The Translucent Patch: A Physical and Universal Attack on Object Detectors

Alon Zolfi, Moshe Kravchik, Yuval Elovici, Asaf Shabtai
Department of Software and Information Systems Engineering
Ben-Gurion University of the Negev
{zolfi,moshekr}@post.bgu.ac.il, {elovici,shabtaia}@bgu.ac.il

Abstract

Physical adversarial attacks against object detectors have seen increasing success in recent years. However, these attacks require direct access to the object of interest in order to apply a physical patch. Furthermore, to hide multiple objects, an adversarial patch must be applied to each object. In this paper, we propose a contactless translucent physical patch containing a carefully constructed pattern, which is placed on the camera's lens, to fool state-of-the-art object detectors. The primary goal of our patch is to hide all instances of a selected target class. In addition, the optimization method used to construct the patch aims to ensure that the detection of other (untargeted) classes remains unharmed. Therefore, in our experiments, which are conducted on state-of-the-art object detection models used in autonomous driving, we study the effect of the patch on the detection of both the selected target class and the other classes. We show that our patch was able to prevent the detection of 42.27% of all stop sign instances while maintaining high (nearly 80%) detection of the other classes.

1. Introduction

In recent years, deep neural networks (DNNs) and particularly convolutional neural networks (CNNs) have become a state-of-the-art solution for computer vision tasks, such as image classification [13, 8], object detection [23, 24], and image segmentation [2, 1]. This is mainly due to DNNs' ability to accurately model complex multivariate data. However, such models' effectiveness depends heavily on their robustness to attacks that target the model itself; i.e., adversarial learning attacks.

Adversarial attacks have become a major focus of the machine learning research community, primarily in the computer vision domain [30, 25]. Recent studies have demonstrated the ability to implement evasion attacks (i.e., misclassification of an object) by applying a physical patch on the targeted object. Some examples include hiding a person using a small printed cardboard plate [33] or wearable clothes [34, 35, 10], concealing a stop sign by attaching small black and white stickers to it [6], or crafting a new stop sign with specific patterns on its background [3]. However, the main limitation of those physical attacks is that they require access to the object itself (to apply the adversarial patch). Furthermore, to hide multiple objects, an adversarial patch must be applied to each object.

Figure 1: Illustrating the physical translucent patch, placed on the camera lens, which results in the failure to detect the stop sign while correctly identifying the other objects.

We propose a novel physical attack aimed at fooling common object detection models by mounting a carefully crafted patch on the camera lens. Our attack (patch) is calculated by a gradient-based optimization process which results in a universal adversarial patch that takes the following goals into consideration: 1) successfully hide *all* instances of a target class from the object detector, 2) minimize the attack's impact on the detection of untargeted classes, and 3) produce a printable patch that is as unnoticeable as possible.

To the best of our knowledge, the only study that attempted to implement a camera-based physical attack is the work performed by Li *et al*. [15], in which an image classification model was targeted. Unlike an image classification model, which predicts the class of a single (dominant) object in the image, an object detection model is capable of *i)* detecting and classifying multiple objects regardless of their location and dominance within the image, and *ii)* processing thousands of candidate bounding boxes centered on each output pixel. Thus, in our attack all of the candidate

objects must be manipulated, significantly complicating the attack. Furthermore, in this research we study the effect of the physical patch on the untargeted classes, which was not addressed by Li *et al*.

To explore the feasibility of our approach, we first demonstrated the ability to deceive Tesla's advanced driving assistance system (ADAS) by applying two simple fully colored translucent patches on the camera's lens. We show that the ADAS misclassifies a stop sign (when a red-colored patch is used) and interprets a red traffic light as a green one (when using a cyan-colored patch). Then, we evaluated our proposed attack on CNN-based object detection models using datasets related to the autonomous car use case. Since autonomous cars operate in a real-world environment, we use the latest real-time object detection model, YOLOv5 [11], a recent improvement to YOLOv3 [23]. In our evaluation, we select the *stop sign* class as the targeted class, with the aim of preventing any of the instances from being detected by the object detector, both in the digital and physical domain. The results show that we are able to decrease the average precision of the detector for the stop sign class by 42.47% (digitally) and 42.27% (physically), while the detection of other object classes remains high.
We summarize the contributions of our work as follows:

- We present the first camera-based physical adversarial attack on object detection models.
- We craft a universal perturbation to fool the model for *all* instances of a specific object class, while maintaining the detection of untargeted objects.
- We demonstrate the transferability of the attack when the patch is generated using a surrogate model and then applied to a different model.
- The design and optimization process we propose take real-world constraints into account, including printing limitations and accurate patch placement, resulting in a practical attack.

2. Related Work

Previous works presenting adversarial attacks in the computer vision domain can be categorized by four main attributes: *i) model task:* image classification [13], object detection [31], or image segmentation [2]; *ii) attack type:* digitally generated perturbation or physical perturbations applied in the real-world (either by perturbing the physical object itself [6] or by perturbing the sensor's perception of the object [15]); *iii) attacker's knowledge:* full knowledge (white-box) [29], or no knowledge (black-box) about the model [21]; and *iv) perturbation type:* sparse noise around the entire image [32] or a centralized dense perturbation in a specific location [17].

As the main novel differentiator of our attack is its physical character, in the following review of related work we categorize the studies based on the *attack type*.

2.1. Digital Attacks

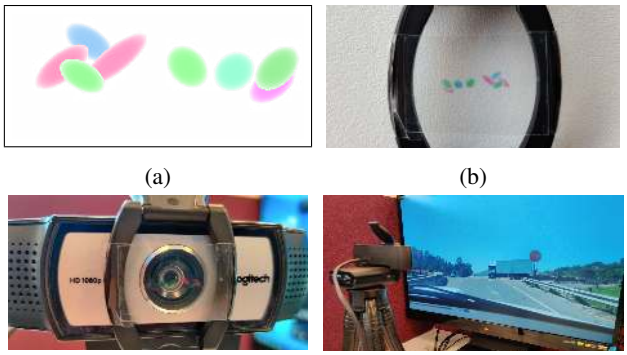
When attacks on DNNs in the computer vision domain were first introduced, they mainly targeted CNN-based classification models [32, 7]. These kinds of attacks have been extensively studied over the years in research proposing various ways to fool classifiers [29, 20]. Whereas these methods generate perturbations to fool DNNs on a single and specific image classes, Moosavi-Dezfooli *et al*. [19] proposed universal adversarial perturbations that can fool any image. Later, attacks against more complex computer vision tasks were shown. For example, Metzen *et al*. [9] demonstrated that imperceptible perturbations could also fool segmentation models. However, all of these studies digitally tampered with the input provided to the model. Since these attacks lacked significant real-world constraints, they do not naturally transfer to the physical world.

2.2. Physical Attacks

In recent years, physical attacks against image classification and object detection models have emerged. Kurakin *et al*. [14] took photos of printed adversarial images with a cell phone camera, successfully fooling a pretrained image classifier. Eykholt *et al*. [6] proposed a type of centralized physical perturbation (i.e., applying black and white stickers on stop signs) to fool image classifiers. Chen *et al*. [3] printed adversarial stop signs by adding specific background patterns, evading the Faster R-CNN [24] object detector, and Sitawarin *et al*. [27] crafted toxic signs, similar to original traffic signs, to deceive autonomous car systems. To avoid facial recognition systems, Sharif *et al*. [26] suggested wearing adversarial eyeglass frames. This work also introduced the *non-printability score*, which forces the adversarial perturbations to use printable colors.

Recently, attacks against person detectors have emerged. First, Thys *et al*. [33] printed an adversarial patch on a small cardboard plate, successfully evading YOLOv2 [22] detection. Following this work, other studies presented attacks in which adversarial patterns were printed on t-shirts [34, 35, 10]. While these methods create less realistic distortions, Duan *et al*. [4] used natural patterns that appear legitimate to human observers and applied them on different objects to fool image classifiers.

Unlike the studies mentioned above, in which the perturbation was applied on the targeted object, in our work, we place the perturbation on the sensor collecting the image stream. While Li *et al*. [15] also applied a physical adversarial perturbation on the camera's lens, their study focused on image classifiers, differing from our work, in which we attack object detectors, a more complex mechanism. In addition, we examine the effect of our attack on target class while not reducing the detection of non-targeted classes, which was not an aspect addressed by Li *et al*.



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

Figure 2: An illustration of: (a) a digital patch with eight shapes; (b) a patch printed on transparent paper; (c) a physical patch applied to the camera's lens; and (d) lab setup of a camera and a screen.

3. Method

In this research, we aim to generate a printable, translucent universal adversarial perturbation (UAP) that can be used to hide all instances of the selected target class by applying the UAP to the camera's lens. As also noted by Li *et al*. [15], accurately attaching pixel-level perturbations onto the camera's lens is impossible and not practical for our case. Therefore, we design and craft a *region-level perturbation*, containing several oval shapes printed on transparent paper, to create a practical attack. By optimizing a custom *loss function*, which considers the main attack goals (i.e., hiding the selected target class instances while maintaining the detection level of the other classes), we generate the region-level perturbation. We assume that the attacker has direct access to the camera's lens to apply the patch (e.g., via a supply chain).

In this section, we start by describing the design requirements for the patch (Section 3.1). Then, we present the process of optimizing the parameters of the patch in order to achieve a successful attack (Section 3.2).

3.1. Patch Design

From digital to physical setup. Our adversarial perturbation is computed digitally and then applied as a real physical patch. Therefore, when computing the digital patch, we need to consider the effect of printing and applying a translucent physical patch on the camera's lens. As Li *et al*. [15] suggested, an approximation of that effect can be achieved using an alpha blending process between the original image and the digital patch. Therefore, in this study, we define our patch as a 2D image with four channels. The first three channels represent RGB colors supplemented with a fourth alpha channel representing how opaque each pixel is. The result of performing alpha blending between the original image and a translucent patch at pixel (i, j) is defined as follows:

$$\begin{aligned}\text{perturbed}(i,j) = \text{original}(i,j) * (1 - \alpha(i,j)) + \\ \gamma(i,j) * \alpha(i,j) \quad (1)\end{aligned}$$

where perturbed is the resulting image consisting of just RGB channels, α represents the patch's alpha channel, and γ represents RGB triplets.

Patch structure. As illustrated in Figure 2a, our patch comprises n (configurable parameter) blurry oval shapes; these shapes are initialized as fully colored dots with shearing in the x and y directions. The following attributes define a single shape:

- $(x_c, y_c) \in [-1, 1] \subset \mathbb{R}^2$ - a tuple representing a shape's center with regard to the center of the image; for example, a shape centered at $(0, 0)$ will be placed in the center of the image.
- $r \in [r_{\min}, r_{\max}] \subset \mathbb{R}$ - the shape's radius in relation to the patch size.
- $(sh_x, sh_y) \in [-1, 1] \subset \mathbb{R}^2$ - a tuple representing the shape's shear in the x and y directions.
- $\gamma \in \mathbb{R}^3$ - a triplet representing an RGB color.
- $\alpha \in [0, 1] \subset \mathbb{R}$ - represents the alpha value.

Patch blending. Usually, when crafting a pixel-level patch, the *total variance* factor [26] is also included in the custom loss function to ensure that the optimizer favors smooth color transitions between adjacent pixels, thus preventing noisy images. Since we aim to generate a region-level patch, the configuration of the patch's alpha channel is implemented as a predefined total variation constraint, forcing neighboring pixels within the same shape to be similar. We achieve this by defining a position-dependent alpha channel at pixel (i, j) for a single shape as follows:

$$\begin{aligned}\alpha(i,j) &= a_{\max} * (-s * d(i,j)^{\beta} + 1) \\ d(i,j) &= \frac{(i - x_{c,\text{norm}})^2 + (j - y_{c,\text{norm}})^2}{r_{\text{norm}}^2} \\ x_{c,\text{norm}} &= (1 - x_c) * \lfloor \frac{p_w}{2} \rfloor \\ y_{c,\text{norm}} &= (1 - y_c) * \lfloor \frac{p_h}{2} \rfloor \\ r_{\text{norm}} &= r * \lfloor \frac{\min(p_w, p_h)}{2} \rfloor\end{aligned} \quad (2)$$

where the parameters above represent the following:

- $a_{\max} \in [0, 1] \subset \mathbb{R}$ - the maximum value of the alpha channel.
- $s \in [0, 1] \subset \mathbb{R}$ - controls the minimum value of the alpha channel, where $\alpha_{\min} = \alpha_{\max} * (1 - s)$.
- $\beta \in \mathbb{R}_+$ - exponential drop-off.

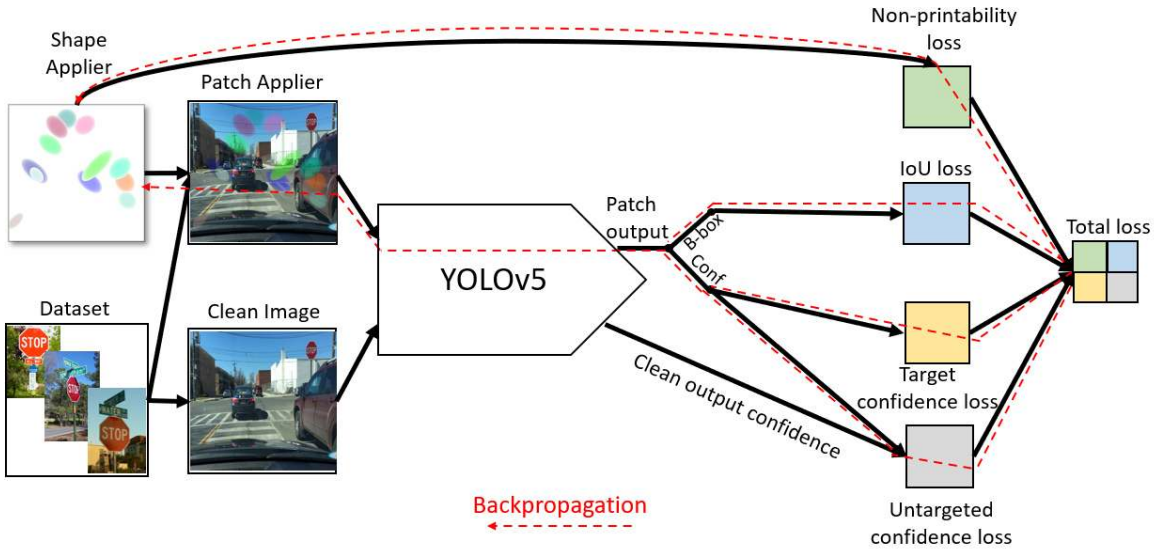


Figure 3: Overview of our method's pipeline.

- $(p_w, p_h) \in \mathbb{N}^2$ - patch width and height.
- $d(i,j) \in [0,1] \subset \mathbb{R}$ - the normalized distance of pixel (i,j) from the shape's normalized center $(x_{c,\text{norm}}, y_{c,\text{norm}})$.

Equation 2 produces distance-dependent opacity along the shapes' area, forming a smooth shape. Pixels close to the shape's center ($d \to 0$) will have an alpha value closer to $\alpha_{\max}$, whereas pixels that lie close to the shape's edge ($d \to 1$) will have near $\alpha_{\min}$ value. The drop-off parameter β controls the smoothness intensity.

Shape's positioning and shearing. To control the positioning and shearing of each shape on the patch, we use 2D affine transformations [28], characterized by the following affine matrix:

$$\begin{bmatrix} 1 & sh_x & x_c \\ sh_y & 1 & y_c \end{bmatrix} \tag{3}$$

Intuitively, the shape's location is characterized by discrete pixel coordinates. However, since we use a gradient-based optimization process, discrete parameters cannot be used. Therefore, we use affine transformation to represent a continuous and differentiable form of the shape's location.

Attack parameters. The parameters discussed in this section can be divided into two groups:

- a set of manually chosen (input) parameters: number of shapes n and Θ_{manual} which characterizes all of the shapes on our patch:

$$\Theta_{\text{manual}} = (\alpha_{\max}, s, \beta, (r_{\min}, r_{\max})) \tag{4}$$

  The specific values chosen for Θ_{manual} are essential for limiting the amount of noise applied to our patch and simulating a printed physical patch, which will be further explained in Section 5.

- The free parameters Θ_{free} are optimized by the proposed algorithm:

$$\begin{gathered} \theta = ((x_c, y_c), r, (sh_x, sh_y), \gamma) \\ \Theta_{\text{free}} = \{\theta_1, .., \theta_n\} \end{gathered} \tag{5}$$

  where θ characterizes a single shape and Θ_{free} is a composition of all of the shapes parameters.

3.2. Patch Optimization

To optimize Θ_{free}, we compose a custom loss function consisting of four components:

$$\begin{gathered} \ell_{\text{total}} = w_1 * \ell_{\text{target conf}} + w_2 * \ell_{\text{IoU}} \\ + w_3 * \ell_{\text{untargeted conf}} + w_4 * \ell_{\text{nps}} \\ \sum_i w_i = 1 \end{gathered} \tag{6}$$

The optimization process minimizes ℓ_{total}, aiming to achieve three main goals, each of which will be discussed throughout this section, noting their relation to the components presented in Equation 6. To determine the optimal weight values w_i, we use the grid search approach. We also allow zeroing w_1 and w_2 during the hyperparameter search to study whether they are both necessary to achieve our goals.

Since our entire attack is differentiable, we use an automatic differentiation tool kit (PyTorch) to optimize our patch parameters using the backpropagation algorithm, as shown in Figure 3.

Eliminating the detection of the target class. The inference output of object detection models provides two unique defining details for each detected object: the bounding box and confidence score. We use these details to achieve the following goals:

- Minimize the confidence score of the target class:

$$\ell_{\text{target conf}} = Pr(\text{objectness}) * Pr(\text{target class}) \quad (7)$$

  where $Pr(\text{objectness})$ and $Pr(\text{class})$ correspond to the YOLO output, which consists of two confidence scores for each cell in the final detection grid: (a) the objectness score - whether a specific cell in the grid contains any object, (b) the class score - whether a specific cell in the grid contains a specific class. The model outputs a detection only if the confidence score surpasses a certain threshold. As noted by Thys *et al.* [33], it is possible to minimize $Pr(\text{objectness})$ and $Pr(\text{class})$ individually. However, in our preliminary experiments, the only combination that led to adequate results was the product of these components.

- Minimize the Intersection over Union (IoU) score between the predicted bounding box and the ground truth bounding box of the target class:

$$\ell_{\text{IoU}} = IoU^{\text{ground truth}}_{\text{predicted}}(\text{target class}) \quad (8)$$

  By doing this, we steer the shapes in our patch toward learning better positions, causing the detector to incorrectly predict the location of bounding boxes and resulting in the misdetection of the object's location. Although our attack aims to hide a selected target class, incorrectly placing a bounding box can also negatively affect the detector's performance. When the location of the predicted bounding box matches that of the ground truth bounding box with high accuracy, the penalty of this component is greater.

Maintaining the detection of untargeted classes. Since our patch is not placed on the target object, we address the issue of affecting the detection of the untargeted classes:

$$\ell_{\text{untargeted conf}} = \frac{1}{M} \sum_{\substack{\text{cls} \in \text{image} \\ \text{cls} \neq \text{target}}} |\text{conf(cls, clean)} - \text{conf(cls, patch)}| \quad (9)$$

where conf(cls,image) represents the confidence score of class *cls* in image *image*, as explained earlier in this section, and M represents the number of classes in the image.

When untargeted classes are detected correctly, the loss will be closer to zero. On the other hand, when the detector does not detect untargeted classes, the penalty of this component is greater.

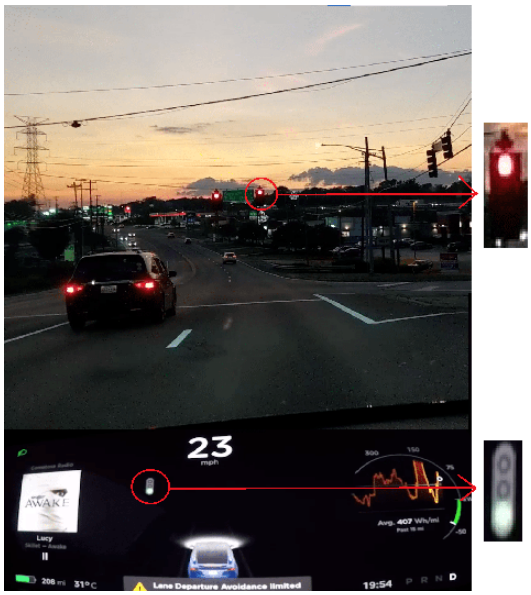


Figure 4: Camera-based attack on a Tesla Model X, using a cyan-colored patch. The street view in the upper part of the image shows the actual scene with a red traffic light. The lower part of the image shows that the car's navigation screen interprets the traffic light's color as green.

Generating a printable patch. We include the non-printability score (NPS) [26], which represents how closely digital colors match colors printed by a standard printer:

$$\ell_{nps} = \sum_{c_{\text{patch}} \in P} \min_{c_{\text{print}} \in C} |c_{\text{patch}} - c_{\text{print}}| \quad (10)$$

where c_{patch} is a color in our patch P (i.e., shape color) and c_{print} is a color in the set of printable colors C. This loss penalizes colors that are far from the set of printable colors.

4. Motivating Example: Tesla Use Case

As a proof of concept, we demonstrated the ability to successfully perform a camera-based attack on an object detection model by applying a physical on-sensor patch. After applying fully colored patches (i.e., not generated by our attack) on the front camera of a Tesla Model X, we were able to deceive the car's advanced driving assistance system (ADAS):

- Traffic light attack - by applying a cyan-colored patch, we managed to change the ADAS's perception of a traffic light's color, so that it interpreted a red light as a green light, as shown in Figure 4.
- Stop sign attack - by applying a red-colored patch, we were able to hide a stop sign and prevent it from being detected.

A demo of both experiments can be found here: https://youtu.be/n6P55eslyvA.

Figure 5: Examples of frames presented to the detection model and its detections when various patches are applied to the camera's lens. Each row represents a single frame, and each column represents the use of a different patch.

5. Evaluation

We evaluate our attack on the use case of autonomous cars, in which object detection models are used to identify obstacles and road signs. Specifically, we aim to devise an attack that will cause the object detection model to fail to detect stop signs while maintaining its capability of detecting other object classes.

We first experiment in the digital domain by using alpha blending between the original images and several different types of patches (see Equation 1). Then, we evaluate the performance of the patches in a real-world setup by printing and attaching them to a camera lens, as shown in Figure 2.

Models. We evaluated our attack using the YOLOv5 one-stage detector [11] (an improvement to YOLOv3 [23]) in a white-box setting. In addition, we examined our patch's transferability to other detectors, YOLOv2 [22] and Faster R-CNN [24], as a black-box setting. For all of the detectors we use pretrained weights on the MS-COCO [16] dataset. MS-COCO contains 80 object categories from several domains. We selected eight relevant categories: person, bicycle, car, bus, truck, traffic light, fire hydrant, and stop sign.

Datasets. We use a combination of multiple driving-related datasets, from which we extracted images containing stop signs, to improve the robustness of our patch:

- LISA traffic sign dataset [18] - tens of videos split into frames containing U.S. traffic signs; ∼500 images containing stop signs.
- Mapillary Traffic Sign Dataset (MTSD) [5] - a diverse street-level dataset obtained from a rich geographic area; ∼750 images containing stop signs.
- Berkeley DeepDrive (BDD) [36] - videos of the driving experience covering many different times of the day, weather conditions, and driving scenarios; ∼500 images containing stop signs.

Hence, the full dataset contains ∼1750 images of stop signs. Since the LISA and MTSD datasets only have annotations of traffic signs, we use our object detection model (YOLOv5) to annotate (label) the rest of the classes examined in this paper. These annotations are later used as the ground truth.

Evaluation setup. We split our full dataset into training, validation, and test sets. The training and validation sets consist of images from the BDD and MTSD datasets (randomly chosen with a split ratio of 90% and 10%, respectively), while the test set contains images from the LISA dataset. By dividing the full dataset this way, we achieve two goals: (1) there is no correlation between the training/validation sets and the test set; thus, achieving good results on the test set implies that our patch is unbiased and universal, and (2) since the LISA dataset is comprised of multiple videos split into frames, it allows us to demonstrate our patch's effectiveness in the physical domain by testing it on driving videos.

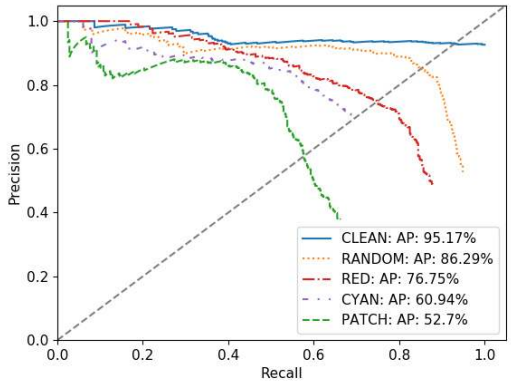
Evaluation metrics. We use the average precision (AP) for digital attacks, representing the area under the precision-recall (PR) curve for both the selected target class and untargeted classes.

In addition, to quantify our physical patch's success in the real world, we define the following metric:

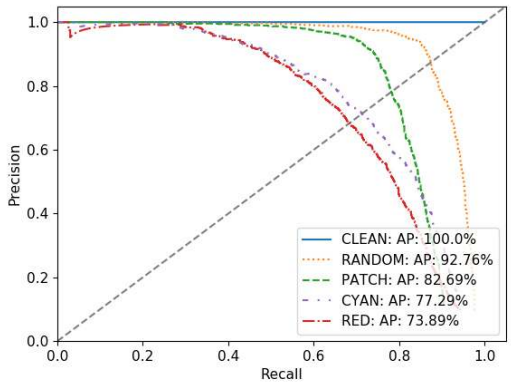
$$\text{Fooling Rate}(\text{class}) = \frac{\#\text{ fooled objects}(\text{class})}{\#\text{ total objects}(\text{class})} \qquad (11)$$

where an object of category *class* is considered 'fooled' if it was not output by the detector (the confidence score does not surpass the minimum threshold). We set our minimum confidence score threshold at 0.4, a point where our detector achieved an AP of 95.17% on the original images containing the stop sign class in the digital domain.

Types of patches evaluated. We compare our patch's performance to several different translucent patches: (a)



(a) Stop sign class.



(b) Other (untargeted) classes.

Figure 6: Precision-Recall (PR) curves of our method (PATCH) with eight shapes compared to different patch types (RANDOM, RED, CYAN) and the original images (CLEAN) on the YOLOv5 (white-box setting).

CLEAN - the original image without a patch, (b) PATCH - our optimized patch, (c) RANDOM - a baseline patch with the same number of shapes as PATCH and random initialization of our attack's optimized parameters Θ_{free}: location, color, and shearing, (d) RED - a fully red-colored patch, (e) CYAN - a fully cyan-colored patch. The different types of patches evaluated are presented in Figure 5.

Implementation details. Throughout this section the results presented are obtained using the following manual parameters: 1) $s = 0.9$, to achieve near zero values at the shape's edge, for smooth transitions between pixels inside and outside the shape's area; 2) $\beta = 2.5$, to apply high intensity around the shape's center, which will be caused in the physical patch by the printer; and 3) the upper radius bound $r_{\max} = 0.25$, to bind the number of pixels changed in the perturbed image, while the lower bound $r_{\min} = 0.03$ is the minimum printable shape.

The settings of our free parameters Θ_{free} are randomly initialized and updated using the Adam optimizer [12]. The initial learning rate is set at $5e^{-3}$, except for the radius update, which is initially set at $8e^{-4}$. Moreover, using a grid search algorithm, we found the optimal w_i setting to be $w_1 = 0.74\,, w_2 = 0.15\,, w_3 = 0.1\,, w_4 = 0.01$.

5.1. Digital Attacks

To evaluate our patch's effectiveness, we conduct digital experiments in which our optimized patch is applied to images in the test set, using alpha blending (see Equation 1).

White-box attack performance. First, we examine our attack's performance in a *white-box* setting in which our patch is optimized using the weights of the YOLOv5 detector and evaluated on the same model. As shown in Figure 6a, using a patch with eight shapes with $\alpha_{\max} = 0.4$, we were able to degrade the detector's AP to 52.7%, which is a decrease of over 42.47% from the detector's best performance (which is 95.17%). In addition, of the patches examined, our patch has the greatest effect on the detector's performance for the *stop sign* class.

Since we do not apply our patch on a specific object but rather on the entire image, it might interfere with the detection of untargeted classes; thus, a comprehensive evaluation must address that issue as well. The results in Figure 6b show that our patch has a minimal effect on untargeted classes. The detector was able to achieve an AP of 82.69%, for the untargeted classes which is only a 10.07% difference from the random patch and better than simple fully-colored patches. It should be noted that our detector's AP on untargeted classes is 100%, because the ground truth labels on the LISA dataset were generated by our detector, as mentioned earlier in this section. Unlike our patch, which affects the detection of the stop sign class more than other classes, the rest of the attacks have a similar effect on all of the classes, both targeted and untargeted.

Attack's performance for different parameter values. A significant aspect of adversarial attacks refers to the amount of noise added to the resulting image, which is mainly controlled by manual parameter setting. The noise can have two kinds of impact, affecting the deception rate as follows:

- The number of pixels changed - in our attack this is affected by two parameters: the number of shapes n and the shapes' radius r. Since r is a free parameter, we examine the effect of the number of shapes with $\alpha_{\max} = 0.4$. As shown in Table 1, we can see that the AP decreases as we

# Shapes (n)	'Stop Sign' Class	Other Classes
3	91.15%	95.44%
5	77.45%	91.72%
7	65.01%	83.23%
10	53.11%	77.65%
15	47.9%	72.49%

Table 1: Average precision as a function of n

add more shapes. It should be noted that at some point, the AP does not decrease as we add more shapes, because shapes may overlay each other.

- The patch's dominance in the resulting image - in our attack the dominance is mainly influenced by $\alpha_{\max}$; therefore, we use a fixed number of shapes (eight) to study $\alpha_{\max}$'s influence. As shown in Table 2, we are able to decrease the detector's AP as the patch's $\alpha_{\max}$ increases in the resulting image, eliminating a large portion of stop sign instances when setting the maximum opacity of the patch to 90%. While high opacity achieves a higher fooling rate, at some level the patch becomes perceptible to the human eye. Thus, the setting of $\alpha_{\max}$ must address the patch's effectiveness on the target class and keep the patch as imperceptible as possible.

Moreover, adding too many shapes or setting the value of $\alpha_{\max}$ too high largely affects the detection of the untargeted classes, which is something that we want to avoid.

$\alpha_{\max}$	'Stop Sign' Class	Other Classes
0.1	93.85%	98.26%
0.3	70.13%	88.25%
0.5	51.75%	81.93%
0.7	38.61%	78.76%
0.9	36.55%	70.45%

Table 2: Average precision as a function of the α_{max}

Transferability of the attack. To perform a comprehensive evaluation of our patch, we further investigate its performance on detectors that it was not trained on (black-box setting), which means that our patch (eight shapes and $\alpha_{\max} = 0.4$) was optimized using a surrogate model and tested on other models. In Table 3, we present our patch's performance on two attacked models: YOLOv2 [22] and Faster R-CNN [24]. We show that our patch can successfully deceive models it was not trained on, reducing the AP of the YOLOv2 and Faster R-CNN on the stop sign class to 57.36% and 54.53% respectively, while maintaining an AP for untargeted classes that is close to the detector's best performance for this task.

5.2. Physical attacks

Finally, to evaluate our patch's performance in the real world, we print it on transparent paper (Figure 2b) and place it on the camera's lens (Figure 2c), filming a computer screen that is projecting videos of our test set (Figure 2d). Since the physical attack evaluation requires access to the object detection model's predictions, we used a testbed which contained the following equipment: a Logitech C930 web camera to simulate the autonomous car's camera, a 21-inch computer screen to project videos on, simulating real driving, and YOLOv5 to simulate the car's ADAS. To print our patch (0.6x0.33 inches) on transparent paper, we used a Xerox 6605DN laser printer.

	'Stop Sign' Class		Other Classes	
Model/Attack	CLEAN	PATCH	CLEAN	PATCH
YOLOv5	95.17%	52.7%	100%	82.69%
YOLOv2	81.54%	57.36%	59.13%	54.92%
Faster R-CNN	94.31%	54.53%	78.31%	70.36%

Table 3: Average precision in black-box setting: patch trained on YOLOv5 and evaluated on other object detectors

We first evaluate the detector's best performance on the test set videos without any attack (i.e., no patch is applied to the camera's lens) and then use this as a reference (ground-truth) to examine the effect of different patches. As the results presented in Table 4 show, our physical patch was able to cause the detection model to fail to detect 42.27% of the total number of stop sign instances (compared to 42.47% in the digital attack), while still detecting nearly 80% of the untargeted class instances (compared to 82.69% in the digital attack). In contrast to the results obtained by our patch, the other patches have major disadvantages: 1) the random patch could not achieve an adequate fooling rate on the stop sign class, and 2) the fully colored patches performed poorly on the untargeted classes (similar to the effect of completely blocking the camera's lens).

During the physical experiments we observed a specific trend when our patch was used - in most of the scenes, the stop sign is detected at a very late stage, which leads to a very small window of time for the ADAS to respond.

Class/Attack	PATCH	RANDOM	RED	CYAN
Stop sign	42.27%	20.57%	93.3%	98.9%
Others	21.54%	19.27%	82.7%	81.6%

Table 4: Fooling rate for the stop sign class and other classes for physical patch attacks

6. Conclusion

We presented a physically-realizable attack against state-of-the-art object detectors without the need to have direct access to the targeted object. We crafted a translucent patch attached to the camera lens by the adversary with a unique design that considers real-world constraints and implemented a custom optimization process to achieve a successful attack in a real-world environment. Our experiments demonstrated that it is possible to prevent a specific class from being detected while simultaneously allowing the detection of other classes. Moreover, we showed that compared to the other evaluated attacks, our method has the best trade-off between the target class's misdetection and the untargeted classes' detection. Our study highlighted the risk that autonomous cars' ADASs face from an adversary capable of simply applying a patch on their cameras. Further research should focus on proposing countermeasures for such attacks, such as anomaly detection or active sensor checks.

References

[1] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4974–4983, 2019. 1

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 1, 2

[3] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng Polo Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 52–68. Springer, 2018. 1, 2

[4] Ranjie Duan, Xingjun Ma, Yisen Wang, James Bailey, A Kai Qin, and Yun Yang. Adversarial camouflage: Hiding physical-world attacks with natural styles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1000–1008, 2020. 2

[5] Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, Gerhard Neuhold, and Yubin Kuang. The mapillary traffic sign dataset for detection and classification on a global scale. *arXiv preprint arXiv:1909.04422*, 2019. 6

[6] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1625–1634, 2018. 1, 2

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015. 1

[9] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. Universal adversarial perturbations against semantic image segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2755–2764, 2017. 2

[10] Lifeng Huang, Chengying Gao, Yuyin Zhou, Cihang Xie, Alan L Yuille, Changqing Zou, and Ning Liu. Universal physical camouflage attacks on object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 720–729, 2020. 1, 2

[11] Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, Adam Hogan, lorenzomammana, tkianai, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Hatovix, Jake Poznanski, Lijun Yu, changyu98, Prashant Rai, Russ Ferriday, Trevor Sullivan, Wang Xinyu, YuriRibeiro, Eduard Reñé Claramunt, hopesala, pritul dave, and yzchen. ultralytics/yolov5: v3.0, Aug. 2020. 2, 6

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 2

[14] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 2

[15] Juncheng Li, Frank R Schmidt, and J Zico Kolter. Adversarial camera stickers: A physical camera-based attack on deep learning systems. *arXiv preprint arXiv:1904.00759*, 2019. 1, 2, 3

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[17] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018. 2

[18] Andreas Mogelmose, Mohan Manubhai Trivedi, and Thomas B Moeslund. Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1484–1497, 2012. 6

[19] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1765–1773, 2017. 2

[20] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016. 2

[21] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks. *arXiv preprint arXiv:1612.06299*, 2016. 2

[22] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2, 6, 8

[23] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 1, 2, 6

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2, 6, 8

[25] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018. 1

[26] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael 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*, pages 1528–1540, 2016. 2, 3, 5

[27] Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. Darts: Deceiving autonomous cars with toxic signs. *arXiv preprint arXiv:1802.06430*, 2018. 2

[28] Charles C Stearns and Karthikeyan Kannappan. Method for 2-d affine transformation of images, Dec. 12 1995. US Patent 5,475,803. 4

[29] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019. 2

[30] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1299–1316, 2018. 1

[31] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013. 2

[32] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 2

[33] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: adversarial patches to attack person detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 1, 2, 5

[34] Zuxuan Wu, Ser-Nam Lim, Larry Davis, and Tom Goldstein. Making an invisibility cloak: Real world adversarial attacks on object detectors. *arXiv preprint arXiv:1910.14667*, 2019. 1, 2

[35] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. *arXiv*, pages arXiv–1910, 2019. 1, 2

[36] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2636–2645, 2020. 6