffers. (a) Sampling the texture of the patch to create an image of the patch's colors. (b) Combining three buffers to create an image of the scene: the patch's colors, the shading of the patch, and an image of the background without the patch.

contains the coordinates of a pixel in the patch's texture P, thus allowing the framework to build an image of the patch's colors by sampling pixels from P (shown in Fig. 3a). Then, the patch's color and lighting and the scene background are combined into one complete image of the complete scene (shown in Fig. 3b). Since the image is built using simple operators, like sampling P, addition, and multiplication, the image $B(P, b_i, p_i)$ is differentiable by P.

3.2.4. Crafting the adversarial patch

After combining the buffers into a set of images $\widetilde{X}$, the attacker can use them to craft the adversarial patch. Crafting an adversarial perturbation under white-box settings is usually done by feeding a benign input sample to the target learning model, calculating both the attack loss based on the model's output and the loss gradient with respect to the input sample, and perturbing the original input sample. The attack loss is used to ensure that the adversarial perturbation meets the attack's requirements, such as fooling the target model to output a specific result, limiting the perturbation size, etc. Given our framework's design, most methods for crafting adversarial perturbations [1–3,10,4,11] can be used in this step to perturb the texture of the empty patch into an adversarial patch.

Since combining the rendering output buffers results in a set of images $\widetilde{X}$ that are differentiable by the patch texture P, the attacker can use any gradient-based method to craft an adversarial perturbation for P. Most methods perturb the patch texture P to fool a neural network M by optimizing an objective function with a customized attack loss $\mathscr{L}$, e.g., finding P such that $\text{argmin}_P\left(\mathscr{L}\left(M\left(\widetilde{X}\right)\right)\right)$. The attacker can construct $\mathscr{L}$ to create a targeted attack, apply constraints, etc. Then, since it is easy to calculate $\frac{\partial \widetilde{X}}{\partial P}$, the attacker can also calculate $\frac{\partial \mathscr{L}\left(M\left(\widetilde{X}\right)\right)}{\partial P}$ to solve the optimization problem using a gradient-based optimizer (e.g., gradient descent). As done in previous studies, the objective function and optimization method should be selected according to the attack's goal and target neural network. While in this study we demonstrate our framework with a white-box attack, future work can replace the attack method we used in this step with a black-box attack (such as [19,20]) to craft an adversarial patch under more restrictive settings.

3.2.5. Examining the patch in the digital space

Next, by adding the patch to the digital replica, the attacker can simulate the adversarial patch's effect on the neural network's predictions concerning the real-world scene. Examining the adversarial patch in the real world requires both the attacker's presence at the original scene and the performance of actions that could be considered abnormal thereby exposing the attacker to the risk of detection. Using the digital replica to simulate the real-world scene allows the attacker to identify potential problems and improve the patch in a controlled and safe environment. Since the attacker controls the digital replica, he/she can simulate events that are challenging to control in real life (e.g., waning daylight, the presence of smoke).

Moreover, the attacker can use the digital replica to compare different adversarial patches, identify the most effective one, and improve the attack's success in the real world. Based on the findings in this step, the attacker might choose to change the attack process, necessitating modifications to the scene's replica, the addition of new transformations to the rendering step, or changes to the attack's optimization function. Since the examination process is performed in the digital space, the attacker can improve and evaluate the patch as long as he/she wishes.

3.2.6. Applying the patch to the physical scene

Finally, the attacker creates the adversarial patch in the real world and adds it to the scene. For instance, the patch can be printed on a sticker or a piece of paper using a home printer.

4. Experimental setup

In this study, we implement the framework using free, open-source software to create an adversarial patch for a typical office scene in which a standard white mug is placed on a desk, as seen

in Fig. 4a. A webcam (Microsoft LifeCam VX-700) films the scene, and then we divide the video stream into photos, crop them into 299×299 pixel color images, and feed them to a state-of-the-art object classifier, InceptionV3 [21]. This classifier was trained on the ImageNet dataset [22], achieving 1-top accuracy of 78% and 5-top accuracy of 93.9% for valid input. We validated that the scene is classified as the original class (i.e., Coffee Mug) for 100% of the images rendered from the digital replica without the patch. Additional information on the experimental setup, code, and parameters is available in [9].

4.1. Implementation of the proposed framework

4.1.1. Modeling

To model the replica of the target scene, we use Blender [23], a free creation software. We start by approximating the objects in the real-world scene: For the desk and the walls of the room, we use default cube shapes in Blender. However, the other objects are more complex, and professional expertise in 3D modeling is required to create them from scratch. Therefore, we use a free, premade mug object [24] and add it to the digital replica. Afterward, we add a yellow point light source that simulates the light bulb in the original office. Then, to create the empty placeholder for the adversarial patch, we crop and edit a portion of the mug model [24] and wrapped the result around the mug's 3D object in the digital scene. For each 3D object, we choose standard configurations for the materials and use textures made from photos of the real-world scene. We note that the replica is modeled using online tutorials for beginners to produce a simple approximation of the real-world scene. Therefore, while our digital scene lacks some realistic elements (Fig. 4b), it can be easily replicated by attackers and researchers with limited experience in 3D modeling.

4.1.2. Rendering

To create the set of views in Eq. 1, we use the Python library ModernGL [25]. The rendering process is implemented based on the examples in the library's repository and uses the configurations and materials to portray the ambient elements in the rendered images (e.g., shadows). Due to the attacker's assumed lack of knowledge about the camera, we estimate the camera's configurations (e.g., field of view). Then, we render a set of views X, i.e., images of the scene under different transformations: For the scene transformations T, we choose translation and rotation in the x, y, and z axes, and changes in the light color, and for the camera transformation C, we choose changes in the camera's position. To determine the ranges that define each distribution of the different transformation functions, we examine both the digital replica and the real-world scene. Based on our findings, we define the ranges according to possible changes in the real-world scene while ensuring that the mug and patch are visible in all views. Additional information about the rendering step (including parameters and code) is available in [9].

In this study, we explore two methods for sampling the parameters for the transformations: *random sampling* and *systematic sampling*. Random sampling is commonly used in the EOT framework, in which a transformation function t_i is randomly sampled from a uniform distribution $T_i \sim U(\alpha_i, \beta_i)$; hence, the parameter that defines t_i is randomly sampled from the range $[\alpha_i, \beta_i]$. We also examine a deterministic approach, which we refer to as systematic sampling, where the transformation functions t_i are predefined by systematically sampling the function's parameter in constant even steps across $[\alpha_i, \beta_i]$. For example, we want to create l scene rotation functions $\{t_{i,1}, \ldots, t_{i,l}\} \in T_i$ that are systematically sampled from the range $[\alpha_i, \beta_i]$. Therefore, we predefine $\{t_{i,1}, \ldots, t_{i,l}\}$, such that for each $1 \leqslant j \leqslant l$, the rotation function $t_{i,j}$ rotates the scene at an angle of $\theta_j = \alpha_i + \frac{j \cdot (\beta_i - \alpha_i)}{l}$. After selecting the transformation functions, we build the set of views by rendering an image of the scene using every combination of those parameters for the different transformation distributions.

4.1.3. Combining

The rendering step outputs a set of views X that is determined by the sampling method; hence, each view is defined by a set of scene and camera transformations. Additionally, for each view, the rendering process outputs four buffers outputs the following four buffers: one buffer with the background image and three buffers that contain the patch properties. The patch property buffers include the texture mapping, lighting, and a mask that defines the parts of the patch object visible in the rendered image. To build a differentiable image from the four buffers, we follow a similar process to the one presented in Fig. 3: We use the texture mapping to sample the colors from P, merge the result with the patch's light-

(a) Real world (b) Digital replica

Fig. 4. The target scene: a white mug placed on an office desk; (a) is a photo of the original real-world scene; and (b) is a rendering of our digital replica with the empty adversarial patch (blue strip) that was modeled using Blender. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

ing, and finally, use the mask to combine the background image with the patch. The result is an image of the complete scene with the patch, similar to the one shown in Fig. 4b. The building process is implemented using TensorFlow [26] and uses simple operations such as sample, add, and multiply tensors; as a result, the output image is differentiable by P. We use this process for each view in X to build a set of differentiable images $\widetilde{X}$.

4.1.4. Crafting

To craft the adversarial patch, we follow previous studies and define an objective function with a customized attack loss:

$$\mathscr{L}\left(\widetilde{X}, P\right) = CE\left(\widetilde{X}, y_{tg}\right) - \kappa \cdot CE\left(\widetilde{X}, y_{og}\right) + \lambda \cdot TV(P) \tag{3}$$

where κ and λ are tuning parameters, y_{og} and y_{tg} are the original and attack target class respectively, CE is the cross-entropy loss, and TV is the total variation. Then, to perturb P, we use the Adam optimizer to solve the following optimization problem: $\text{argmin}_P\left\{\mathscr{L}\left(\widetilde{X}, P\right)\right\}$. Eq. 3 uses the cross-entropy loss to cause the scene with the patch to be classified as y_{tg} and not as y_{og}, while maintaining the smoothness of the patch by minimizing the total variation distance. Examples of two adversarial patches are presented in Fig. 5.

4.1.5. Examining

To examine the patch, we use the digital evaluation process, which is described in Section 4.2. Based on our findings, we perform several changes in the transformation ranges and updates to the 3D replica to improve it (e.g., creating realistic lighting by modeling the room as a box instead of three visible walls). The attack was initially designed for the Armadillo target class; later, we change the seed and initialization parameters and use the attack to craft patches for the Armadillo target class and nine additional target classes. We use the same attack to perform a non-biased comparison between the patches of the different target classes. However, it is more realistic to build a tailored attack for each target class, and we plan to explore this approach in future research.

4.1.6. Applying

We print the patch on an A4 piece of paper using a Xerox WorkCentre 6605, manually crop it, and apply it to the mug using transparent adhesive tape.

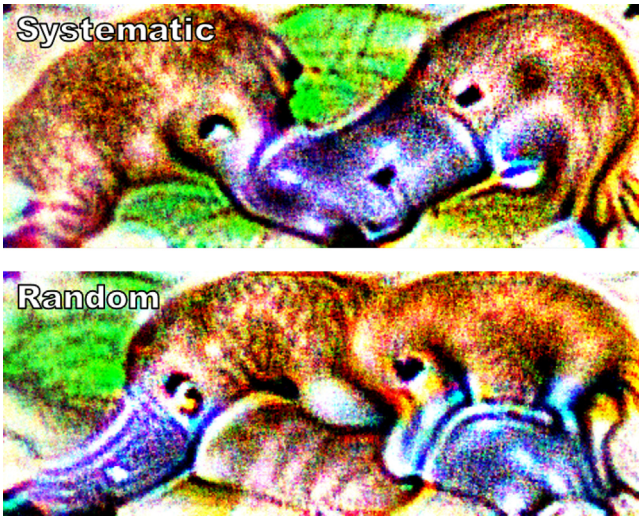


Fig. 5. The systematic (upper) and random (bottom) adversarial patches for the Platypus target class.

4.2. Evaluation process

The ability to perform the same experiment multiple times under the same conditions and obtain similar results is essential for our study's integrity and to enable future research to reproduce our work. Therefore, we present a replicable two-step evaluation process, the first step of which takes place in the digital space; the second step takes place in the real world. It is important to note that the evaluation process was designed for research purposes and is not part of our framework.

4.2.1. Digital space

In the first step, the digital replica is used to simulate the evaluation in the real world, similarly to our framework's examination step (step 5 in Fig. 2). Here, we add the adversarial patch to the scene, render a set of images under the expected real-world settings and transformations (e.g., camera position), send the images to the neural network, and analyze the predictions.

4.2.2. Real-world

To ensure that the real-world evaluation process is reproducible, we suggest using the evaluation setup presented in Fig. 6. The setup includes a camera slider that can be placed at different distances from the scene and allows the camera to film the scene from predefined positions. As shown in Fig. 7a, the camera is placed on a spinning platform, which is located on a cart that moves the camera from side to side in front of the scene. The position of the camera can be changed by spinning the platform to a specific angle, which is indicated by marks on the gradations on the base of the platform. Then, a screw is used to secure the camera in place to ensure that the camera angle remains fixed during the experiment. A motor spins a screw rod, which moves the cart across two metal rods at a constant speed. We suggest defining small ranges on the slider, as shown in Fig. 7b, from which the scene is filmed at different angles. For example, after choosing a camera position, we identify the range on the slider in which the mug is visible to the camera; during the experiment, the cart moves only in the defined range to avoid filming irrelevant parts of the scene. Each range should be defined based on the required observation area (from the center, left, or right sides of the slider), along with the corresponding position of the camera, and can be set by physically limiting the cart's movement or by configuring the motor's behavior. Additionally, as shown in Fig. 7c, the slider can be moved forward and backward across a grooved base to film the scene from different distances. Finally, since the real-world scene might change, the location and position of each object (including the adversarial patch) must be marked, thus ensuring that the same scene can be re-evaluated. By marking each configuration, the same actions can be performed in future experiments and thus, ensuring that similar results are achieved.

4.2.3. Our evaluation setup

In this study, we use the evaluation process described above, in both the digital space and the real world, to evaluate the performance of the patches we create using our framework.

Digital space: We identify 3,360 different positions and locations combinations for the camera around the object, based on the patch's visibility in the digital replica; for each of them, we render an image and send it to the object classifier.

Real world: We build the evaluation setup, as shown in Fig. 6 and Fig. 7 and described above, with the following configuration: The slider can be placed at one of three predefined distances from the scene (close, middle, and far). We define three ranges (left, center, and right) for each distance and mark each of the nine ranges using metal eye straps. The location of the mug, the position of the mug's handle, and the location of the patch on the mug are

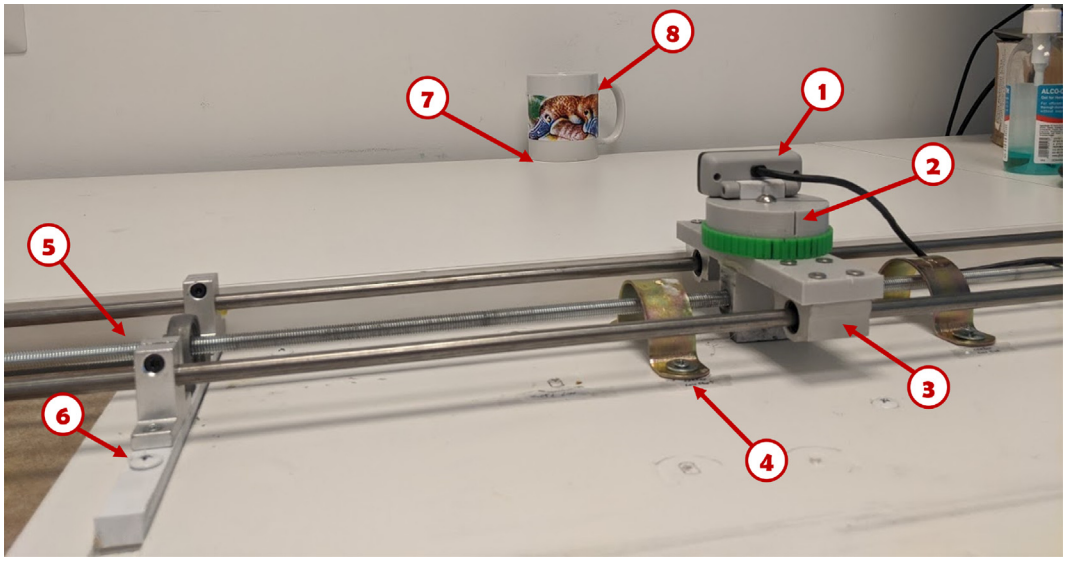


Fig. 6. Our real-world evaluation setup: The position of the camera (1) is controlled by spinning a platform which has gradations on its base (2). The platform is located on a cart (3), which moves within predefined ranges (4) on a spinning screw rod (5). The slider can be moved forward and backward into one of three predefined distances (6). We can rebuild the scene using the markings for the mug's location and position (7) and the patch's location on the mug (8).

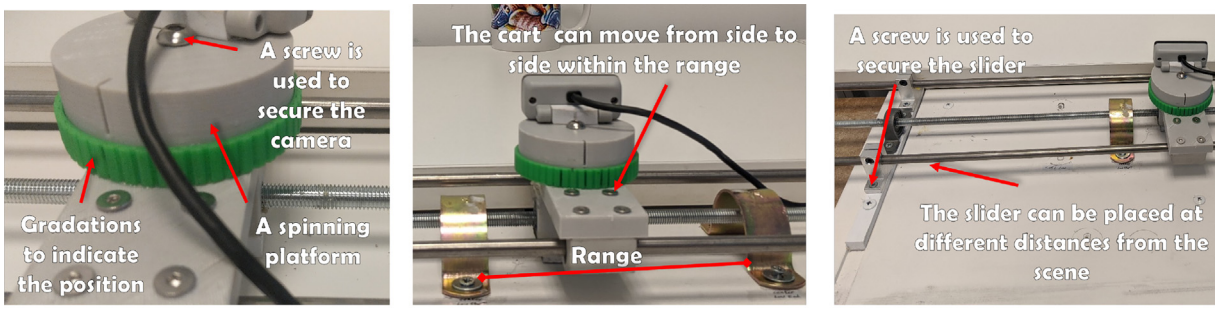


Fig. 7. A closeup of the three main components of the real-world evaluation setup: (a) the spinning platform for the camera, (b) the cart that moves the camera in front of the scene, and (c) the slider, which can be placed at different distances from the scene.

marked, thus ensuring that the same scene is used throughout the various experiments. Sketches and instructions for replicating our real-world evaluation setup are available in [9].

5. Results

For each target class, we craft four patches: *random, systematic, google,* and *imagenet*. The *random* and *systematic* patches are adversarial patches crafted by using our framework with the random or systematic sampling of the transformations during the rendering step. The *google* and *imagenet* patches are made from images of the target class that were obtained from a Google search or the ImageNet dataset respectively; these patches are used to ensure that the results are non-biased. We note that the images used for the *imagenet* patches were obtained from the dataset that was used to train the classifier. We also examine the classification results for a mug without a patch (*clean*) and a patch with random pixel values (*noise*). For each target class and its four patches, we evaluate the percentage of images classified as the original class (Coffee Mug), target class, or other classes (from the ImageNet dataset).

5.1. Results of the evaluation in the digital space

After changing the seed and initialization parameters, we craft *random, systematic, google,* and *imagenet* patches for the following classes: Armadillo, Hat, Nipple (bottle), Platypus, Mask, Pencil Box, Syringe, Screw, Mousetrap, and Ladybug. Then, we perform the evaluation process in the digital space for *clean*, with the *noise* patch, and with each of the patches described. All of the 3,360 images of the rendered replica for *clean* and with the *noise* patch are classified as the original class, Coffee Mug. Table 1 presents the results for the patches.

We expected that the *imagenet* patches, which were taken from the dataset that was used to train Inception V3, would act like an adversarial patch thus causing the scene to be classified as the target class. However, although the scene is simple, both the *google* and *imagenet* patches, with clear images of the target class, do not affect the classifier's predictions. On average, 99% of the rendered images of the replica with the *google* and *imagenet* patches are classified as the original class, and none of them are classified as the target class. In contrast, more than 99% and 97% (on average) of the images are classified as the target class for the *systematic* and

Table 1
The classification results (percentage) in the digital space for the original (Og), target (Tg), and other (Ot) classes. All of the images (100%) of *clean* and with the *noise* patch are classified as the original class.

	Systematic			Random			Google			Imagenet		
Target Class	Og	Tg	Ot	Og	Tg	Ot	Og	Tg	Ot	Og	Tg	Ot
Armadillo	0.3	99.5	0.2	0.8	98.5	0.7	98.1	0	1.9	98	0	2
Hat	0.3	99.6	0.1	0.1	99.5	0.4	99.5	0	0.5	99.1	0	0.9
Nipple	0.1	97.6	2.3	0.4	93.1	6.5	99.6	0	0.4	100	0	0
Platypus	1	97.5	1.5	1.3	98	0.7	98.1	0	1.9	96.4	0	3.6
Mask	1	99	0	6.9	92.9	0.2	99.7	0	0.3	99.7	0	0.3
Pencil Box	0	99.9	0.1	0.7	98.7	0.6	99.7	0	0.3	99.9	0	0.1
Syringe	0	98.9	1.1	0.8	95	4.2	99.2	0	0.8	97.6	0	2.4
Screw	0.1	99.8	0.1	0.6	97.6	1.8	100	0	0	97.9	0	2.1
Mousetrap	0	100	0	0.1	99.1	0	100	0	0	97.9	0	2.1
Ladybug	0	100	0	0	100	0	99.7	0	0.3	100	0	0

random patches respectively. When comparing the two types of adversarial patches, the systematic sampling approach is significantly better at causing the scene to be classified as the target class than the random sampling approach (the p-value is 0.04 for a paired sample t-test).

5.2. Results of the evaluation in the real world

For the real-world evaluation, we use the same patches we created in the digital space for the following target classes: Armadillo, Hat, Nipple, and Platypus. Each of these patches, including the *noise* patch, is printed on a piece of paper, cropped, and placed on the designated location on the coffee mug using transparent adhesive tape. Then, we follow the setup presented in Fig. 6 to take approximately 700 photos, from all nine ranges, of the real-world scene with each patch and *clean*. Similarly to the evaluation in the digital space, 100% of the photos of the real-world scene for *clean* and with the *noise* patch are classified as the original class. Table 2 summarize the results for the remaining patches.

The results for the real-world evaluation are similar to the results obtained in the digital space. For all target classes, the scene with the *google* and *imagenet* patches is almost always classified as the original class (for 98.5% of the photos, on average) and is never classified as the target class. Similarly, the scene with the *systematic* and *random* adversarial patches is mainly classified as the target class; the average difference between the digital and real-world results is 5%, and the actual difference never exceeds 7%.

We further examine the patches with the greatest difference between the digital space and real-world results. The *google* patch for the Armadillo target class and the *imagenet* patch for the Hat target class are classified as the original class in 98.1% and 99.1% of the images respectively but only in 95.8% and 95.9% of the photos (respectively) of the real-world scene. Both patches are classified as the Candle class from the ImageNet dataset (other) for photos taken from positions in which the mug's handle is less visible; therefore, the misclassification might stem from the cylindrical shape of the mug without the handle. We also observe that the *random* patch for the Platypus class is less effective when the camera is located far from the scene, in both the digital space and the real world. Since the digital space evaluation reveals its weaknesses, the attacker can utilize this information to improve the patch by rendering more views in which the camera is located far from the scene. This is an example of how the attacker can identify the patch's flaws in advance, learn how to improve the patch, and increase the chances of a successful attack.

6. Resilience to unexpected changes

In realistic scenarios, the attacker cannot control the real-world scene, which means that by the time the attacker returns to the scene with the patch, the scene may have changed. As mentioned, the attacker can use our framework to improve the patch's robustness to expected transformations; however, this is not the case for unpredictable major changes of the scene (e.g., the removal of an object). If the patch is only effective in the modeled scene and under expected changes, then our framework is less feasible in real-world settings. Therefore, we should examine how unexpected changes to the real-world scene affect the ability of adversarial patches created using our framework to fool the target neural network. To do so, we apply seven new changes to change the real-world scene. The adversarial patches are not expected to be robust to these changes, since they are not part of the digital replica.

As shown in Fig. 4a, the original scene contains a white coffee mug with a handle on the right side and a patch placed in the middle of the mug, which is placed on an office desk. We refer to the seven unexpected changes as *up, down, red, wood, color, shape*, and *flipped*. *Up* and *down* are changes to the patch's location on the mug: the patch is placed on the top and bottom of the mug, respectively. We also change the surface on which the mug is placed to a red circle (*red*) and a wooden mat (*wood*). For *color* and *shape*, we replace the original white mug; for *color*, we use a different color mug (a dark background with colorful illustrations) with the same shape, while for *shape*, we use a mug with a similar

Table 2
The classification results (percentage) in the real world for the original (Og), target (Tg), and other (Ot) classes. All of the photos (100%) of *clean* and with the *noise* patch are classified as the original class.

	Systematic			Random			Google			Imagenet		
Target Class	Og	Tg	Ot	Og	Tg	Ot	Og	Tg	Ot	Og	Tg	Ot
Armadillo	1.5	92.8	5.7	2.1	95.3	2.6	95.8	0	4.2	99.7	0	0.3
Hat	0	99.1	0.9	0.2	98.9	0.9	99.3	0	0.7	95.9	0	4.1
Nipple	0	98.7	1.3	0.7	92.7	6.6	100	0	0	99.1	0	0.9
Platypus	0.3	92.6	7.1	6.2	87	6.8	99.2	0	0.8	99.1	0	0.9

color (light gray instead of white) which has a different shape. The *shape* mug is shorter and cone-shaped, and has a smaller handle which is located at a higher position on the mug. Finally, for *flipped*, we rotate the original mug by 180°. In this experiment, we apply each change (some of the changes are shown in Fig. 8) to the scene and perform the real-world evaluation process with the two adversarial patches (*systematic* and *random*) for the Nipple target class.

As seen in Table 3, unexpected changes cause the scene to be classified as the target class less often and more often as other classes. Additionally, the scene is classified as the original class in only 2.9% of the photos (on average) and in no more than 6.5% of the photos. This shows that the neural network fails to identify the mug even when there are major unexpected changes in the scene. It seems that the patches are robust to changes in the patch's location on the mug; in *up* and *down*, the scene is classified as the original class in less than 2.7% of the photos. However, changes in the object that the mug is placed on show mixed results; while the patches are more robust to *red*, *wood* is classified as the target class in less than 70% of the photos, and as the original class in nearly 5.8% of the photos. We believe that the difference may stem from the shape of the mats: unlike the wooden mat, the red mat and the desk in the original scene have a solid body. Similarly, replacing the original mug also shows mixed results; both patches perform the worst with *color* but perform well with *shape*. With *color*, the scene is classified as the target class in less than 40% of the photos and as the original class in up to 6.5% of the photos. We assume that the use of a mug with an inconsistent design (i.e., many colorful illustrations) degrades the performance of the patches. Although the patches are the least robust to *color*, the neural network still fails to identify the original class of the scene in more than 90% of the photos. In contrast, the scene with *shape* is classified as the target scene in 71.7% and 87.1% of the photos for the *random* and *systematic* patches respectively, and the *systematic* patch is never classified as the original class. Additionally, we find that *flipped* affects the adversarial patches differently: *systematic* performs well, with 85.8% of the photos classified as the target class, but with *random*, only 55.1% of the photos are classified as the target class. Finally, in most cases, the *systematic* patch performs better than the *random* patch.

This experiment was designed to examine whether using a digital replica limits the attack to a specific version of the scene thus causing the adversarial patch to become ineffective when unexpected changes occur in the real world. To examine this, we performed noticeable, yet realistic, changes to the original real-world scene: adjustment to the object's location and position, the replacement of an object, and the addition of a new object to the scene. The results show that although the changes reduce the attacker's ability to control the neural network's prediction, the patches can still fool the neural network so that it misclassifies the scene. On average, the scene is not classified as the original class in over 97% of the photos, which supports the feasibility of our framework. Therefore, the attacker can improve the attack's robustness to real-world changes by creating a 3D replica of the scene (as we proposed), suffering just a minor reduction in the patch's fooling capability if unexpected changes occur in the target scene.

7. Conclusions

In this work, we demonstrated how 3D modeling techniques and tools can be used to craft inexpensive adversarial patches real-world scenes. By creating a digital replica of the target scene, our method gives the attacker control of every aspect of the scene, including the objects, lighting, and more. The replica simulates the real-world scene, thus allowing the attacker to test and improve the attack without the risk of detection. We also demonstrate that such an approach can improve the patch's robustness to both

Fig. 8. Unexpected changes to the real-world scene; (a) the three coffee mugs on the wooden mat used for *wood* (from left to right: *shape*, the original mug, and *color*), and (b) *red*: the original coffee mug with an adversarial patch on a red mat. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 3
The results of the real-world evaluation for the Nipple adversarial patches with seven unexpected changes applied to the scene. The values presented indicate the percentage of photos classified as the original, target, or other classes.

	Systematic			Random		
Change	Original	Target	Other	Original	Target	Other
Up	0.6	88.9	10.5	2.2	77.9	19.9
Down	2.7	83.7	13.6	2.7	87.8	9.5
Red	1.9	84.6	13.5	2.7	76.5	20.8
Wood	5.8	68.9	25.3	2.4	64.7	32.9
Color	6.5	39.1	54.4	5.5	28.1	66.4
Shape	0	87.1	12.9	1.2	71.7	27.1
Flipped	2.7	85.8	11.5	4.3	55.1	40.6

expected and unexpected changes in the real-world scene. Additionally, we present an evaluation process that enables other researchers to reproduce our experiments and validate our results. We believe that such an evaluation process can be used in future studies and contribute to replicating, examining, and improving other real-world attacks.

However, we acknowledge that a researcher with no background in 3D modeling might find our framework complicated to implement. We address this by publishing all of the resources used in this research, including code, and a detailed guide for recreating our evaluation setup [9]. Additionally, this study focused on a single implementation which was used to target a simple scene with little likelihood of being attacked in reality. Therefore, in future work, we plan to improve the suggested framework by adding 3D elements (e.g., reflections and normal maps) to support more complex scenes. Then, we will use the framework to tailor attacks for other domains, like the facial recognition domain, where our framework can improve individual's privacy concerning such systems. We also aim to create imperceptible perturbations that attract less attention. Finally, our results suggest that using the EOT framework with systematic sampling might be better than random sampling, and we plan to perform additional experiments to examine this further.

CRediT authorship contribution statement

Yael Mathov: Conceptualization, Methodology, Software, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization. **Lior Rokach:** Methodology, Supervision. **Yuval Elovici:** Resources, Supervision, Funding acquisition.

Declaration of Competing Interest

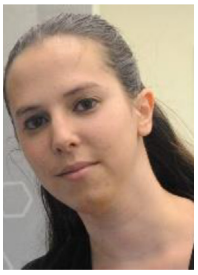
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge Matan Yesharim for his major contribution to the development of both the attack and the evaluation process. We also thank Boris Zadov for his professional expertise. Finally, we extend a special thank you to Mathov Designs for designing and building the real-world evaluation setup and for allowing us to publish its designs which will assist the research community worldwide.

References

[1] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, arXiv preprint arXiv:1312.6199 (2013).
[2] I.J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572 (2014).
[3] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z.B. Celik, A. Swami, The limitations of deep learning in adversarial settings, 2016 IEEE European symposium on security and privacy (EuroS&P), IEEE (2016) 372–387.
[4] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial examples in the physical world, arXiv preprint arXiv:1607.02533 (2016).
[5] M. Sharif, S. Bhagavatula, L. Bauer, M.K. Reiter, Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition, in: Proceedings of the 2016 acm sigsac conference on computer and communications security, 2016, pp. 1528–1540.
[6] I. Evtimov, K. Eykholt, E. Fernandes, T. Kohno, B. Li, A. Prakash, A. Rahmati, D. Song, Robust physical-world attacks on deep learning models, arXiv preprint arXiv:1707.08945 (2017).
[7] M. Lee, Z. Kolter, On physical adversarial patches for object detection, arXiv preprint arXiv:1906.11897 (2019).
[8] A. Athalye, L. Engstrom, A. Ilyas, K. Kwok, Synthesizing robust adversarial examples, in: International conference on machine learning, PMLR, 2018, pp. 284–293.
[9] Y. Mathov, Author repository: Enhancing real-world adversarial patches with 3d modeling techniques. https://github.com/yaliMa/Adversarial-Patch-3D, 2021. Accessed 15 July 2021.
[10] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, 2017 IEEE symposium on security and privacy (sp), IEEE (2017) 39–57.
[11] M. Sadeghi, E.G. Larsson, Physical adversarial attacks against end-to-end autoencoder communication systems, IEEE Commun. Lett. 23 (2019) 847–850.
[12] T.B. Brown, D. Mané, A. Roy, M. Abadi, J. Gilmer, Adversarial patch, arXiv preprint arXiv:1712.09665 (2017).
[13] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, Y. Chen, Dpatch: An adversarial patch attack on object detectors, arXiv preprint arXiv:1806.02299 (2018).
[14] J.-C. Su, M. Gadelha, R. Wang, S. Maji, A deeper look at 3d shape classifiers, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.
[15] C. Xiang, C.R. Qi, B. Li, Generating 3d adversarial point clouds, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9128–9136.
[16] C. Xiao, D. Yang, B. Li, J. Deng, M. Liu, Meshadv: Adversarial meshes for visual recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 6898–6907.
[17] X. Zeng, C. Liu, Y.-S. Wang, W. Qiu, L. Xie, Y.-W. Tai, C.-K. Tang, A.L. Yuille, Adversarial attacks beyond the image space, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4302–4311.
[18] L. Alton, What kind of computer do you need for 3d rendering?. https://www.computer.org/publications/tech-news/trends/what-kind-of-computer-do-you-need-for-3d-rendering, 2020. Accessed 22 February 2022.
[19] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, C.-J. Hsieh, Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models, in: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, 2017, pp. 15–26.
[20] J. Chen, M.I. Jordan, M.J. Wainwright, Hopskipjumpattack: A query-efficient decision-based attack, 2020 IEEE symposium on security and privacy (sp), IEEE (2020) 1277–1294.
[21] X. Xia, C. Xu, B. Nan, Inception-v3 for flower classification, in: 2017 2nd International Conference on Image, Vision and Computing (ICIVC), IEEE, 2017, pp. 783–787.
[22] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, 2009 IEEE conference on computer vision and pattern recognition, IEEE (2009) 248–255.
[23] B.O. Community, Blender – a 3D modelling and rendering package, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. Accessed 15 July 2021.
[24] SEED.EA, Pica pica – seed coffee mug. https://sketchfab.com/3d-models/pica-pica-seed-coffee-mug-b0f7d098678049749b1fc0fe4e881465, 2018. Accessed 15 July 2021.
[25] S. Dombi, Moderngl, high performance python bindings for opengl 3.3+, https://github.com/moderngl/moderngl, 2020. Accessed 15 July 2021.
[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., Tensorflow: A system for large-scale machine learning, in: 12th USENIX symposium on operating systems design and implementation (OSDI 16), 2016, pp. 265–283.



Yael Mathov is a data science researcher at Ben-Gurion University of the Negev (BGU) Cyber Security Research Center and a Ph.D. student of the Software and Information Systems Engineering department at BGU. She holds a B.Sc. degree in Computer Science and M.Sc. degree in Software and Information Systems Engineering (SISE) with a specialization in Cyber Space Security from BGU. Her primary research interests lie in the development and implementation of real-world adversarial learning techniques.

Lior Rokach is a data scientist and a professor of SISE BGU. His research interests lie in the design, development, and analysis of Machine Learning and Data Mining algorithms and their applications in Recommender Systems, Cyber Security and Medical Informatics. Rokach has co-founded four AI companies and has been awarded 22 patents for his inventions in AI and information technology. Prof. Rokach is the author of over 300 peer-reviewed papers in leading journals and conference proceedings. He is also the author of six books and the editor of three books, including two bestselling books: Recommender Systems Handbook (Springer, 1st edition, 2011; 2nd edition, 2015, 3rd edition, 2021) and The Data Mining and Knowledge Discovery Handbook (1st edition, Springer, 2005; 2nd edition, 2010).

Yuval Elovici is the director of the Telekom Innovation Laboratories at BGU, head of BGU Cyber Security Research Center, Professor in the Department of Software and Information Systems Engineering at BGU. He holds B.Sc. and M.Sc. degrees in Computer and Electrical Engineering from BGU and a Ph.D. in Information Systems from Tel-Aviv University. His primary research interests are computer and network security, cyber security, web intelligence, information warfare, social network analysis, and machine learning.