DARTS: Deceiving Autonomous Cars with Toxic Signs

CHAWIN SITAWARIN, Princeton University, USA
ARJUN NITIN BHAGOJI, Princeton University, USA
ARSALAN MOSENIA, Princeton University, USA
MUNG CHIANG, Purdue University, USA
PRATEEK MITTAL, Princeton University, USA

Sign recognition is an integral part of autonomous cars. Any misclassification of traffic signs can potentially lead to a multitude of disastrous consequences, ranging from a life-threatening accident to even a large-scale interruption of transportation services relying on autonomous cars. In this paper, we propose and examine *realistic security attacks* against sign recognition systems for **D**eceiving **A**utonomous ca**R**s with **T**oxic **S**igns (we call the proposed attacks ***DARTS***).

Leveraging the concept of *adversarial examples*, we strategically modify innocuous signs/advertisements in the environment in such a way that they seem normal to human observers but are interpreted as the adversary's desired traffic sign by autonomous cars. Further, we pursue a fundamentally different perspective to attacking autonomous cars, motivated by the observation that the driver and vehicle-mounted camera see the environment from different angles (the camera commonly sees the road with a higher angle, e.g., from top of the car). Bridging concepts from optics (in particular, *lenticular printing*), security, and computer vision, we propose a novel attack against vehicular sign recognition systems: we create signs that change as they are viewed from different angles, and thus, can be interpreted differently by the driver and sign recognition.

We extensively evaluate the proposed attacks under various conditions: different distances, lighting conditions, and camera angles. We first examine our attacks *virtually*, i.e., we check if the digital images of toxic signs can deceive the sign recognition system. Further, we investigate the effectiveness of attacks in real-world settings: we print toxic signs, install them in the environment, capture videos using a vehicle-mounted camera, and process them using our sign recognition pipeline. We find our attacks to achieve attack success rates of over 90% in both the digital and real-world settings. We further suggest a countermeasure based on adversarial training to protect against adversarial example-based attacks.

Our proof-of-concept attacks shed light on a fundamental security challenge associated with the use of sign recognition techniques in autonomous cars, paving the way for further investigation of overlooked security challenges of autonomous cars.

Additional Key Words and Phrases: Adversarial examples, Autonomous cars, Autonomous vehicles, Lenticular printing, Security, Sign recognition, Toxic signs, Traffic signs

ACM Reference Format:
Chawin Sitawarin, Arjun Nitin Bhagoji, Arsalan Mosenia, Mung Chiang, and Prateek Mittal. 2018. DARTS: Deceiving Autonomous Cars with Toxic Signs. 1, 1 (February 2018), 27 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: Chawin Sitawarin, Princeton University, Department of Electrical Engineering, Princeton University, Princeton, NJ, 08540, USA; Arjun Nitin Bhagoji, Princeton University, Department of Electrical Engineering, Princeton University, Princeton, NJ, 08540, USA; Arsalan Mosenia, Princeton University, Department of Electrical Engineering, Princeton University, Princeton, NJ, 08540, USA; Mung Chiang, Purdue University, Department of Electrical and Computer Engineering, Purdue University, Princeton, NJ, 08540, USA; Prateek Mittal, Princeton University, Department of Electrical Engineering, Princeton University, Princeton, NJ, 08540, USA.

XXXX-XXXX/2018/2-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

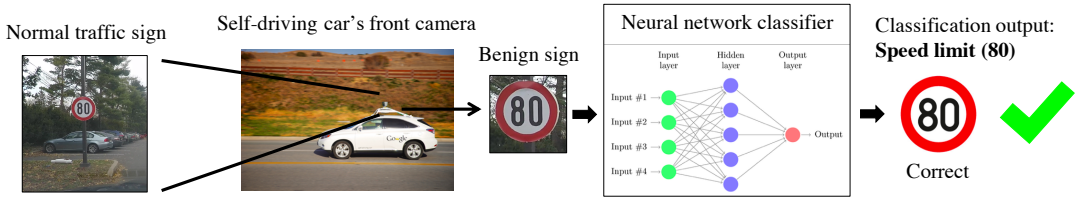
1 INTRODUCTION

The rapid technological and scientific advancements in artificial intelligence (AI) and machine learning (ML) have led to their deployment in ubiquitous, pervasive systems and applications, such as authentication systems [1, 2], healthcare applications [3, 4], and several vehicular services [5–8]. The ubiquity of machine learning provides adversaries with both opportunities and incentives to develop strategic approaches to fool learning systems and achieve their malicious goals [9, 10]. A number of powerful attacks on the test phase of ML systems used for classification have been developed over the past few years, including attacks on Support Vector Machines [11, 12] and deep neural networks [13–15]. These attacks work by adding carefully crafted perturbations to benign examples to generate adversarial examples. In the case of image data, these perturbations are typically imperceptible to humans. While these attacks are interesting from a theoretical perspective and expose gaps in our understanding of the working of neural networks, their practical importance remains unclear. The main question to be addressed is "What is the nature and extent of the attacks that can be carried out on real-world ML systems?'.

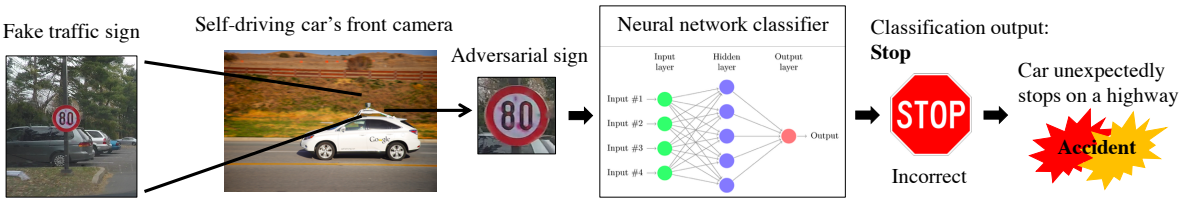
Real-world ML systems can broadly be categorized into two categories, cloud-based ML as a Service (MLaaS) systems [16–19] and systems which interact with the physical-world, such as face recognition for authentication and surveillance [20], and computer vision subsystems of self-driving cars [21, 22]. MLaaS providers often make their models available to clients [23], so attackers who can generate adversarial examples for these models represent a threat to a large number of possible targets. MLaaS providers typically provide query-based access to their models for users, a fact which has been leveraged to develop powerful query-based attacks on these systems [24–27]. These attacks are all carried out in a virtual setting and are not affected by the noise introduced when sensing and capturing the physical world. A limited number of attacks on ML classifiers which interact with the physical world have been proposed [28, 29]. Kurakin et al. [28] print out virtual adversarial examples and take pictures of these, which are then passed through the original classifier. Sharif et al. [29] attack face recognition systems by allowing subjects to wear printed glasses with the adversarial perturbations necessary for the faces to be misclassified. However, both these attacks were carried out in a very controlled laboratory setting, making their effectiveness unclear in a real-world scenario. Further, they rely entirely on the presence of adversarial examples to attack the underlying systems. In this paper, we expand the scope of attacks on real-world ML systems by demonstrating physically realizable and robust attacks on the computer vision systems of autonomous cars, which are widely used for *traffic sign recognition* among other uses.

We focus on autonomous cars since they are arguably the most important upcoming application of ML [5, 6]. A number of corporations have forecast the presence of autonomous cars on our roads in the next decade [30] and the size of the future global market has been estimated to be as high as seven trillion dollars [31]. Since the computer vision systems of current and future autonomous cars are likely to be based on neural networks [21, 22], the existence of physical world attacks on neural networks would represent a significant threat. However, as mentioned earlier, the attacks on virtual systems listed above do not translate directly to the real world. This occurs because the methods used generate virtual adversarial examples do not account for varying physical conditions which may include brightness variation, camera artifacts, shadow and reflection, orientations and distances and the loss of detail from image re-sizing. A technical report by Evtimov et al. [32] has performed a preliminary investigation of these issues. A detailed comparison with our work is in Section 6. Another issue that has not been investigated in this setting before is that the adversary may not have access to the internal details of the computer vision system used, and will thus have to carry out attacks in a *black-box* setting.

We overcome these hurdles and propose new attacks targeting the *traffic sign recognition* modules of autonomous cars. The main thrust of the attack revolves around creating a fake sign that has one semantic meaning to humans but is classified as a different, and in some situations, dangerous, traffic sign by autonomous cars. The misclassification of traffic signs by autonomous cars can lead to serious real-world consequences including road



(a) Operation of the computer vision subsystem of an AV under *benign conditions*



(b) Operation of the computer vision subsystem of an AV under *adversarial conditions*

Fig. 1. **Difference in operation of autonomous cars under benign and adversarial conditions**. Figure 1b shows the classification result for a drive-by test for a physically robust adversarial example generated using our Adversarial Traffic Sign attack.

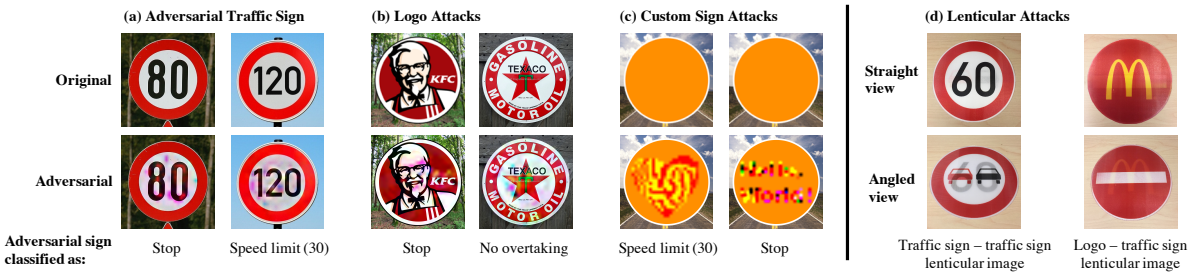


Fig. 2. **Toxic signs for our traffic sign recognition pipeline generated using the Adversarial Traffic Sign attack and the Sign Embedding attacks (2(a)-(c)) (Logo and Custom Sign) as well as the Lenticular Printing attack (2(d))**. The adversarial examples are *classified as the desired target traffic sign with high confidence* under a variety of physical conditions when printed out. The Lenticular Printing attack samples flip the displayed sign depending on the viewing angle, simulating the view of a human and the camera in a AV.

accidents and widespread traffic confusion. For instance, a fake speed limit sign, created by adding perturbations barely visible to humans to an actual speed limit sign could be classified as a stop sign by an AV, causing the car to slow down on a busy highway when it is not meant to. This is illustrated in Figure 1.

To demonstrate our attacks, we generate physically robust adversarial examples from *innocuous signs* as well as from existing road signs as shown in Figure 2(a)-(c). We also go beyond the phenomenon of adversarial examples by demonstrating attacks based on Lenticular Printing, shown in Figure 2(d), which can operate in a purely black-box setting. In these attacks, the sign that is visible changes depending on the viewing angle, so a sign appearing as a innocuous logo to a human could appear as a potentially dangerous traffic sign to the camera on the AV. Thus, our DARTS attack comprises three separate attacks, namely Sign Embedding, Adversarial Traffic

Sign and Lenticular Printing. We evaluate the real-world viability of the adversarial examples by setting up a realistic evaluation pipeline shown in Figure 3 which we test in both white-box and black-box settings. We also propose a possible defense against the attacks we demonstrate based on the concept of adversarial training [14]. All the code and data required to reproduce our results are available at https://github.com/anonymous [1]. *Videos of successful drive-by tests* with our adversarial examples are provided as supplementary material.

In summary, the contributions of this paper are as follows:

(1) We propose new, physically realizable attacks (DARTS) based on the concept of adversarial examples on traffic sign recognition systems that are robust in real-world settings. The novel Sign Embedding attack modifies innocuous signs such that they are classified as the adversary's desired traffic signs while the Adversarial Traffic Sign attack modifies existing traffic signs.
(2) We are the first to explore the idea of *Lenticular Printing* for creating traffic signs that can mislead traffic sign recognition systems. This method allows us to create signs that look different to humans and to the camera on an AV, based on differences in the viewing angle. We demonstrate the ease of carrying out such an attack and how it might be used in a real-world setting. Samples of the lenticular signs are in Figure 2(d).
(3) We construct an end-to-end pipeline for creating adversarial examples that fool traffic sign recognition systems and are resilient to transformations of the image that may occur during the image capture phase. Using this pipeline, we carry out an extensive evaluation of our attacks in both physical as well as virtual settings over various sets of parameters. We consider both the white-box threat model, where the adversary has access to the details of the traffic sign recognition system, and the black-box one, where such access is not present. In the virtual white-box setting, our Adversarial Traffic Sign attack (generated from the GTSRB test set) has a 99.07% success rate without randomized image transformations at test time and 95.50% with. We found that even in the black-box setting, the success rates remain high at 38.08% with transformations and at 47.77% without. This result suggests that on average one in three attacks succeeds in the physical black-box setting. Therefore, while not having any access to the classifier's information, the adversary only needs to put up a few fake signs that work on the substitute model locally trained by the adversary in order to have a successful attack with high probability.
(4) We also conduct a real-world drive-by test, where we attach a video camera to a car's dashboard and extract frames from the video for classification as we drive by (Figure 9). In the *white-box case*, our best Adversarial Traffic Sign and Sign Embedding attacks have success rates of 92.82% and 96.51% respectively, where the success rate is the number of frames in which the adversarial image is classified as the target divided by the total number of frames. In the *black-box case* using a different classifier, the same set of fake signs exhibit equally high success rates (96.68% and 97.71%).
(5) We also explore a possible defense against the attacks proposed. The defense, based on the concept of adversarial training, aids in securing ML systems against adversarial examples. The adversarially trained model substantially reduces the white-box adversarial success rate of the Adversarial Traffic Sign and the Sign Embedding attacks (36.35% for the virtual white-box attack on the GTSRB test set). However, future work is required to reduce the attack success rate even further, considering the adverse implications of each successful attack.

The rest of the paper is organized as follows: Section 2 describes both adversarial example and lenticular printing based attacks in detail, Section 3 provides numerous experimental results for these attacks, Section 4 explores possible defenses against these attacks, Section 5 discusses limitations and future work, Section 6 provides an overview of related work and Section 7 concludes.

[1]Link anonymized for double-blind submission

2 METHODOLOGY

In this section, we present new methods to generate physically robust adversarial examples as well as attacks based on lenticular printing. In particular, we introduce Sign Embedding attacks, which modify innocuous signs such that they are detected and classified as potentially dangerous traffic signs. We also examine the Adversarial Traffic Sign attack which modifies existing traffic signs such that they are classified in the attacker's desired class.

2.1 Threat models

We consider two commonly used threat models for the generation of adversarial examples against deep neural networks: *white-box* and *black-box*. The black-box setting also applies to the Lenticular Printing attack.

2.1.1 White-box. In the white-box setting, we assume that the adversary has complete access to the traffic sign recognition model including its architecture and weights. Further, we focus on the creation of *targeted* adversarial examples, since these are more relevant to an adversary aiming to misclassify traffic signs. We briefly justify the powerful attacker considered in the white-box setting. Consider an attacker who wishes to cause an AV to detect and misclassify a sign in its environment. In order to carry out this attack, it is conceivable that the attacker can purchase or rent a vehicle of the kind that is to be attacked, and 'reverse engineer' the classifier. It is known by querying a target classifier on a dataset and obtaining the classification outcomes, a surrogate classifier that closely mimics the target classifier can be trained [33]. Also, recent work has shown that query-based attacks can generate adversarial examples which are as effective as those generated using white-box attacks [25]. Thus, the white-box setting is an important one to consider.

2.1.2 Black-box. In this setting the adversary does not have direct access to the target model and may only have query access. In this work, we only consider the case when the adversary does not even have query access. In this setting, black-box attacks rely on the phenomenon of transferability [13, 34, 35], where adversarial examples generated for a model trained locally by the attacker, remain adversarial for a different, target model. The Lenticular Printing attack naturally operates in the black-box setting since it relies on an optical phenomenon and not on the internal structure of the classifier.

2.2 ML systems and adversarial examples

Machine learning systems typically have two phases, a training phase and a test phase [36]. Our focus is on attacks during the test phase, which are typically known as *evasion attacks*. These have been demonstrated in the virtual setting for a number of classifiers [14, 15, 37, 38]. These attacks aim to modify benign examples by adding a perturbation to them such that the modified examples are *adversarial*, i.e. they are misclassified by the ML system. In the case of attacks on the computer vision systems of autonomous cars, the goal of the adversary is to generate signs that appear benign to humans but are misclassified by the traffic sign recognition system.

2.2.1 Notation and description of ML classifiers. A classifier $f(\cdot;\theta) : \mathcal{X} \to \mathcal{Y}$ is a function mapping from the domain $\mathcal{X}$ to the set of classification outputs $\mathcal{Y}$. The number of possible classification outputs is then $|\mathcal{Y}|$. θ is the set of parameters associated with a classifier. Throughout, the target classifier is denoted as $f(\cdot;\theta)$, but the dependence on θ is dropped if it is clear from the context. $\mathcal{H}$ denotes the set in which an adversarial example must lie. $\ell_f(\mathbf{x}, y)$ is used to represent the loss function for the classifier f with respect to inputs $\mathbf{x} \in \mathcal{X}$ and their true labels $y \in \mathcal{Y}$.

An adversary can generate adversarial example $\tilde{\mathbf{x}}$ from a benign sample $\mathbf{x}$ by adding an appropriate perturbation of small magnitude [38]. Such an adversarial example $\tilde{\mathbf{x}}$ will either cause the classifier to misclassify it into a targeted class (targeted attack), or any class other than the ground truth class (untargeted attack). We focus entirely on *targeted attacks* since these are more realistic from an attacker's perspective.

Since *deep neural networks* (DNNs) [39] achieve very high classification accuracies in a variety of image classification settings [40, 41], they are being used for the computer vision systems of autonomous cars [21, 22]. We thus focus on attacks on DNNs in this paper and define some notation specifically for neural networks. The outputs of the penultimate layer of a neural network f, representing the output of the network computed sequentially over all preceding layers, are known as the logits. We represent the logits as a vector $\boldsymbol{\phi}^f(\mathbf{x}) \in \mathbb{R}^{|\mathcal{Y}|}$. The final layer of a neural network f used for classification is usually a softmax layer represented as a vector of probabilities $\mathbf{p}^f(\mathbf{x}) = [p_1^f(\mathbf{x}), \ldots, p_{|\mathcal{Y}|}^f(\mathbf{x})]$, with $\sum_{i=1}^{|\mathcal{Y}|} p_i^f(\mathbf{x}) = 1$ and $p_i^f(\mathbf{x}) = \frac{e^{\phi_i^f(\mathbf{x})}}{\sum_{j=1}^{|\mathcal{Y}|} e^{\phi_j^f(\mathbf{x})}}$. The loss functions we use are the standard *cross-entropy loss* [39] and the *logit loss* from Carlini and Wagner [15].

2.2.2 Generating virtual adversarial examples. To generate a *targeted* adversarial sample $\tilde{\mathbf{x}}$ starting from a benign sample $\mathbf{x}$ for a classifier f, the following optimization problem studied in [15] leads to state-of-the-art attack success rates in the virtual setting:

$$\begin{aligned} \min \quad & d(\tilde{\mathbf{x}}, \mathbf{x}) + \lambda \ell_f(\tilde{\mathbf{x}}, T), \\ \text{s.t.} \quad & \tilde{\mathbf{x}} \in \mathcal{H}. \end{aligned} \tag{1}$$

Here, $\ell_f(\cdot, \cdot)$ is the loss function of the classifier, d is an appropriate distance metric for inputs from the input domain $\mathcal{X}$, T is the target class and $\mathcal{H}$ is the constraint on the input space. λ controls the trade-off between minimizing the distance to the adversarial example and minimizing the loss with respect to the target. In essence, the optimization problem above tries to find the closest $\tilde{\mathbf{x}}$ to $\mathbf{x}$ such that the loss of the classifier at $\tilde{\mathbf{x}}$ with respect to the target T is minimized. For a neural network, $\ell_f(\cdot, \cdot)$ is typically highly non-convex, so heuristic optimizers based on stochastic gradient descent have to be used to find local minima [15].

For traffic sign recognition systems, the method described above produces adversarial examples which do not work well under the variety of conditions encountered in the real world. That is, if the examples generated using this method were directly used in Step 3 (mount sign and drive-by test) of 3, they would not perform well. In light of this, there has been some work towards generating physically robust adversarial examples [32, 42]. In this paper, we greatly refine their methods. Comparisons with our work are expanded upon in Section 6.

2.3 Robust adversarial example generation

In this section, we describe our methodology for generating robust, physically realizable adversarial examples. We first provide an overview of our attack pipeline from 3 and then describe each of its components in detail.

2.3.1 Attack pipeline. Our pipeline has three steps:

Step 1. Obtain the original image $\mathbf{x}$ and choose target class T that the adversarial example $\tilde{\mathbf{x}}$ should be classified as.

Step 2. Generate the digital version of the physically robust adversarial example as follows:

1. Generate mask M for the original image (A mask is needed to ensure that the adversary's perturbation budget is not utilized in adding perturbations to the background.)

2. Re-size both the original image and and the mask to the input size of the target classifier[2].

3. Run the optimization from Equation 2 to obtain the perturbation $\boldsymbol{\delta}$.

4. Re-size the output perturbation $\boldsymbol{\delta}$ and add it to the original image.

Step 3. Print and test the generated adversarial example for robustness in real-world conditions.

2.3.2 Image selection. We use a single image of a cropped traffic sign, re-sized to the input size of the classifier. This is because, in our case, for the gradient propagation to be applicable, the input image is transformed by the

[2] This is 32×32 pixels for the classifiers we use, described in Section 3.1.3

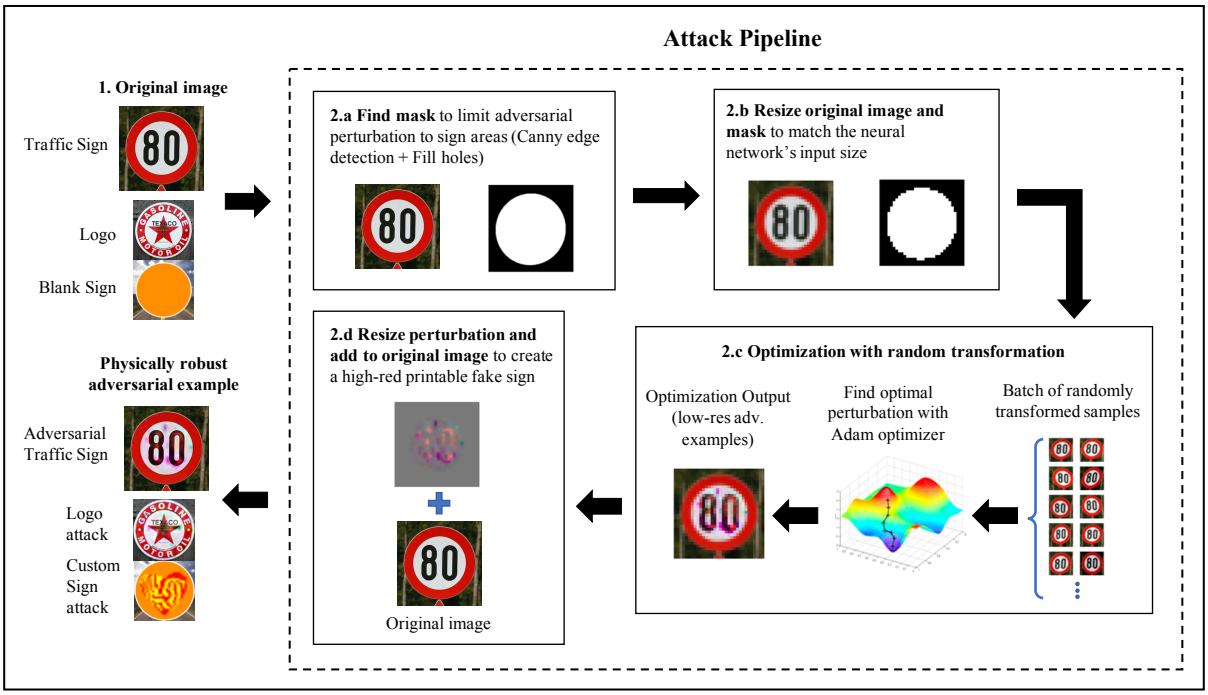


Fig. 3. **Overview of the Attack pipeline**. This diagram provides an overview of the process by which adversarial examples are generated for both the Sign Embedding attack and Adversarial Traffic Sign attacks.

set of predefined transformations with randomized parameters. Our method thus results in a low cost for the attacker with regards to constructing an adversarial example. In the case of Adversarial Traffic Sign attacks, the adversary only needs one image of the same class as the original sign which can be easily found on the internet. The image also does not need to be a real photograph; as we show in our Sign Embedding attacks, any logo or schematic drawing can be used as well.

2.3.3 Mask generation. A mask is a mapping from each pixel of the image to zero or one. Pixels mapped to zero are the ones that are not the sign. On the other hand, those mapped to one are on the sign area. To create a *mask*, we employ a simple image segmentation algorithm using Canny edge detection to outline the boundary of the sign and binary dilation to fill in the hole. The code is adapted from [43]. This algorithm works well on all the sign shapes given that the input image is not of poor resolution.

2.3.4 Optimization problem. Our adversarial example generation method involves heuristically solving a non-convex optimization problem using the Adam optimizer [44]. The problem set-up is adapted from the general concept of expectation over transformations [42]. An updating gradient is averaged over a batch of randomly transformed versions of the original image [32, 45]. The robust adversarial example can be written as a solution to the minimization problem given below for any input $\mathbf{x}$:

$$\min_{\boldsymbol{\delta} \in \mathbb{R}^n} \quad c \cdot \frac{1}{B} \sum_{i=1}^{B} [F(\tau_i(\mathbf{x} + M \cdot \boldsymbol{\delta}))] + \max(\|\boldsymbol{\delta}\|_p, L) \tag{2}$$

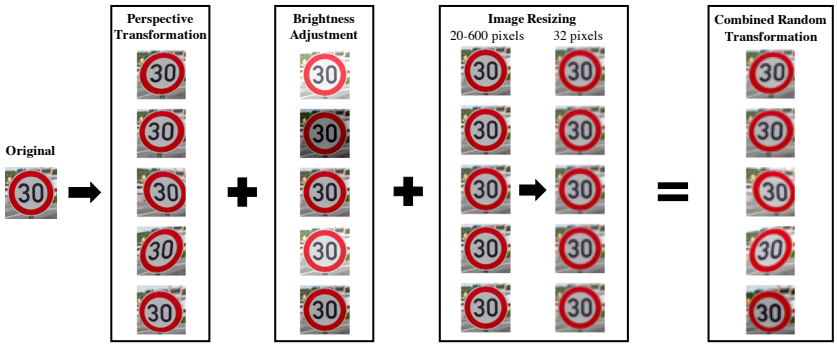


Fig. 4. **Transformations used during both training and evaluation to simulate real-world conditions.** The final column of images displays the type of images that are included in the sum in Eq. (2).

where $F(\mathbf{x}) = \max(\max_{j \neq T}\{\boldsymbol{\phi}(\mathbf{x})_j\} - \boldsymbol{\phi}(\mathbf{x})_t, -K)$ is the logit loss from [15] and $\boldsymbol{\phi}(x)_j$ is the j-th logit of the target network. $\tau_i : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a transformation function mapping within the image space ($n = 32 \times 32 \times 3$). M is a mask or a matrix of zeros and ones with the same width and height as the input image, and $M \cdot \boldsymbol{\delta}$ is an element-wise product of the mask and the perturbation to constrain the feasible region to the sign area. The objective value is penalized by a p-norm of the perturbation $\boldsymbol{\delta}$ in order to keep the adversarial perturbation imperceptible by humans, and the constant c is adjusted accordingly to balance between the real objective function and the penalty term. The constant K determines the desired objective value and thus, controls *confidence score* of the adversarial example. We introduce an additional constant L to explicitly encourage the norm of the perturbation to be at least L since overly small perturbations can disappear in the process of printing and video capturing.

To summarize, the optimization tries to minimize the average of the objective function evaluated on a batch of B images which are generated from a set of transformation functions $\tau_1, ..., \tau_B$. For our experiments, the transformation function is a composition of three different image modifications: (1) perspective transform, (2) brightness adjustment and (3) re-sizing which are described in detail next.

2.3.5 Transformations. Here, we describe the set of image transformations used in the optimization which are chosen to virtually generate a batch of input images resembling the varying real-world conditions in which images of the generated physical adversarial example might be taken. We included three different transformations in our experiments, each of which was randomized for a particular image while solving the optimization problem as well as while performing evaluations in the virtual setting. Examples of transformations used for both the optimization and evaluation phases are shown in Figure 4.

Perspective transformation - Orientations of a sign that appears on an image vary between camera angles, distances and relative heights. These orientations can be captured by perspective transformation, an image transformation that maps each of the four corners of a 2D image to a different point in the 2D space. Any perspective transformation can be characterized by eight degrees of freedom which can be represented by a 3×3 *projective transform matrix* with the last entry being set to 1. Other common image transformations such as rotation, shearing or scaling are subsets of the perspective transformation. Additionally, this transformation is simply a matrix multiplication so it is differentiable.

Brightness adjustment - Naturally, cars drive any time of the day so an adversary might need the fake sign to be able to fool the system under different lighting conditions. The simplest way to simulate settings with different amounts of light or brightness is to add a constant value to every pixel and every channel (R, G, B) of an image and then clip the value to stay in the allowed range of $[0, 1]$. Again, this transformation is differentiable

because it consists of an addition and clipping similar to an ReLU activation function which is differentiable almost everywhere.

Image resizing - Due to varying distances between a camera and a sign, an image of the sign detected and cropped out of the video frame could have different sizes or numbers of pixels. After generated, adversarial perturbation is upsampled to match its original image's size and when the adversarial sign is detected on the camera, it is cropped out and downsampled back to 32×32 pixels for the CNN to classify. This resampling process could blur or introduce artifacts to the image, diminishing the intended effect of adversarial perturbation. We use a *nearest neighbours* re-sizing technique for the mask and *bilinear interpolation* for all other images.

The randomized transformation τ used during both the generation and evaluation of adversarial examples is a concatenation of the three transformations mentioned above. The degree of randomness of each transformation is controlled by a set of tunable parameters to keep them in a range that provides realistic simulations. In the generation process, we experiment with different combinations of the parameters to observe their effect and find the best setting.

2.3.6 Setting up the optimization problem. As mentioned earlier, our optimization program introduces a number of parameters (c, K, L, T, θ, step size, choices of norm) that can be fine-tuned to control the trade-off between robustness of adversarial examples and their visibility to humans. We choose to use the L_1 norm to constrain the added perturbation which aids in the solution of the optimization problem and serves as a fairly good surrogate for human perceptibility of perturbations [46]. The parameters can each be optimally tuned using a line search or a grid search but when optimized in tandem they offer more benefits as they can be adjusted to suit the particular setting in which the adversary desires to operate. A detailed analysis of the procedure we followed to obtain the optimization parameters is given in Section 3.4.

2.4 Attack types

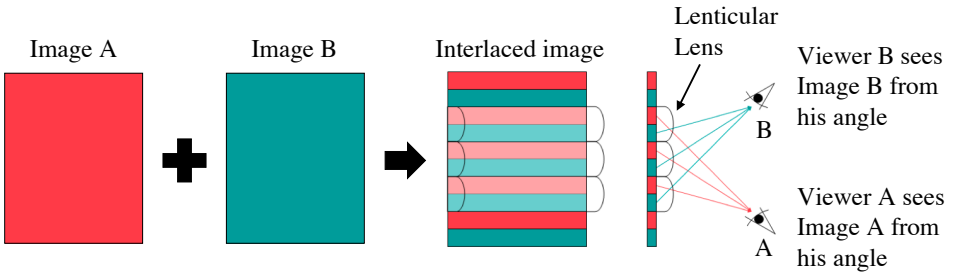
Next, we describe two different attacks that rely on the optimization problem laid out in Section 2.3.4: (1) the Sign Embedding attack, where the adversary is free to disguise adversarial examples as advertisement signs, drawings, graffiti, etc. and (2) the Adversarial Traffic Sign attack, where the adversary modifies existing traffic signs to make them adversarial.

2.4.1 Sign Embedding attacks. We propose a novel attack based on the concept of adversarial examples by exploiting the fact that the shape-based detection part of the traffic sign recognition pipeline can pick up a circular object that may not be a traffic sign. Under ordinary conditions when no adversarial examples are present, the false detection does not unduly affect the traffic sign recognition system since

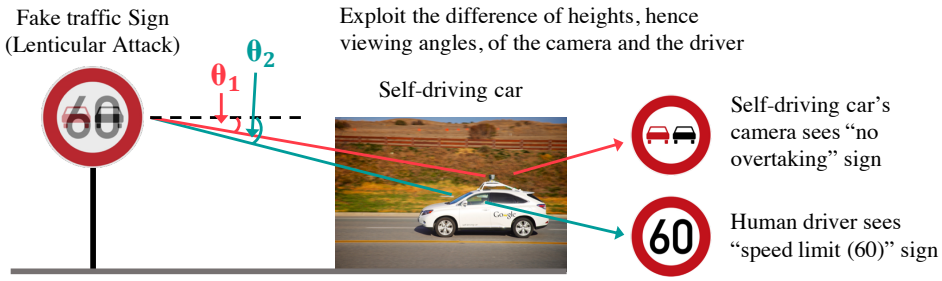
(1) the confidence scores corresponding to the predicted labels of these objects are low
(2) these circular objects are not consistently classified as a certain sign. The predicted label changes randomly as the background and the viewing angle varies across multiple frames in the video.

Therefore, a traffic sign recognition system, including ours, can choose to treat any detection with these two properties as an erroneous detection by setting the confidence threshold close to 1 and/or ignoring objects that are inconsistently classified. However, adversarial examples generated from these benign circular objects using our optimization are *classified consistently as target traffic signs with high confidence under varying physical conditions*. We demonstrate these observations experimentally in Section 3.2.3. Thus, using our attack pipeline, the adversary is now free to disguise adversarial traffic signs as advertisement signs, drawings, graffiti, etc. Here, we demonstrate two possible settings in which adversarial traffic signs can be embedded.

Setting 1: Logo attacks - In this attack, images of commonly found logos are modified such that they are detected and classified with high confidence as traffic signs (Figure 8b). Since these logos are omnipresent, they allow an adversary to carry out attacks in a large variety of settings. In this scenario, the problem setup (objective



(a) Overview of the process by which lenticular images are generated. Part of image modified from existing image on Wikimedia Commons.



(b) Principle behind an attack based on lenticular printing.

Fig. 5. **Illustration of the process of generating a lenticular image and using it to fool the traffic sign recognition system of a car.**

and constraints) is exactly the same as Equation 2 (i.e. the adversarial perturbation should be as small as possible while still being effective under transformations).

Setting 2: Custom Sign attacks - In this attack, the adversary creates a custom sign that is adversarial starting from a blank sign (Figure 8c). Custom-made masks on blank signs can lead to the embedding of adversarial traffic signs in inconspicuous, graffiti-like objects in the environment. This allows the adversary to create adversarial signs in almost any imaginable setting by using a mask to create images or text that are appropriate for the surroundings. In this attack, the original sign is a solid color circular sign and the norm of the perturbation is not penalized by the optimization problem but only constrained by the shape of the mask. This allows the adversary to draw almost any desired shape on a blank sign and the optimization will try to fill out the shape with colors that make the sign classified as the target class. In practice, this attack can also be carried out by the same optimization problem by setting c and L to some large numbers so that the optimization will focus on minimizing the loss function without penalizing the norm of the perturbation.

2.4.2 Adversarial Traffic Sign attack. In this attack, *existing* traffic signs are modified using imperceptible perturbations such that they are classified as a different traffic sign (Figure 8a). This attack is not as wide-ranging as the Sign Embedding attack, since it requires the modification of an existing sign or the installation of a new one. From a practical standpoint, installing a modified traffic sign is likely to be draw more attention than an innocuous logo, which leads us to believe that the Sign Embedding attack is more practical.

Remarks: Adversarial examples can be created starting from any sign such that they are classified with high confidence as a potentially dangerous traffic sign. Benign signs in the Sign Embedding attack case do not have this effect since they are usually classified with low confidence.

2.5 Lenticular Printing

Lenticular printing is a process used to create images that look different when viewed from different angles. A diagrammatic overview of the process is given in Figure 5a. The change in angle needed to view the different images can be small, in which case it can be used to provide a '3-D' effect (since the left eye and the right eye see slightly different images) or, more related in our case, a 'flip' effect where two or more images are interlaced and each one only shows up at certain viewing angles. For the purposes of fooling the computer vision systems of autonomous cars, we need the fake traffic sign to appear different to the camera mounted on the AV and the human controller or passenger seated in the car (Figure 5b).

Industrial quality lenticular printing involves specialized machinery and trained personnel to operate it. Nevertheless, a simple lenticular image can also be produced without the need for specialized equipment by using either an inkjet or a laser color printer and a lenticular lens. For our purposes, to demonstrate a proof of concept attack, creating a lenticular image involves two simple steps. First, we obtain two traffic sign images of the same dimensions, one of which is the adversary's desired target, and interlace them. Note that we could also use a logo or a custom sign of the same dimensions to interlace with the desired traffic sign. Then, we print the interlaced image on photo-quality paper and stick it on the back of a commercially available lenticular lens.

We use a free software called "SuperFlip" available online to interlace the chosen images [47] We choose lenticular lens with optically clear adhesive on one side for attaching the interlaced image. Otherwise, a laminator is required to attach the image to the lens, but using a laminator will also result in lenticular images with better quality. There are three specifications of lenticular lens that play an important role in getting the flip effect at the desired angles and distances: lines per inch (LPI), viewing distance which depends on LPI, and viewing angles. While we use commonly available lens with 40 LPI and a 49 degree viewing angle, industrial-quality lenses offer a wide range of the three specifications to choose from. The adversary may choose the lens that best suit a particular road or height of a target sign.

3 EXPERIMENTS

In this section, we first describe our experimental setup, after which we present results for both virtual and real-world attacks, in both white and black-box settings for adversarial examples. We also describe experiments related to lenticular printing.

3.1 Experimental setup

For our experiments we used a GPU cluster of 8 NVIDIA Tesla P100 GPUs with 16GB of memory each, running with no GPU parallelization.

3.1.1 Overview of traffic sign recognition pipeline. Our traffic sign recognition pipeline (Figure 6) consists of two stages: detection and classification. We utilize a commonly used recognition pipeline based on the Hough transform [48–50]. The shape-based detector uses the circle Hough transform [51] to identify the regions of a video frame that contain a circular traffic sign. Before using Hough transform, we smooth a video frame with a Gaussian filter and then extract only the edges with Canny edge detection [52]. Triangular signs can be detected by a similar method described in [49]. The detected image patch is cropped and passed on to the neural network classifier to determine whether it is a traffic sign and assign its label. Images classified with a low confidence score are discarded as false positives for detection. Figure 6 summarizes our traffic sign recognition pipeline, and the details are given below.

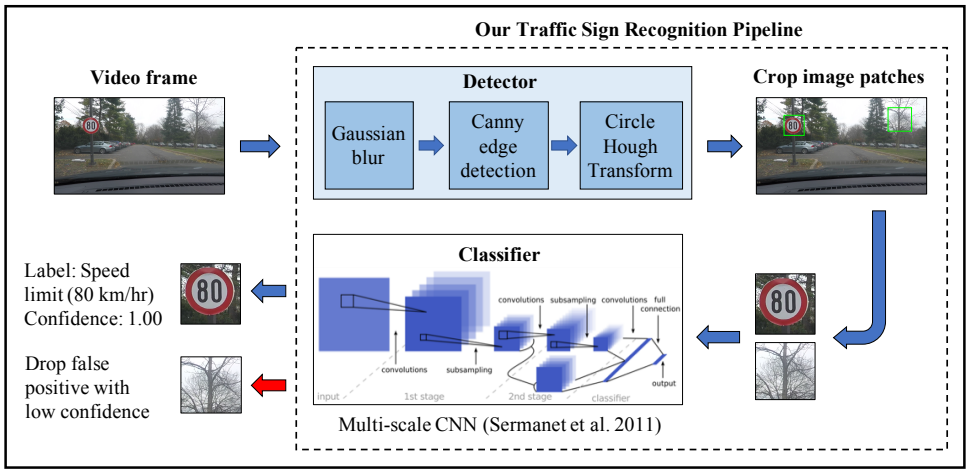


Fig. 6. **Sign recognition pipeline**. The pipeline consists of an initial detection phase followed by a multi-scale CNN as a classifier. In the virtual setting, a video frame is replaced by a still image.

3.1.2 Detector. For simplicity and without loss of generality, we design our detector to only locate circular signs on images using a well-known shape-based object detection technique in computer vision, Hough transform [51]. Triangular signs can be detected by a similar method described in [49]. In detail, our pipeline takes a video frame as an input and smooths it with a Gaussian filter to remove random noise. The processed images are passed through a Canny edge detector and then a circle Hough transform which outputs coordinates of the center and radii of the circles detected. The outputs are used to determine a square bounding box around the detected signs. The section of the input frame inside the bounding boxes are cropped out of their original *unprocessed* image and resized to 32×32 pixels which is the input size of the neural network classifier.

3.1.3 Classifier in the white-box setting (CNN A). Our classifier is based on a multi-scale convolutional neural network (CNN) [53]. The resized outputs from the detector are directly fed to the CNN which, in turn, outputs confidence scores for each of the 43 classes. The label with the highest confidence is chosen as the final output only if its confidence is above a certain threshold. The detected objects that are not traffic signs generally have a low confidence score and are filtered out by the threshold. This assumption is further proven by examples in Section 3.2.3.

Our CNN has 3 convolutional layers and 2 fully connected layers. There is also an extra layer that pools and concatenates features from all convolutional layers, hence the name multi-scale. The training procedure is adpated from [54]. The model is trained on data-augmented training set (86,000 samples, 2,000 for each class) generated by random perspective transformation and random brightness and color adjustment. There is no image preprocessing. The training takes 41 epochs with learning rate of 0.001. Dropout and weight regularization ($\lambda = 1e{-}3$) are used to reduce overfitting. The accuracy of the model, which will be referred to as CNN A, on the validation set is **98.50%**. The model is defined in Keras [55] using Tensorflow [56] backend. The entire recognition pipeline (the detector combined with the classifier) is tested with the German Traffic Sign Detection Benchmark (GTSDB) [57] where it achieves a reasonable mean average precision (mAP) [58] of **77.10%** at the intersection-over-union (IoU) of 0.5.

3.1.4 Classifier in the black-box setting (CNN B). For the black-box threat model, described in Section 2.1.2, we assume that the adversary has no access to any information regarding the neural network classifier. Therefore,

Fig. 7. **The auxiliary dataset**. Some high-resolution images sampled from the auxiliary dataset. The auxiliary dataset is a combination of real traffic sign photos and computer-generated drawings in front of a simple background.

to evaluate our attacks in this case, we keep the recognition pipeline the same but simply replace CNN A with a different CNN which will be referred to as CNN B for the rest of this paper. CNN B is a standard CNN (not multi-scale) with four convolutional layers and three dense layers. It is trained on the same dataset as CNN A, and achieves an accuracy of **98.66%** on the validation set of GTSRB. The mean average precision (mAP) of the entire pipeline with CNN B on the GTSDB dataset is **81.54%**. In the black-box setting, the attacks are generated based on CNN A which is assumed to be trained by the adversary as a substitute model for the true classifier in use, CNN B. All experiments in Section 3.3 use this black-box pipeline to evaluate the attacks.

3.1.5 Datasets. We use the German Traffic Sign Recognition Benchmark (GTSRB) [59] to train and test the classifier. GTSRB is a widely-used standard dataset for traffic sign recognition. It contains more than 50,000 images of 43 types of traffic signs in total. Each image is a 32×32 RGB image with the pixels scaled to lie in $[0, 1]$. We choose GTSRB over the LISA Traffic Sign Dataset [60] because GTSRB offers a much larger number of samples so we can be more confident that the success of the attacks does not come from the weakness of the classifier due to the lack of training data.

Auxiliary high-resolution dataset. To create fake traffic signs that look realistic, we searched for a number of high-resolution traffic sign images to use as original images to generate the Adversarial Traffic Sign attacks with our attack pipeline. They are a mixture of real photographs and computer-generated drawings, which are much easier to find than the former, on an arbitrary background image. We passed the images through our mask generation module and chose only ones that provide an accurate mask. We ended up with 22 images, some of which are shown in Figure 7. For the rest of the paper, we refer to them as *the auxiliary dataset*.

3.1.6 Evaluation metrics. We again note that unless specified otherwise, we only consider the *targeted attack* scenario and an adversarial example is considered successful if it is classified as the target label. Thus, the attack success rate in the virtual setting is

$$\text{Virtual attack success (VAS)} = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{f(\tilde{\mathbf{x}}_i)=T_i \text{ and } f(\mathbf{x}_i)=y_i\}}}{\sum_{i=1}^{N} \mathbb{1}_{\{f(\mathbf{x}_i)=y_i\}}} \tag{3}$$

where $\tilde{\mathbf{x}}_i$ is the adversarial example generated from the i^{th} original sample $\mathbf{x}_i$ with the target label T_i and the original label y_i.

Manually printing out and evaluating the effectiveness of adversarial examples in a real-world setting is prohibitively expensive. In light of this, we propose a method to evaluate how physically robust adversarial examples are likely to be by simulating varying conditions in which the images of them might be taken. This allows us, and by extension, an adversary, to determine which adversarial examples are likely to work in a real-world setting and test those in the type of real world setup described in Section 3.5. The physical conditions

Attacks	VAS	SPAS	DR	Avg. norm (L_1)
Vanilla Optimization	96.38%	9.88%	89.75%	0.18
Vanilla Optimization (our parameters)	97.91%	46.74%	52.26%	30.45
Adversarial Traffic Sign	99.07%	95.50%	3.60%	31.43

Table 1. **Comparison of our method (Eq. (2)) with the Vanilla Optimization problem (Eq. (1)) in terms of attack success rates, deterioration rate and average norm in the virtual white-box setting.** These results are for 1000 randomly chosen samples from the GTSRB test set.

are virtually simulated by a combination of the randomized image transformations described in Section 2.3.5. w (we use 10 here) of these sets of transformation are applied to each adversarial and original sample, and the transformed images are directly fed into the target classifier to determine the predicted labels. The *simulated physical attack success* is then

$$\text{Simulated physical attack success (SPAS)} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{w} \mathbb{1}_{\{f(\tau_j(\tilde{\mathbf{x}}_i))=t_i \text{ and } f(\tau_j(\mathbf{x}_i))=y_i\}}}{\sum_{i=1}^{N}\sum_{j=1}^{w} \mathbb{1}_{\{f(\tau_j(\mathbf{x}_i))=y_i\}}} \tag{4}$$

Another important metric needed to properly evaluate the physical robustness of adversarial examples, is the *deterioration rate* of attack success after transformations. It can be formalized as

$$\text{Deterioration rate (DR)} = 1 - \frac{\text{SPAS}}{\text{VAS}} \tag{5}$$

A higher deterioration rate implies that the adversarial examples are more likely to degrade under the transformations, and by extension in a real-world setting.

In order to measure the perceptibility of the perturbations, we compute the L_1 norm of the perturbation for each adversarial example, successful or not, and average it over the entire set of data. Similarly, to determine how confident the classifier is on successful adversarial examples, we compute and report the average confidence of the target class over all successful adversarial examples.

Remark (virtual and physical settings). All results reported in the *virtual* setting involve benign and adversarial examples that remain in a digital format throughout. The results for *simulated physical attacks* (SPA) also pertain to images of this nature. Only the results in Section 3.5 are for adversarial examples that were printed out and then subsequently passed through the recognition pipeline.

3.2 Virtual white-box attacks

The slowest step in the generation of each adversarial example is running the optimization from Eq. (2). Each example takes about 60s to generate on the hardware we use. We stop the optimization run after 3000 steps for each sample since the change in loss between subsequent runs is vanishingly small.

3.2.1 Adversarial traffic sign. We compare the adversarial examples generated using our method to those generated by the optimization problem in Eq. (1) (referred to as Vanilla Optimization) on a random subset of 1000 traffic signs chosen from the test data of the GTSRB dataset. The result shown in Table 1 demonstrates that our attack has a much lower deterioration rate (DR) compared to the Vanilla Optimization attacks. The Vanilla Optimization attack with the same parameters as our attack indicates that forcing perturbation to be large (having a large norm) can increase its robustness to transformation to a certain degree. However, our attack can achieve a much lower deterioration rate with a similar average norm emphasizing the substantial effect the

Attacks	VAS	SPAS	DR	Avg. confidence
Adversarial Traffic Sign (auxiliary dataset)	54.34 %	36.65 %	32.6%	0.9721
Logo	85.71%	65.07%	24.1%	0.9753
Custom Sign	29.44%	18.72%	36.4%	0.9508

Table 2. **Attack success rates and deterioration rate for Sign Embedding and Adversarial Traffic Sign attacks in the virtual white-box setting with auxiliary dataset.**

addition of random transformations to the optimization problem has on improving the physical robustness of adversarial examples.

In addition to the experiment with test images from the GTSRB dataset, we use high-resolution images from the auxiliary dataset to generate Adversarial Traffic Sign attacks for the physical setting. The GTSRB test images are too low-resolution for large-scale printing. Similarly to the Logo attack experiment, each of the 22 images in the auxiliary dataset is used to generate 20 Adversarial Traffic Sign attacks for 20 randomly chosen target traffic signs. Therefore, one experiment contains 440 attacks in total. The attack success rates of the attack with the auxiliary dataset is reported in Table 2.

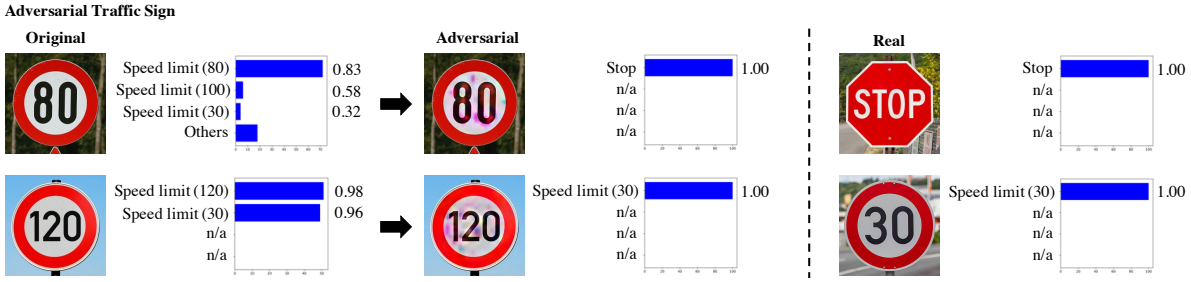
Remarks. We note that there is a difference in the VAS achieved with the auxiliary dataset and the GTSRB data for the Adversarial Traffic Sign attack. We believe this difference occurs due to the existence of a mask for the auxiliary dataset, which restricts perturbations to only lie on the sign and also due to the up- and down- sampling that occurs during the process of adding adversarial perturbations. This same effect carries over to the Logo and Custom Sign attacks, which use data of a higher resolution than the input to the classifier.

3.2.2 Sign Embedding attack. In this attack, we modify high-resolution images of innocuous signs and logos such that they are classified in the desired target class.

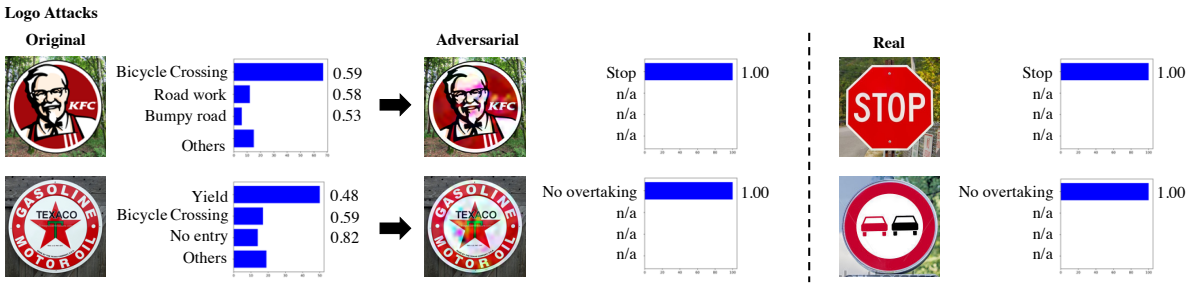
Logo attack. We use seven different circular logos as the original images to generate the Logo attacks. They are high-resolution images, similar to the auxiliary dataset. Each of the seven original logos is passed through our attack pipeline to create 20 different Logo adversarial examples, each meant to be classified as a different, randomly chosen target traffic signs, totalling 140 Logo attacks. While the adversarial perturbation on the logos significantly affects the classification results, it is generally not very visible to humans and usually blends in with the detail of the logos. The attacks achieve an impressive VAS of 85.71% and SPAS of 65.07% as reported in Table 2. Some successful Logo attacks are displayed in Figure 8b. The adversarial signs are classified as their corresponding target label with high confidence. This way, the adversary can expect the desired misclassification with much higher probability and consistency than simply putting up a random sign and hoping that it will be classified as a certain traffic sign.

Custom Sign attack. Three circular blank signs of the colors blue, orange, and white are drawn on the computer and used as the original images for the Custom Sign attacks. Each blank sign is matched with six different custom masks (a heart, a star, words "Hello, World!", etc.). Each of the total 18 pairs of a blank sign and a mask is used to generate 10 Custom Sign attacks which would be classified as 10 randomly chosen target traffic signs. In total, 180 Custom Sign attacks are created for the virtual evaluation.

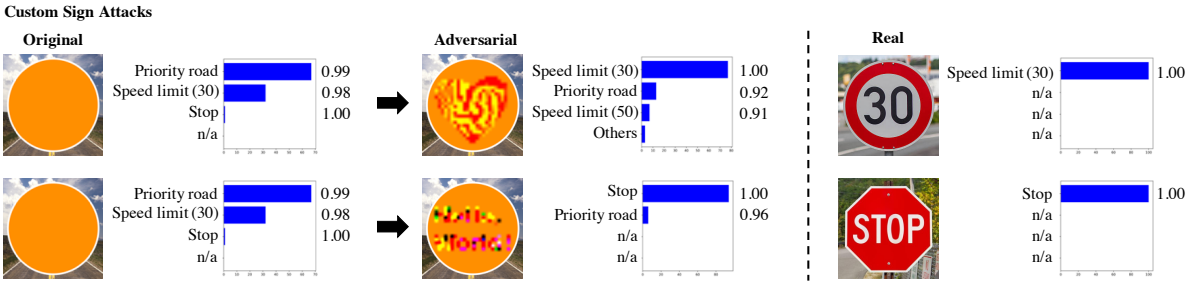
Some Custom Sign attacks are shown in Figure 8c. The attack produces adversarial signs that contain the shape of the mask filled with various colors. In Figure 8c, we pick some of the signs whose mask is filled well so the text or the shape is clearly visible. Some of the attacks do not fill up the entire mask resulting in incomplete shapes. This attack is considerably different from the Logo and the Adversarial Traffic Sign attacks as the optimization constraint is moved from the norm of the perturbation to its location instead. This new constraint seems to be more strict than the previous one, causing the attack success rate to drop as shown in Table 4.



(a) Classification of Adversarial Traffic Sign attack examples. The adversarial examples are classified with high confidence as a real traffic sign.



(b) Classification of Logo attack examples. The adversarial examples are classified with high confidence as a real traffic sign, in spite of being out of the dataset.



(c) Classification of Custom Sign attack examples. The adversarial examples are classified with high confidence as a real traffic sign, in spite of being custom made signs.

Fig. 8. **Frequency of the top-3 labels the given images are classified as under 100 different randomized transformations**. The bar chart on the right of each image provides the average confidence of the top classification outcomes over the 100 randomized transformations.

3.2.3 Confirming hypothesis about classifier confidence. To confirm our earlier hypotheses with regard to the confidence of classification for signs outside of the training set, we apply random transformations to create 100 images for each of the logo signs. The "Original" column of Figure 8b shows that the logo signs are classified as different classes depending on the transformation and with low confidence on average. As a comparison, the

Attacks	Black-box VAS	Black-box SPAS	Avg. confidence
Adversarial Traffic Sign (GTSRB test data)	47.77%	38.08%	0.8838
Adversarial Traffic Sign (auxiliary dataset)	7.14%	6.08%	0.9273
Logo	14.28%	12.57%	0.8717
Custom Sign	3.33%	6.00%	0.7193

Table 3. **Black-box attack success rates for Adversarial Traffic Sign and Sign Embedding attacks.** Adversarial examples generated for CNN A are tested on CNN B.

"Real" column of Figure 8 shows that real traffic signs (one of the 43 labels) are consistently classified as the correct label with probability very close to 1.

Again, under 100 different randomized transformations, the Custom Sign attacks are mostly classified as the target class with high probability. Note that even though the original solid color signs shown in the "Original" column of Figure 8c are classified with high confidence, their classified labels are uncontrollable by the adversary and are highly dependent on the signs' orientation and brightness. For example, a slight change in the camera angle may yield different labels which is an undesirable effect in the adversary's perspective.

Main takeaway. Both the Sign Embedding and Adversarial Traffic Sign attacks achieve high attack success rates of up to 90% in the virtual white-box setting, along with high-confidence classification for the successful adversarial examples. We note that from a practical perspective, the adversary only needs a few signs to achieve her aims, so even an attack success rate of 50% is high, since this presents the adversary with a large number of possible options to try.

3.3 Virtual black-box attacks

As discussed in Section 2.1.2, the phenomenon of transferability enables black-box attacks on ML classifiers when query-access is not present. Since we generate all our adversarial examples for CNN A, we test their effectiveness on CNN B. Note that we assume the adversary has access to the training dataset but this is a valid assumption in many cases since datasets are often open-sourced.

Below, we demonstrate that our physical adversarial examples are effective even in the black-box scenario where the adversary does not have access to the architecture nor the weights of the neural net classifier. We tested the adversarial examples generated from the recognition pipeline with CNN A on the one with CNN B, as described in detail in Section 3.1.4.

Table 3 shows adversarial success rate in a black-box setting, suggesting the transferability of adversarial examples. As expected, while there is a significant drop in adversarial success rates, in real-world scenarios, the adversary can simply put up different fake signs, and only one of them needs to successfully fool a traffic sign detector. We demonstrate the success of our black-box attack in a real-world setting in Section 3.5.

3.4 Parameter tuning

The set of initial parameters was manually chosen to maximize the VAS on the auxiliary dataset while maintaining a low average perturbation as measured in the L_1 norm. We use the L_1 norm since we found it to achieve the best trade-off between the various performance metrics such as VAS, SPAS and DR that we considered. Due to this trade-off, we chose not to use a grid search to fine the optimization parameters. Once we had a set of parameters that worked well, we changed each of the parameters individually to understand their effect on

Description	Parameters	VAS	SPAS	DR	Avg. norm (L_1)
Chosen params	$c = 3, K = 100, L = 30, T = 128, \theta_1 = 0.07, \theta_2 = 0.15$	54.34%	36.65%	32.55%	37.71
Perturbation norm	L_2 norm ($c = 0.2, L = 2$)	27.04%	15.09%	37.23%	76.74
	L_2 norm ($c = 0.02, L = 2$)	14.03%	7.76%	44.69%	54.13
	L_∞ norm ($c = 5e{-}5, L = 0.1$)	43.37%	24.41%	43.72%	162.76
Adversarial confidence	$K = 50$	48.21%	28.58%	40.71%	32.89
	$K = 200$	**59.69%**	**50.91%**	**14.71%**	**50.53**
# of transformed samples	$T = 32$	54.59%	32.08%	41.23%	35.44
	$T = 512$	46.43%	37.81%	18.57%	35.82
Degree of transformation (θ_1: perspective transformation, θ_2: brightness adjustment)	$\theta_1 = 0, \theta_2 = 0$	31.89%	6.26%	80.37%	31.87
	$\theta_1 = 0.03$	44.13%	8.88%	79.88%	33.21
	$\theta_1 = 0.15$	**51.79%**	**44.88%**	**13.34%**	**43.70**
	$\theta_2 = 0$	52.55%	34.56%	34.23%	36.61
	$\theta_2 = 0.075$	52.80%	35.68%	32.42%	36.94
	$\theta_2 = 0.30$	50.26%	39.34%	21.72%	39.40

Table 4. **Variation in attack success rates, deterioration rates and average norm of attack with different sets of optimization parameters.** Rows with numbers in **bold** represent parameter settings that achieve a good trade-off between the various performance metrics. However, both these rows have a higher average perturbation norm than the chosen set of parameters.

the attack performance metrics. Fist, we set the parameter L to control the norm of the output perturbation. Depending on the type of norms used in the objective function, L can be chosen roughly to any number larger than zero to force the algorithm to explore more solutions that cause targeted misclassification instead of finding those that merely minimize the norm. With L chosen for each norm, we vary the constant c, norm p, the number of transformation T, and degree of randomness of the transformations. We use θ_1 for perspective transformation, θ_2 for brightness adjustment and θ_3 for image resizing. We find that varying θ_3 does not significantly affect the outputs and thus do not discuss it further

The baseline attack uses L_1 norm with the parameters $c = 3, K = 100, L = 30, T = 128, \theta_1 = 0.07, \theta_2 = 0.15$. Table 4 shows results from 13 experiments each of which vary specified parameters from the baseline. We use Adam optimizer to solve the optimization problem with learning rate (step size) of 0.02 without decay.

It must be noted that all four result columns of Table 4 must be viewed in conjunction, instead of only considering one column, because they represent the trade-offs of the attacks. For example, we can see that by increasing the degree of randomness (θ_1, θ_2), attack success rates generally increase, but the norms also become larger making the perturbation more noticeable. In particular, setting $\theta_1 = 0.15$ results in the second highest physical success rate and the lowest deterioration rate, but it also produces perturbation with the second largest norm. Similarly, setting K to be larger encourages the optimization to look for more *adversarial* solutions which also comes with the cost of a large perturbation. Increasing the number of transformations makes the attack more successful under transformation while reducing the success rate in the virtual setting slightly. In fact, using 512 transformations produces both high physical attack success rate and a small norm for the perturbation with only the expense of longer computation time. For an adversary who only needs a few adversarial signs, he or she can afford to use a large number of transformation or even run a search for optimal parameters for a specific setting. For the Custom Sign attack, since there is no constraint on the norm of

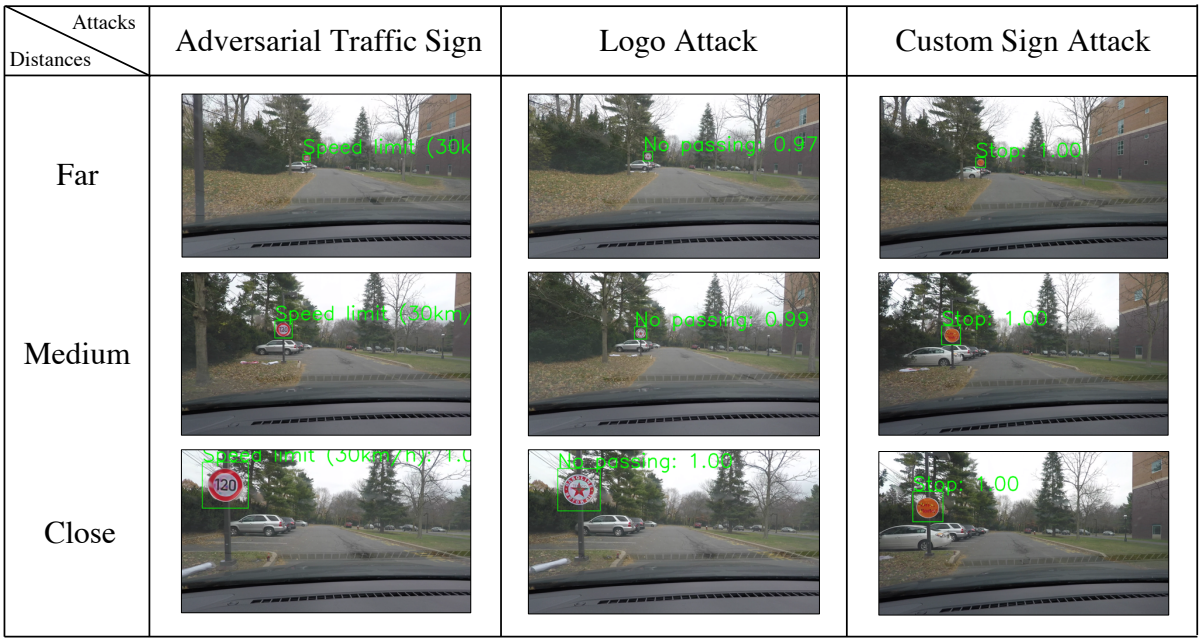


Fig. 9. **Classification outcomes for adversarial examples in the drive-by test.** All combinations of distances and attacks lead to the desired classification outcome with high confidence. Even frames captured at around 25 metres from the sign lead to high confidence targeted misclassification.

the perturbation both c and L and are increased to obtain perturbations that fit within the mask and are adversarial.

Main takeaway. The number of tunable parameters in the optimization represents an advantage for the adversary since they can be tuned to achieve the desired trade-off between the different performance metrics. We chose a particular set of parameters that worked well for our evaluation setting, and clarified the changes in performance that occur when these are tweaked.

3.5 Real-world attacks

To evaluate real-world attacks, we use a different metric since the number of adversarial examples used is smaller. In this section, the attack success rate reported is computed by counting the number of frames in which a sign was detected in the input frame and classified as the target class and dividing this number by the total number of frames in which there was a detection. In other words, we can express this as:

$$\text{Drive-by attack success rate} = \frac{\text{No. of frames that the sign is misclassified}}{\text{No. of frames that the sign is detected}} \tag{6}$$

3.5.1 Adversarial examples: white-box. To demonstrate the effectiveness of our adversarial examples in the real world, we carried out *drive-by tests* (shown in Figure 9) on two samples from each of our attacks (Adversarial Traffic Sign, Logo, and Custom Sign). We chose only those samples that performed well under the simulated physical attack. Each sample is re-sized to 30×30 inches and printed on a high-quality poster. The printed signs are stuck on a pole on the left side of the road that the car would be passing by. A GoPro HERO5 was mounted behind the car's windshield to take videos of 2704×1520 pixels at 30 frames per second. Starting around 25 meters

(a) Camera height of 1.5 metres. Signs classified as 'Speed limit 60', 'Turn left' and 'Turn right' respectively, with a confidence of 1.0.

(b) Camera height of 0.8 metres. Signs classified as 'No passing', 'Speed limit 60' and 'Turn left' respectively.

(c) Camera height of 0.5 metres. Signs classified as 'No passing', 'Speed limit 60' and 'Turn left' respectively.

Fig. 10. **Proof of concept implementation of the Lenticular Printing attack.** These images show that if the camera used for the traffic sign recognition module of an AV is at a different height from the human controller, then the Lenticular Printing attack can fool the classifier, while appearing to be correct to the human.

from the sign, the car approached it with an approximate speed of 16kmph. The traffic sign recognition pipeline only detects and classifies once every five frames to reduce the processing time and redundancy.

Each of the drive-by attack success rates reported below is an average of three runs. For the Adversarial Traffic Sign, 92.82% of the detected frames are classified as the adversary's target label, 52.50% for Logo attack, and 96.51% for the Custom Sign attack.

3.5.2 Adversarial examples: black-box. In a manner identical to Section 3.3, to set up the black-box setting, we replaced the neural network classifier CNN A, in the traffic sign recognition pipeline with CNN B and kept the other parameters the same. We use this new pipeline to evaluate the same videos captured in the drive-by tests.

With the same set of signs that perform best in the white-box case, the Adversarial Traffic Sign and the Custom Sign attacks both achieve very high success rates of 96.68% and 97.71% respectively, comparable to the white-box setting. The Logo attack, on the other hand, has an adversarial success rate of 0.00%, compared to 52.50% in the white-box case. However, a different Logo attack achieves an adversarial success rate of 47.43% in the white-box case and 32.73% in the black-box one. These results ascertain that some adversarial examples generated from one network are capable of fooling other networks, thus relaxing the assumption that the adversary must have access to weights and architecture of the target network.

3.5.3 Lenticular printing: black-box. In these experiments, we aim to show that the difference in viewing angle between the camera on an AV and the human controller or passenger can lead to an attack, since the human will see an appropriate sign, while the camera will see the targeted sign, which could be dangerous in that environment. To simulate what the human will see, we take pictures from different heights showing the sign

changing. We emphasize that the adversarial nature of these examples stems purely from the viewing angle, and the human and the camera would recognize the sign as the same if they viewed it from the same angle.

In our experiments with the Lenticular Printing attack, we stuck three different signs on an indoor surface (Figure 10). These signs flip between 'Speed limit 60' and 'No Passing', 'Turn left' and 'Speed limit 60', and 'Turn left' and 'Turn right' respectively. We take pictures of these signs at different heights using a standard mobile phone camera at a distance of 1.9 metres from the signs. The camera is held parallel to the plane of the cupboard. In all cases, when passed through our traffic sign recognition pipeline, each sign classified was classified as the appropriate one for that viewing angle with high confidence.

Main takeaway. In the drive-by tests, both the Sign Embedding and Adversarial Traffic Sign attacks achieve attack success rates in excess of 90%. *Videos of these attacks in practice are provided as supplementary material.* These decrease to around 40% in the black-box setting, but even this represents a successful attack for the adversary, since she only needs it to be misclassified in a few frames to cause harm. The Lenticular Printing attack is also demonstrably successful in an indoor setting.

4 COUNTERMEASURES: ADVERSARIAL TRAINING

In this section we examine a possible defense against the attacks described earlier in the paper. The defense is based on the concept of *adversarial training* [14]. In this paper, we are the first to analyze the effectiveness of adversarial training, and its recent variant, iterative adversarial training [61] against physically robust adversarial examples. While adversarial training can aid in defending against attacks based on adversarial examples, it is not effective against lenticular printing based attacks.

The simplest attack against DNNs is known as the Fast Gradient Sign (FGS) [14] attack, and it is an *untargeted attack* that involves adding a perturbation proportional to the sign of the gradient of the loss function of the neural network:

$$\tilde{\mathbf{x}} = \mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} \ell_f(\mathbf{x}, y)). \tag{7}$$

The advantage of this attack is that it is extremely fast to carry out so it can be incorporated into the training phase of neural networks to make them more robust to adversarial examples. The *training loss* is modified as follows:

$$\ell_f^{\text{adv}}(\mathbf{x}, y) = \alpha \ell_f(\mathbf{x}, y) + (1 - \alpha)\ell_f(\tilde{\mathbf{x}}, y), \tag{8}$$

where $\alpha \in [0, 1]$ controls the influence of the adversarial component of the loss. This modification to the loss function has been found to increase the robustness of DNNs against both targeted and untargeted adversarial examples [14]. Ideally, in order to defend against adversarial examples generated using Eq. (2), the $\tilde{\mathbf{x}}$ in Eq.(8) should be those same examples. However, as mentioned earlier, each adversarial example generated using our method takes around 60s to generate. This makes it impractical to run the optimization for every training batch.
Note about norms. We note that the perturbation in the FGS attack is constrained using the L_∞ norm. This choice is not unique with regards to carrying out fast attacks and any p norm can be used. In our experiments, we tried training with adversarial examples constrained with the L_1, L_2 and L_∞ norms and found the defense to work best with an L_∞ norm constraint.

4.1 Results with adversarial training

We trained CNN A using the loss function in Eq. (8) with $\alpha = 0.5$ and $\epsilon = 0.3$ for each adversarial example generated during the training process. While our attack samples have on average a maximum L_1 perturbation of around 30, the L_1 with that radius is too large to train with, since the ball has to lie in the unit hypercube. Since most of the adversarial examples do not modify each individual pixel by more than 0.3, it is a reasonable upper

Attacks	VAS	SPAS	DR	Avg. confidence
Adversarial Traffic Sign (GTSRB test data)	36.35%	27.52%	24.29%	0.9428
Adversarial Traffic Sign (auxiliary dataset)	2.53%	2.47%	2.37%	0.9358
Logo	11.42%	7.42%	35.02%	0.8054
Custom Sign	6.67%	5.77%	13.49%	0.9957

Table 5. **Attack success rates and deterioration rate on adversarially trained CNN A for Sign Embedding and Adversarial Traffic Sign attacks in the virtual white-box setting with auxiliary dataset.**

limit to use while training. The model was trained for 15 epochs and has an accuracy of 96.37% on the GTSRB validation set. The virtual white-box attack results are given in Table 5 and demonstrate the effectiveness of adversarial training as a defense as the VAS on the GTSRB test data drops from 99.07% to 36.35%. Similarly, the VAS for the Adversarial Traffic Sign attack using auxiliary dataset drops to 54.34% to 2.53%, from 85.71% to 11.42% for the Logo attack and from 29.44% to 6.67% for the Custom Sign attack. It has a similar effect on the SPAS.

Main takeaway. While the adversarial training defense does not completely eliminate the presence of adversarial examples, it helps to mitigate their impact. It results in a small drop in accuracy on benign samples, which can be problematic in real-world settings. We leave the exploration of countermeasures with better trade-offs and which are able to handle the Lenticular Printing attack for future work.

5 LIMITATIONS AND FUTURE WORK

In this section, we discuss some limitations of our approaches for both attacks and defenses and outline possibilities for future work to overcome these limitations.

5.1 More complex detectors

To improve the robustness of the traffic sign recognition system and reduce the false positive rate in real-world settings, the system could be set up to filter out detected objects that are not consistently classified (with high confidence scores) as the same type of signs across a sequence of multiple frames. This design decision can help reduce the false positive rate because erroneous detection such as random noise and other circular objects are unlikely to be classified as the same label with high confidence across multiple frames. In our current setup, we find a low false positive rate for the adversarial examples, which indicates that they would perform well even under such a modified system.

Further, while our system has performed well in real-world settings in spite of its simplicity, more robust traffic sign recognition systems have been proposed recently in the literature. These are typically neural network based detectors such as *Faster R-CNN* [62] and *YOLOv2* [63] that simultaneously handle both detection and classification in real time. However, attacking these detectors is not a simple extension of ideas explored in this paper due to the mechanism by which recognition occurs. Adversarial examples for these detectors need to remain adversarial regardless of their position in the captured frame [64]. While some recent work has proposed attacks on these detectors, these attacks are either restricted to virtual settings [65] or add overly large perturbations making recognition difficult for humans as well [66]. We are currently working on overcoming these issues and demonstrating powerful real-world attacks on state-of-the-art detectors.

5.2 Fooling synthesis of sensor inputs

The computer vision subsystem of an AV, while critical, is not the only actor in the decision making process when confronted with a new situation. A number of other sensors such as LIDAR, GPS, radar etc. also provide inputs which are then synthesized to come up with a decision [67]. While the computer vision subsystem is the only one able to recognize a traffic sign, the other sensors may be able to indicate that the sign recognized is incompatible with their inputs. This consideration is out of the scope of the current work, but we plan to explore simultaneous attacks on these varied subsystems in future work.

5.3 Black-box attacks

While our results in Section 3.5 for real-world black-box attacks were encouraging, there is scope for improvement. In particular, methods such as using an ensemble of local models to increase targeted transferability [35] have been successfully demonstrated for real-world attacks on ML models hosted by MLaaS providers and could be useful in achieving higher rates of transferability. We plan to investigate if our process of generating physically robust adversarial examples by adding transformations during the optimization process plays a role in increasing the transferability of adversarial examples. Further, the existence and effectiveness of black-box adversarial examples for neural network based detectors is an open research question which we plan to explore in future work. The results for lenticular printing we presented in this work are exciting as they represent a completely unexplored dimension for attacks on autonomous cars but we believe there is scope for further research in this direction. We plan to explore the creation of more robust and consistent lenticular printed samples in future work, which are effective for a broader range of angles.

5.4 Choice of norm

In this paper, we chose the L_1 norm to measure the visibility of adversarial perturbations as done in several previous works on virtual attacks [15, 46, 68]. However, it is still an open research question if this is the best choice of distance function to constrain adversarial perturbations. Previous work on generating adversarial examples has explored the appropriateness of other commonly used Euclidean norms such as L_∞ [14] and L_2 [15, 69] as well. Especially in the context of future work on generating physically realizable adversarial examples, we encourage further work on conducting user studies to determine the most appropriate proxy for measuring human perceptibility of adversarial perturbations.

6 RELATED WORK

There is now a large body of work on virtual attacks on machine learning systems. Evasion attacks, occurring in the test phase, have been proposed for Support Vector Machines [11], random forests [70] and neural networks [13–15, 71]. Poisoning attacks, which occur during the training phase, have also been proposed for a variety of classifiers and generative models [72–76]. Attacks have also been proposed on policies for reinforcement learning [77] and models with structured prediction outputs [78]. All the evasion attacks described above are for the virtual white-box setting. Black-box attacks on real-world models deployed by MlaaS providers have been proposed using the phenomenon of transferability [13, 27, 35]. Since these attacks require access to a representative training dataset which may not be available to the adversary in certain situations, black-box attacks which depend purely on queries to the target model have also been demonstrated [24–26].

There has been little previous work on evasion attacks that work in a real-world setting. The first to explore the possibility that visual adversarial examples remain adversarial even when passed through cameras were Kurakin et al. [28]. They printed out adversarial examples generated using the Fast Gradient Sign attack and passed them through a camera to determine if the resulting image was still adversarial. This attack was restricted to the white-box setting and the effect of varying physical conditions was not taken into account. Sharif et al. [45]

investigated how face recognition systems could be fooled by having a subject wear eyeglasses with adversarial perturbations printed on them. While their attack results were encouraging, they did not rigorously account for varying physical conditions and only took multiple pictures of a subject's face to increase the robustness of the attack. Further, they only tested their black-box attacks in a virtual setting with query access to the target model. Athalye et al. [42] introduced the concept of Expectation over Transformations to automatically generate adversarial examples robust to varying physical conditions and carried out a preliminary indoor investigation of the method. Petit et al. [79] examine the susceptibility of the LIDAR and camera sensors in an autonomous car but their attacks are unable to cause targeted misclassification.

Concurrent work in a similar vein to ours is presented in a technical report by Evtimov et al. [32]. They have performed a preliminary investigation of the threat posed to autonomous cars by physical adversarial examples by printing these out and evaluating them in a drive-by test. They account for varying physical conditions while creating adversarial examples by starting from pictures of traffic signs taken under different conditions and performing a joint optimization over these. In comparison, our attack utilizing random image transformations only requires a single image or even a drawing of the target sign, instead of a large set of real photographs. This makes our approach much more scalable. Another distinction between our method and the one used by Evtimov et al. [32] is in the optimization formulation. They update the perturbation with average of the gradients of the objective function with respect to the transformed input space, but our method uses the direct end-to-end gradient of the objective function with respect to the original input space of images. We achieve this by exploiting gradient back-propagation from the transformed input space through the composition of differentiable transformation functions to the original space. This method yields a more accurate update for solving the optimization problem. To the best of our knowledge, we are the first to explore Sign Embedding attacks, Lenticular Printing attacks, black-box attacks in the physical setting as well as possible countermeasures using adversarial training.

7 CONCLUSION

In this paper, we have demonstrated a wide range of attacks on the computer vision subsystem of self-driving cars. Sign Embedding attacks allow an adversary to convert any sign or logo into into a targeted adversarial example, while the Adversarial Traffic Sign attack converts one traffic sign into another for a ML classifier with the addition of small perturbations. The Lenticular Printing attack moves beyond the paradigm of adversarial examples to create images that look different from varying heights, allowing an adversary to stealthily embed a potentially dangerous traffic sign into an innocuous one, with no access to the internals of the classifier. We demonstrated the effectiveness of our attacks in both virtual and real-world settings. We are the first to carry out black-box attacks in a real-world setting as well as to propose and evaluate possible countermeasures against physical realizations of adversarial examples. We have also provided a comprehensive analysis of future research directions which we hope will encourage further research into securing physically deployed machine learning systems.

REFERENCES

[1] A. Mosenia, S. Sur-Kolay, A. Raghunathan, and N. K. Jha. Caba: Continuous authentication based on bioaura. *IEEE Transactions on Computers*, 66(5):759–772, May 2017.

[2] SYMM Kung, M Mak, and S Lin. *Biometric authentication: a machine learning approach*. Prentice Hall Press, 2004.

[3] A. Mosenia, S. Sur-Kolay, A. Raghunathan, and N. K. Jha. Wearable medical sensor-based system design: A survey. *IEEE Transactions on Multi-Scale Computing Systems*, 3(2):124–138, 2017.

[4] A. M. Nia, M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha. Energy-efficient long-term continuous personal health monitoring. *IEEE Transactions on Multi-Scale Computing Systems*, 1(2):85–98, 2015.

[5] NVIDIA. Self driving vehicles development platform. http://www.nvidia.com/object/drive-px.html. Accessed: 2016-10-31.

[6] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[7] A. Mosenia, J. F. Bechara, T. Zhang, P. Mittal, and M. Chiang. Procmotive: Bringing programability and connectivity into isolated vehicles. *Accepted for publication in Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (IMWUT)*, 2018.

[8] A. Mosenia, X. Dai, P. Mittal, and N. Jha. Pinme: Tracking a smartphone user around the world. *IEEE Transactions on Multi-Scale Computing Systems*, PP(99):1–1, 2017.

[9] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha. Systematic poisoning attacks on and defenses for machine learning in healthcare. *IEEE Journal of Biomedical and Health Informatics*, 19(6):1893–1905, Nov 2015.

[10] Arsalan Mosenia and Niraj Jha. A comprehensive study of security of Internet of Things. *IEEE Trans. Emerging Topics in Computing*, 5(4):586–602, 2017.

[11] Battista Biggio, Igino Corona, Blaine Nelson, Benjamin IP Rubinstein, Davide Maiorca, Giorgio Fumera, Giorgio Giacinto, and Fabio Roli. Security evaluation of support vector machines in adversarial environments. In *Support Vector Machines Applications*, pages 105–153. Springer, 2014.

[12] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. *arXiv preprint arXiv:1511.04599*, 2015.

[13] 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.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[15] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[16] Clarifai | image & video recognition API. https://clarifai.com. Accessed: 2017-08-22.

[17] Bigml.com is machine learning made easy. https://bigml.com. Accessed: 2017-08-22.

[18] Vision API - image content analysis | Google cloud platform. https://cloud.google.com/vision/. Accessed: 2017-08-22.

[19] Watson visual recognition. https://www.ibm.com/watson/services/visual-recognition/. Accessed: 2017-10-27.

[20] Face ++ cognitive services - leading facial recognition technology. https://www.faceplusplus.com/. Accessed: 2018-02-06.

[21] Tesla. Autopilot | tesla. https://www.tesla.com/autopilot. Accessed: 2017-12-05.

[22] James Vincent. Apple's latest ai research explores the problem of mapping systems for self-driving cars. https://www.theverge.com/2017/11/22/16689810/apple-ai-research-self-driving-cars-autonomous. Accessed: 2017-12-10.

[23] Amy Liu. Clarifai featured hack: Block unwanted nudity in blog comments with disqus. https://goo.gl/TCCVrR, 2016. Accessed: 2017-08-22.

[24] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui 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*, pages 15–26. ACM, 2017.

[25] Arjun Nitin Bhagoji, Warren He, Bo Li, and Dawn Song. Exploring the space of black-box attacks on deep neural networks. *arXiv preprint arXiv:1712.09491*, 2017.

[26] Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *arXiv preprint arXiv:1712.04248*, 2017.

[27] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. In *Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*, 2017.

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

[29] 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. ACM, 2016.

[30] Alexander Hars. Forecasts| driverless cars. http://www.driverless-future.com/?page_id=384. Accessed: 2017-12-05.

[31] Phil LeBeau. The $7 trillion promise of self-driving vehicles. https://www.cnbc.com/2017/06/01/the-7-trillion-promise-of-self-driving-vehicles.html. Accessed: 2017-12-05.

[32] Ivan Evtimov, Kevin Eykholt, Earlence Fernandes, Tadayoshi Kohno, Bo Li, Atul Prakash, Amir Rahmati, and Dawn Song. Robust physical-world attacks on machine learning models. *CoRR*, abs/1707.08945, 2017.

[33] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction apis. In *USENIX Security*, 2016.

[34] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 506–519, New York, NY, USA, 2017. ACM.

[35] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *ICLR*, 2017.

[36] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[37] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 387–402. Springer, 2013.
[38] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.
[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.
[40] 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.
[41] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.
[42] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017.
[43] Image segmentation — skimage v0.14dev docs. http://scikit-image.org/docs/dev/user_guide/tutorial_segmentation.html. (Accessed on 12/12/2017).
[44] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[45] 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*, CCS '16, pages 1528–1540, New York, NY, USA, 2016. ACM.
[46] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *arXiv preprint arXiv:1502.02590*, 2015.
[47] freesuperflip. http://www.vuethru.com/freesuperflip.html. (Accessed on 02/13/2018).
[48] Vincenzo Barrile, Giuseppe M Meduri, and Domenico Cuzzocrea. Automatic recognition of road signs by hough transform: Road-gis. *Journal of Earth Science and Engineering*, 2(1), 2012.
[49] P. Yakimov and V. Fursov. Traffic signs detection and tracking using modified hough transform. In *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, volume 05, pages 22–28, July 2015.
[50] Miguel Ángel García-Garrido, Miguel Ángel Sotelo, and Ernesto Martín-Gorostiza. *Fast Road Sign Detection Using Hough Transform for Assisted Driving of Road Vehicles*, pages 543–548. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.
[51] Hough circle transform — opencv-python tutorials 1 documentation. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghcircles/py_houghcircles.html. (Accessed on 12/11/2017).
[52] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, Nov 1986.
[53] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks*, pages 2809–2813, July 2011.
[54] Traffic signs classification with a convolutional network - alex staravoitau's blog. https://navoshta.com/traffic-signs-classification/. (Accessed on 12/12/2017).
[55] François Chollet et al. Keras. https://github.com/keras-team/keras, 2015.
[56] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
[57] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 2013.
[58] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, Jun 2010.
[59] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 2012.
[60] A. Mogelmose, M. M. Trivedi, and T. 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, Dec 2012.
[61] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv:1706.06083 [cs, stat]*, June 2017.
[62] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
[63] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[64] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. Standard detectors aren't (currently) fooled by physical adversarial stop signs. *arXiv preprint arXiv:1710.03337*, 2017.
[65] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *International Conference on Computer Vision. IEEE*, 2017.
[66] Jiajun Lu, Hussein Sibai, and Evan Fabry. Adversarial examples that fool detectors. *arXiv preprint arXiv:1712.02494*, 2017.
[67] Fernando Mujica. Scalable electronics driving autonomous vehicle technologies. *Texas Instrument*, 2014.
[68] Pin-Yu Chen, Yash Sharma, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. Ead: elastic-net attacks to deep neural networks via adversarial examples. *arXiv preprint arXiv:1709.04114*, 2017.
[69] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.
[70] Alex Kantchelian, JD Tygar, and Anthony D Joseph. Evasion and hardening of tree ensemble classifiers. In *Proceedings of the 33rd International Conference on Machine Learning (ICML-16)*, 2016.
[71] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.
[72] Battista Biggio, Blaine Nelson, and Pavel Laskov. Poisoning attacks against support vector machines. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1807–1814, 2012.
[73] Benjamin IP Rubinstein, Blaine Nelson, Ling Huang, Anthony D Joseph, Shing-hon Lau, Satish Rao, Nina Taft, and JD Tygar. Stealthy poisoning attacks on pca-based anomaly detectors. *ACM SIGMETRICS Performance Evaluation Review*, 37(2):73–74, 2009.
[74] Huang Xiao, Battista Biggio, Gavin Brown, Giorgio Fumera, Claudia Eckert, and Fabio Roli. Is feature selection secure against training data poisoning? In *ICML*, 2015.
[75] Shike Mei and Xiaojin Zhu. The security of latent dirichlet allocation. In *AISTATS*, 2015.
[76] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Data poisoning attacks against autoregressive models. In *AAAI*, 2016.
[77] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.
[78] Moustapha Cisse, Yossi Adi, Natalia Neverova, and Joseph Keshet. Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*, 2017.
[79] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe*, 11:2015, 2015.