

GRAPHITE: Generating Automatic Physical Examples for Machine-Learning Attacks on Computer Vision Systems

Ryan Feng¹, Neal Mangaokar¹, Jiefeng Chen², Earlence Fernandes², Somesh Jha², Atul Prakash¹

¹University of Michigan, Ann Arbor, ²University of Wisconsin, Madison

¹{rtfeng, nealmgkr, aprakash}@umich.edu, ²{jiefeng, earlence, jha}@cs.wisc.edu

Abstract—This paper investigates an adversary’s ease of attack in generating adversarial examples for real-world scenarios. We address three key requirements for practical attacks for the real-world: 1) *automatically* constraining the size and shape of the attack so it can be applied with stickers, 2) *transform-robustness*, i.e., robustness of a attack to environmental *physical* variations such as viewpoint and lighting changes, and 3) supporting attacks in not only white-box, but also black-box *hard-label* scenarios, so that the adversary can attack proprietary models. In this work, we propose GRAPHITE, an efficient and general framework for generating attacks that satisfy the above three key requirements. GRAPHITE takes advantage of transform-robustness, a metric based on expectation over transforms (EoT), to *automatically* generate small masks and optimize with gradient-free optimization. GRAPHITE is also flexible as it can easily trade-off transform-robustness, perturbation size, and query count in black-box settings. On a GTSRB model in a hard-label black-box setting, we are able to find attacks on all possible 1,806 victim-target class pairs with averages of 77.8% transform-robustness, perturbation size of 16.63% of the victim images, and 126K queries per pair. For digital-only attacks where achieving transform-robustness is not a requirement, GRAPHITE is able to find successful small-patch attacks with an average of only 566 queries for 92.2% of victim-target pairs. GRAPHITE is also able to find successful attacks using perturbations that modify small areas of the input image against PatchGuard, a recently proposed defense against patch-based attacks.

Index Terms—adversarial examples, patch attacks, physical attacks, black-box attacks, graphite

1. Introduction

Machine learning (ML) models have had resounding success in several scenarios such as face and object recognition [1]–[5]. Therefore, such models are now a part of perception pipelines in cyber-physical systems like cars [6]–[8], UAVs [9], [10] and robots [11]. However, recent work has found that these models are vulnerable to subtly perturbed adversarial examples that cause misclassification [12]–[15]. While these early digital white-box attacks were useful in understanding model weaknesses, researchers are now considering how to make attacks more practical for adversaries to accomplish in the real world [16]–[21].

Practical, real-world attacks have three key requirements. The first is that they should *automatically* choose

small areas to perturb, so that they can be applied with stickers on existing physical objects. The second is that practical, real-world attacks should exhibit *transform-robustness*, i.e., be robust to *physical*-world effects such as viewpoint and lighting changes as measured by a metric based on expectation over transforms [16]. The final requirement is that the most powerful real-world attacks should only require *hard-label* access. This means that the adversary can succeed with only the final top-1 decision, which *must* be provided by a deployed ML model for it to be useful even if its internals are closed source and designed to be difficult to completely reconstruct due to proprietary formats, implementations, and stripped binaries [22]. The adversary should also be able to work with smaller query budgets when that is a constraint.

Regarding the first requirement, the art of *automatically* picking limited areas to perturb remains relatively unexplored. Existing patch attacks [17], [23]–[26] limit perturbations to small patches, but are restricted to pre-defined patch shapes. The adversary must also either search for a good location and size or optimize over an expectation of patch locations and sizes. This is inefficient and, in black-box settings, query-intensive. Existing work also shows that targeted attacks require larger patches [26]. RP₂ [18] uses masks to constrain the space of the attack, but the mask creation process requires some manual experimentation, making it harder to scale. The Carlini & Wagner ℓ_0 attack [12] generates arbitrary, small sets of pixels to perturb, but is designed for digital attacks, not for physical settings or where only hard-label access to a model is available (requirements 2 and 3).

Likewise, existing work on finding attacks in the hard-label case do not satisfy the first two requirements; instead, they find digital attacks that require tens or hundreds of thousands of queries [20], [21], [27], [28].

We generalize these principles and *combine all three requirements* for the first time, to our knowledge, creating a more practical and efficient real-world attack framework called GRAPHITE (Generating Robust Automatic PHysical Image Test Examples). Such an attack framework could be potentially useful for security testing of and improving future defenses. We formulate our automatic framework of physical attacks as a joint optimization problem, balancing the opposing goals of minimizing perturbation size (or “mask size”) and maximizing transform-robustness. More broadly, the framework is designed to find attacks that permit tradeoffs among different constraints of query budgets (in hard-label settings), mask size, and transform-robustness.

We first instantiate this framework in the white-box

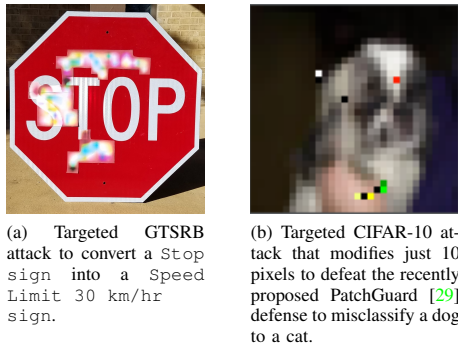


Figure 1. Example GTSRB and CIFAR-10 GRAPHITE-generated black-box hard-label attacks.

setting by adapting the Carlini-Wagner ℓ_0 attack [12] to help satisfy the three requirements. We show that the resulting attacks require a nearly 2x lower number of forward/backward passes compared to typical square patch attacks and exhibit more transform-robustness.

We then propose an instantiation of the framework for the more challenging hard-label setting. We show that there are significant algorithmic differences between the white-box and black-box attacks that are motivated by the general difficulties of naïve ℓ_0 style minimization attacks in the highly discrete and discontinuous hard-label optimization space. As a result, we show experimentally that simply extending white-box attacks to hard-label black-box settings using strategies such as OPT [20] yields poor results in Section 5.1. We solve these challenges by separating the joint optimization into a two-stage approximation: automatic mask generation and perturbation boosting, while using transform-robustness to provide a more continuous measure (as compared to just success or failure of an attack) to determine the relative value of keeping or removing pixels.

We evaluate our GRAPHITE framework in hard-label black-box settings both digitally on all 1,806 victim-target pairs of German Traffic Sign Recognition Benchmark (GTSRB) classes, and physically with printed stickers placed on a real Stop sign. We add an additional constraint in these experiments to restrict the attack to be contained within the victim object for physical-realizability. Digitally, we achieve an average transform-robustness of 77.8% and an average mask size of 16.63% of the victim image with 126k queries. Physically, we ran tests over 420 field test images of printed stickers placed on a real Stop sign. This includes 210 images of a Speed Limit 30 km/hr sign attack with a 95.7% attack success rate. We then demonstrate our framework’s ability to trade off mask size, transform-robustness, and query count. In one such setting, when transform-robustness is not required or low values suffice, GRAPHITE is able to find successful small-patch attacks for over 92.2% of GTSRB victim-target pairs with just 566 queries per successful pair. In contrast, state-of-the-art hard-label attacks of any type (including those bounded by ℓ_2 or ℓ_∞ norms) on similar datasets all report requiring tens to hundreds of thousands of queries, i.e., one to three orders of magnitude higher.

Finally, to demonstrate GRAPHITE’s ability to *thwart existing defenses*, we select a state-of-the-art patch attack

defense called PatchGuard [29] and attack a defended CIFAR-10 model in the hard-label black-box setting. Across 100 random victim-target image pairs, we are able to successfully attack PatchGuard by perturbing as little as 1% of the pixels (e.g., see Figure 1b). This also demonstrates the value of a framework that can generate more generalized forms of patch attacks (e.g., attacks using multiple patches) than those considered by PatchGuard; PatchGuard primarily evaluated its scheme on patches in a single part of an image.

The code is available at: <https://github.com/ryan-feng/GRAPHITE>.

Our Contributions:

- We proposed a general framework for generating practical, real-world attacks that combine three key requirements (that they are *automatically* space constrained, *physically* robust to transformations, and only require black-box *hard-label* access). The algorithms based on the framework generate attacks efficiently in both white-box and black-box settings.
- We introduce the first automatic, robust physical perturbation attacks in the black-box hard-label setting. We presented a novel mask generation process that *automatically* generates candidate masks to constrain the perturbation size and showed that transform-robustness is a sufficiently smooth function to optimize with gradient-free optimization. We test both digitally and physically, with printed stickers placed on physical objects.
- We demonstrated GRAPHITE’s ability to trade off transform-robustness, query budget, and mask size depending on the adversary’s limitations and priorities. As an example, we generate attacks with non-negligible transform-robustness in 92.2% of GTSRB victim-target pairs with an average of just 566 queries and an average mask size of 10.5%.

2. Related Work

Automatic Attacks. The art of constraining the size and shape of perturbations remains relatively unexplored. One approach to limit the perturbation size is to use a patch attack. Patch attacks were first proposed as a white-box technique that crafted small perturbation patches that could be added to any scene to cause a misclassification [17]. Later work then explored patch attacks on object detection [23], [24] and the soft-label setting [25], [26], where the model output probabilities are available. Oftentimes, the patches are made robust to different spatial locations by optimizing over different patch locations. However, these approaches must run inferences over many locations and are stuck to static shapes. In contrast, we are able to efficiently generate arbitrarily shaped perturbations and do so with only hard-label access, where only the top-1 class prediction is known.

Other work that limits the size of perturbation includes RP₂ [18], a physical white-box attack that designs adversarial stickers and the Carlini-Wagner (C&W) ℓ_0 attack [12]. RP₂ introduces the notion of a *mask* to limit the size of the perturbation. However, their process requires some manual experimentation based on the visual results

of an ℓ_1 approximation. The C&W ℓ_0 attack alternates between pixel removal and the C&W ℓ_2 attack to optimize an ℓ_0 , but is designed for digital white-box setting, failing to meet requirements 2 and 3.

Physical Attacks. Existing work for robust physical perturbation attacks on image models remains in the *white-box* setting. Examples include printing images of perturbed objects [30], modifying objects with stickers [18], [19], and 3D printing perturbed objects [16]. Such attacks typically optimize the expectation over a distribution of transformations, known as *Expectation over Transformation (EoT)* [16]. Such transformations model changes in viewing angle, distance, lighting, etc. In this work, we describe the expectation of attack success over such transformations as the *transform-robustness* of an attack, approximated by measuring the attack success rate with some number of transformations. Some attacks also introduce the notion of a mask that constrains the perturbation to small regions of the object [18]. However, existing work does not fully automatically generate suitable masks, and the mask is not jointly optimized with the noise.

In contrast, we automatically generate small masks to confine perturbations. We also extend our construction of such attacks to the black-box hard-label setting.

Black-box Attacks. Recent work has explored *digital* hard-label attacks [20], [21], [27], [28], [31], where only the top-1 label prediction is available. Such attacks are difficult because the hard-label optimization space is discontinuous and discrete. One approach is to reformulate the objective into a continuous optimization problem that finds the perturbation direction at which the decision boundary is closest. This approach is taken by OPT-attack [20], which uses the Randomized Gradient Free (RGF) method [32], [33] to optimize this distance-based objective. The boundary distance is calculated through a binary search process. The perturbation is initialized by interpolating with a target class image.

In our work, we generate physically robust examples within small, masked areas with only hard-label access. As we show in Section 5.1, directly adding EoT to this distance-based reformulation is both inefficient and ineffective, leaving visible artifacts of the intended target. Unlike OPT-attack, we also generate masks.

Other black-box work includes transfer and soft-label attacks. Transfer attacks train a surrogate model, generate white-box examples on the surrogate, and hope they transfer to the target [34]. Unfortunately, targeted examples often fail to transfer [35]. Many techniques also require access to similar training sets that may not be available, whereas our work only requires query access to the target model. Soft-label attacks, or *score-based* attacks, require access to the softmax layer output in addition to class labels [36], [37]. In contrast, our threat model only allows the adversary top-1 predicted class label access.

3. Setting up Automatic Physical Attacks

In this section, we lay the groundwork for our general joint optimization problem for automatic physical attacks. We define automatic physical attacks to be ones that: 1) automatically pick a small mask consisting of pixels to perturb and 2) are robust to some set of transformations, set to model environmental variation such as viewpoint

and lighting changes. We describe this joint optimization problem as the ideal problem we strive for and will provide solvers for in Sections 4 and 5. We also show that mask generation is an NP-complete problem.

3.1. Problem Setup

Our goal is to find some perturbation δ and a small mask M such that when δ is applied to a victim image x in the area defined by mask M , our model F predicts the target label y_{tar} with high transform-robustness. Transform-robustness is estimated over t transforms sampled from a distribution of transforms T . We describe a targeted attack formulation but can easily adapt to untargeted.

3.2. Optimizing Mask and Transform-robustness

The optimal perturbation maximizes transform-robustness (i.e., EoT metric) while occupying only a small area of the object such as a traffic sign. M_{ij} is 1 if pixel at position (i, j) is perturbed and 0 otherwise. A way to model such constrained optimization problems is to model is as a joint optimization problem, which we give below. $\lambda > 0$ is the relative weight given to size of the mask M in the joint objective:

$$\operatorname{argmin}_{\delta, M} \lambda \cdot \|M\|_0 - \mathbb{E}_{t \sim T} \left[F \left(t(x + M \cdot \delta) \right) = y_{tar} \right] \quad (1)$$

where the second term is defined as *transform-robustness*, computed as an expectation of attack success rate. In practice, we estimate this with a set of randomly sampled transforms. We observe that directly solving this objective is generally challenging, especially in the hard-label setting as we see in Section 5. Furthermore, the problem of finding an optimal solution with a constrained mask size is NP-complete, as we prove in Appendix A. One approach to constructing such masks is to adopt an ℓ_0 “heuristic” (the ℓ_0 norm itself is fundamentally non-differentiable, so straightforward adaptations of gradient descent or OPT-attack cannot be used). As an example, JSMA [14] builds a mask by considering the pixels most “relevant” towards causing a misclassification and Carlini and Wagner’s ℓ_0 attack [12] reduces a mask by removing least “relevant” pixels; both approaches require doing a large number of forward passes over the model (e.g., a pass for each pixel) to find most relevant or least pixel, since the problem is non-differentiable. This forward-pass method works in the white-box setting but as further discussed in Section 5, does not work in the hard-label setting where logits are not available; changing a pixel is unlikely to change the classification result, making it challenging to identify most relevant or least relevant pixels.

3.3. General Algorithmic Pipeline

We describe the general GRAPHITE solver framework with pseudocode in Algorithm 1. As the ℓ_0 norm inherently makes (1) difficult to solve directly, we alternate between reducing the mask size and improving the attack with the set of perturbable pixels. After initializing a perturbation and gradient, the framework iteratively selects

TABLE 1. A SUMMARY OF CHARACTERISTICS BETWEEN ALL BASELINES AND GRAPHITE. WE ADAPT C&W ℓ_0 TO WHITE-BOX PHYSICAL ATTACKS AND TO BLACK-BOX PHYSICAL ATTACK BASELINES. UPON FINDING THESE BASELINES TO BE INSUFFICIENT, WE SPLIT THE JOINT OPTIMIZATION INTO A TWO-STEP OPTIMIZATION PROBLEM WITH A SINGLE ROUND OF MASK GENERATION AND BOOSTING.

Algorithm	Works in Hard-Label?	Uses EoT?	Pixel Removal Strategy	Internal Attack	Num. Pixel Removal + Attack Rounds
C&W ℓ_0 [12]			Gradient-based	C&W ℓ_2 [12]	Multiple
White-box GRAPHITE (Sec. 4)		✓	Gradient-based	EoT PGD	Multiple
ℓ_0 and OPT (Sec. 5.1.2)	✓		RGF-based	OPT [20]	Multiple
ℓ_0 and OPT w/ EoT (Sec. 5.1.3)	✓	✓	RGF-based	OPT w/ EoT	Multiple
ℓ_0 and Boosting (Sec. 5.1.4)	✓	✓	RGF-based	Boosting (Sec. 5.2.3)	Multiple
Black-box GRAPHITE (Sec. 5.2)	✓	✓	Mask Gen. (Sec. 5.2.2)	Boosting (Sec. 5.2.3)	1

Algorithm 1 General GRAPHITE Framework

Input: Victim Image x , Target Image x_{tar} , Initial Mask M_{init} , Model F , Target Label y_{tar}

Output: Attacked Image A , Mask M , Perturbation δ

- 1: $M \leftarrow M_{init}$
- 2: $\delta, g \leftarrow \text{INIT_PERT_+_GRAD}(x, x_{tar}, M, F, y_{tar})$
- 3: **while** not done **do**
- 4: $S \leftarrow \text{SELECT_PIXELS}(x, x_{tar}, M, \delta, y_{tar}, g)$
- 5: $M \leftarrow \text{REMOVE_PIXELS}(M, S)$
- 6: $A, \delta, g \leftarrow \text{ATTACK}(x, x_{tar}, M, \delta_{init}, F, y_{tar})$
- 7: $A, \delta \leftarrow$ Last Successful Attack

pixels to remove, removes them, and attacks the remaining pixels. This process is repeated until a specified stopping criteria is met.

The C&W ℓ_0 attack is a natural starting point for our framework as it alternates between removing the pixel with the least impact and attacking the remaining pixels with C&W ℓ_2 until it can no longer find an adversarial example (Algorithm 2). Impact is measured by multiplying the gradient and the perturbation. We are able to adapt the C&W ℓ_0 attack to incorporate transform-robustness for our white-box GRAPHITE algorithm (Section 4).

We first consider simple adaptations of the C&W ℓ_0 attack, such as combining it with variations of OPT [20] and EOT [16], but find that they fail to find satisfactory hard-label physical attacks (Section 5.1). We consider these adaptations (listed in rows 3-5 of Table 3) as "baselines" for our algorithm. This then motivates us to separate the optimization into a mask generation phase and perturbation boosting phase, which leads to our novel hard-label black-box GRAPHITE algorithm (Section 5.2). Table 1 summarizes our algorithms and baselines.

4. White-box Automatic Physical Attacks

We now create a white-box GRAPHITE algorithm as an instantiation of the general GRAPHITE framework (Algorithm 1). We do so by extending the Carlini-Wagner ℓ_0 attack [12] to have high transform-robustness, creating an automatic, physical attack in the white-box setting.

White-box Attack Algorithm. We extend the C&W ℓ_0 attack to add the notion of transform-robustness by replacing the inner C&W ℓ_2 attack with an EoT PGD attack. By using an inner attack that applies EoT [16], we are able to generate attacks that have higher transform-robustness. Then, replacing the C&W ℓ_2 minimization attack with PGD [38] enables us to increase efficiency and align our attack more similarly to our later black-

Algorithm 2 Original C&W ℓ_0 Attack

Input: Victim Image x , Model F , Initial Mask M_{init} , Target Label y_{tar}

Output: Attacked Image A , Mask M , Perturbation δ

- 1: $M \leftarrow M_{init}$
- 2: $A, \delta, g \leftarrow \text{C\&W_}\ell_2_ATTACK(x, M, F, y_{tar})$
- 3: **while** $F(A) = y_{tar}$ **do**
- 4: $S \leftarrow \{\text{argmin}_i g_i \cdot \delta_i\}$
- 5: $M_{\{Pixels \in S\}} \leftarrow 0$
- 6: $A, \delta, g \leftarrow \text{C\&W_}\ell_2_ATTACK(x, M, F, y_{tar})$
- 7: $A, \delta \leftarrow$ Last Successful Attack

Algorithm 3 White-box GRAPHITE Attack

Input: Victim Image x , Model F , Initial Mask M_{init} , Target Label y_{tar}

Output: Attacked Image A , Mask M , Perturbation δ

- 1: $M \leftarrow M_{init}$
- 2: $A, \delta, g \leftarrow \text{EoT_PGD_ATTACK}(x, M, F, y_{tar})$
- 3: **while** Transform-Robustness of $A > tr_{min}$ **do**
- 4: $S \leftarrow$ set of z patches with smallest $g_i \cdot \delta_i$ values
- 5: $M_{\{Patches \in S\}} \leftarrow 0$
- 6: $A, \delta, g \leftarrow \text{EoT_PGD_ATTACK}(x, M, F, y_{tar})$
- 7: $A, \delta \leftarrow$ Last Attack with transform-robustness $> tr_{min}$ if one was found, else first attack

box implementation. With these changes, we now set the stopping criteria as our EoT PGD attack being unable to find an attack with at least tr_{min} transform-robustness.

Additionally, for further efficiency and to encourage larger sticker patches (for printing convenience), we remove z patches of pixels at a time, where each patch is a $p \times p$ square of pixels. To collect the list of patches, we stride by a step size of s .

Experimental Setup. To test our attack, we run targeted attacks for all victim-target pairs in a varied, 10 class subset of GTSRB: Stop, Speed Limit 30 km / hr, Speed Limit 80 km / hr, Pedestrians, Turn Left Ahead, Yield, Caution, Roundabout, End of Overtaking Limit, Do Not Enter. Foreshadowing our black-box attack, where we want to show dataset independence, we use images outside of the GTSRB dataset. Since we had a real-world Stop Sign available in our lab, we used a photo we took of that physical sign. For all other GTSRB classes, we attack traffic-sign images downloaded from the Internet. The images are 244×244 . We set the minimum transform-robustness threshold tr_{min} to 80%. Additional details on hyperparameters are included in Appendix C.

TABLE 2. WHITE-BOX COMPARISONS BETWEEN OUR WHITE-BOX VERSION OF GRAPHITE (ALGORITHM 3) AND A PATCH-PGD ATTACK THAT TRIES THE 4 CORNERS AND THE CENTER ON 90 VICTIM-TARGET TRAFFIC SIGN PAIRS. WHILE PATCH-PGD CAN PERFORM WELL ON THE RIGHT EXAMPLES, GRAPHITE CAN MUCH MORE RELIABLY AND QUICKLY FIND SUCCESSFUL ATTACKS WITH LESS PERTURBATION SIZE. TR STANDS FOR TRANSFORM-ROBUSTNESS. GRAY COLUMNS REPORT STATISTICS OVER JUST THE SAMPLES WITH $> 80\%$ TR.

Method	Avg. TR, All	Avg. Mask Size, All	Num. Samples with $>80\%$ TR	Avg. TR over Samples with $>80\%$ TR	Avg. Mask Size over Samples with $>80\%$ TR	Avg. Inferences
GRAPHITE	77.53%	8.82%	84	80.60%	6.35%	40,475.6
Patch-PGD	10.1556%	6.88%	4	95.25%	6.88%	100,000

Patch-PGD. For comparisons sake, we also compared against the popular square patch attack that simply runs Patch-PGD [29], [39] (with EoT) attack over a square patch of 64×64 pixels (about 6.88% of the image). Rather than test every possible location, which is expensive, we test the four corners (as in the weaker adversary in [39]) and the center. We ran it with 100 transforms, 200 steps, and a step size of 4 / 255.

Results. Table 2 presents the results from our white-box algorithm and the Patch-PGD baseline on all 90 victim-target pairs of the GTSRB subset. We also report results on the images with $> 80\%$ transform-robustness (the gray columns) to characterize the white-box GRAPHITE attacks that reached the tr_{min} threshold.

Our approach finds patches with higher transform-robustness more reliably and efficiently than choosing static squares, even when using pre-selected patch locations. Patch-PGD finds attacks with low transform-robustness (10.1556% on average, only 4 samples with $> 80\%$) and a high number of inferences (100k). In contrast, our white-box GRAPHITE algorithm averages 77.53% transform-robustness, including 84 samples with $> 80\%$. It also uses fewer inferences and achieves a smaller mask size over the successful samples (6.35% for GRAPHITE vs. 6.88% for Patch-PGD. Finally, our white-box algorithm does not require a user to specify the shape or size of the mask or its location. Example images of these attacks are included in Appendix C.

5. Black-box (Hard-label) Automatic Physical Attacks

We now create a new hard-label GRAPHITE algorithm as an instantiation of the general GRAPHITE framework (Algorithm 1) to generate automatic physical attacks in the hard-label setting. Our white-box algorithm (Section 4) cannot be trivially extended here, as our prior selection of pixels for removal required gradient access.

We first attempt to leverage the C&W ℓ_0 algorithm in combination with existing digital black-box attacks, with three new “baseline” black-box instantiations of GRAPHITE (Section 5.1). However, we find these “baselines” to be unsatisfactory (see the results in Section 5.1).

To address these limitations, we split the joint optimization problem in 1 into a two-step optimization instead (Section 5.2.1). Within this two-step process, we then (a) present a novel mask generation algorithm (Section 5.2.2) for the pixel selection and removal procedures (lines 4 and 5 of Algorithm 1) and (b) adopt Randomized Gradient Free (RGF) optimization of transform-robustness itself as the attack procedure (line 6 of Algorithm 1), referred to as “perturbation boosting” (Section 5.2.3).

5.1. Baselines

These baselines modify the C&W ℓ_0 paradigm of alternating pixel removal and attacking with each baseline differing only in terms of the inner attack choice. For our first two baselines, we adopt a vanilla and EoT version of the digital black-box OPT-attack [20]. OPT-attack is able to craft attacks in the black-box setting and provide estimated gradients that can be used to order pixels for pixel selection and removal. For our third baseline, we deviate from OPT-attack and instead adopt perturbation boosting — our RGF optimization attack that leverages transform-robustness itself as the objective function.

5.1.1. Setup. To evaluate our baselines in this section, we run targeted attacks for all 90 victim-target pairs in the 10 class subset of GTSRB used in Section 4. All hyper-parameters are identical to those used in our white-box experiments, and performed on 32×32 images. We also initialize the perturbation direction θ as the difference between the target and victim images.

5.1.2. ℓ_0 and OPT baseline. This baseline employs the vanilla OPT-attack [20] as its choice of attack algorithm. OPT-attack crafts an adversarial example using a distance-based objective:

$$\operatorname{argmin}_{\theta} g(\theta) \quad (2)$$

where the objective $g(\theta)$ is the distance to the decision boundary, formally defined as:

$$\operatorname{argmin}_{\mu > 0} \mu \text{ s.t. } \left(F \left(x + \mu \frac{\theta}{\|\theta\|_2} \right) = y_{tar} \right) \quad (3)$$

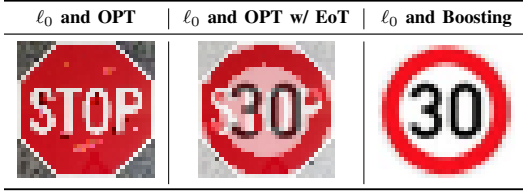
and where θ is the direction and μ is the distance to the nearest adversarial example x^* in that direction. We consider an attack a success so long as it elicits misclassification, regardless of transform-robustness (as transforms have not been incorporated in any manner).

Results. We find that OPT-attack is generally unable to find attacks with any significant transform-robustness (Table 3, Table 4). In particular, the average transform-robustness is only $\sim 1.5\%$. This is likely an artifact of (2) not incorporating transforms. OPT-attack also results in a significant number of queries ($\sim 920k$) because of its intense decision boundary-searching procedures. The final mask size is the only aspect of this baseline that is modestly respectable, with a perturbation that covers $\sim 24\%$ of the input image.

TABLE 3. QUANTITATIVE RESULTS FOR HARD-LABEL BASELINE ATTACKS. WE REPORT THE AVERAGE AND STANDARD DEVIATION OF TRANSFORM-ROBUSTNESS, # OF PIXELS OUT OF $32 \times 32 = 1024$ IN THE MASK (I.E., ℓ_0 DISTANCE), AND NUMBER OF QUERIES ISSUED TO THE MODEL. STATISTICS ARE PRESENTED ONLY FOR SUCCESSFUL ATTACKS (I.E., ATTACKS THAT ARE ABLE TO CRAFT AN ADVERSARIAL EXAMPLE).

Baseline	# Successes	Transform-robustness (%)	Queries	Mask Size
ℓ_0 and OPT	90/90	1.52 ± 4.26	$919.82k \pm 605.24k$	244.36 ± 307.11
ℓ_0 and OPT with EoT	81/90	80.01 ± 0.11	$1958.67k \pm 1317.49k$	601.90 ± 190.91
ℓ_0 and Boosting	81/90	97.89 ± 3.61	$5.50k \pm 9.17k$	990.72 ± 56.78

TABLE 4. SAMPLES FROM OUR THREE HARD-LABEL BASELINE ATTACKS. SHOWN ARE TARGETED ATTACKS FROM A STOP SIGN TO SPEED LIMIT 30 KM / HR SIGN. THE LEFTMOST IMAGE SUFFERS FROM EXTREMELY LOW TRANSFORM-ROBUSTNESS. THE MIDDLE AND RIGHTMOST IMAGES EXHIBIT HIGHER TRANSFORM-ROBUSTNESS AT THE COST OF EXTREME VISIBLE ARTIFACTS FROM THE TARGET IMAGE.



5.1.3. ℓ_0 and OPT with EoT baseline. This baseline employs an EoT version of OPT-attack [20] as its choice of attack algorithm, to alleviate the low transform-robustness of the first baseline. We first define a wrapper function $W(x)$:

$$W(x) = \begin{cases} y & \text{if } \mathbb{E}_{t \sim T} \left[F(t(x)) = y_{tar} \right] \geq 80\% \\ -1 & \text{otherwise} \end{cases} \quad (4)$$

Then, with this wrapper function, we modify $g(\theta)$ to the following:

$$\operatorname{argmin}_{\mu > 0} \mu \text{ s.t. } \left(W \left(x + \mu \frac{\theta}{\|\theta\|_2} \right) = y_{tar} \right) \quad (5)$$

We now consider an attack a success only if it exhibits a transform-robustness greater than the tr_{min} threshold, as this baseline directly incorporates transforms via EoT.

Results. We find that wrapping existing hard-label attacks with EoT is an inefficient approach to generating attacks with high transform-robustness (Table 3, Table 4). While, as expected, the introduction of EoT significantly raises average transform-robustness to $\sim 80\%$, it comes at the cost of doubling in queries $\sim 1,959k$.

Further, we find this approach yields attacks with unacceptably large target image artifacts. The average mask size covers nearly 59% of the input image. We also observe that even for the successful attacks, OPT attack is unable to escape the initial perturbation direction (i.e., the target image itself). This results in attack examples with very visible artifacts from the target.

5.1.4. ℓ_0 and Boosting Baseline. This baseline employs perturbation boosting as its choice of attack algorithm, i.e., an RGF optimization attack that leverages transform-robustness itself as the objective function. This choice is motivated by the intuition that directly optimizing for our desired metric (instead of a proxy objective function provided by existing black-box attacks) is likely a more

straightforward optimization problem, which is likely to result in fewer queries and a smaller mask, while still achieving high transform-robustness. For further details regarding the details and suitability of perturbation boosting for GRAPHITE, see Section 5.2.3, where it becomes a part of our final, recommended instantiation algorithm for GRAPHITE in the black-box setting.

Results. We find that this baseline yields transform-robust attacks with unacceptably large masks (Table 3, Table 4). While this baseline achieves a near optimal $\sim 98\%$ transform-robustness as expected (since we are now directly maximizing transform-robustness), the mask sizes cover nearly 97% of the input image.

Close inspection reveals that this is because the estimated RGF optimization gradients eventually become zero at some iteration (i.e., the area around the example appears to be flat, thereby obstructing sampling-based based techniques like RGF). Zero gradients prevent the algorithm from creating a reliable importance ordering of pixels, thereby ending the attack as pixel selection/removal cannot proceed. This also results in several artifacts from the target image, as 1-2 rounds are not enough to create a sufficiently small mask or produce a more obscure perturbation. These early exits also explain how this baseline is able to operate with significantly fewer queries than previous baselines ($\sim 5.5k$).

5.2. Black-box GRAPHITE Algorithm

Our black-box baseline instantiations in Section 5.1 failed to craft acceptable automatic physical attacks. However, these failures provided key insights towards developing automatic physical attacks: (a) creating high transform-robustness examples requires direct incorporation of transforms (EoT) into the attack, (b) naïve application of EoT to existing black-box attacks is an indirect and query-inefficient approach to doing so, and (c) reliance upon leveraging gradient-based pixel ordering can fail in the black-box setting, thus leading to large masks.

We instead solve a *two-step optimization problem* that separates out mask generation and boosting (Section 5.2.1). While the C&W ℓ_0 framework of alternating between pixel removal and attacking is able to capture a joint optimization in the white-box setting, our baselines showed that jointly optimizing the mask and the perturbation in the hard-label setting is difficult without gradients. Thus, we instead focus on generating a viable, small mask first before focusing on perturbation boosting. We then present algorithms for solving these optimization problems (Sections 5.2.2-5.2.3).

5.2.1. Two-Step Optimization. We now present a two-step optimization problem that first solves for an optimal mask given an initial perturbation and then second solves

Algorithm 4 Black-box GRAPHITE Attack

Input: Victim Image x , Target Image x_{tar} , Model F , Initial Mask M_{init} , Target Label y_{tar}

Output: Attacked Image A , Mask M , Perturbation δ

```

1:  $M \leftarrow M_{init}$ 
2:  $\delta \leftarrow M \cdot (x_{tar} - x)$ 
3: for One Iteration do
4:   // Select and Remove Patches (Mask Generation)
5:    $\mathcal{P} \leftarrow \text{HEATMAP\_ESTIMATION}(M, \delta, F)$ 
6:    $M, \mathcal{P} \leftarrow \text{COARSE\_REDUCTION}(M, \delta, F, \mathcal{P})$ 
7:    $M \leftarrow \text{FINE\_REDUCTION}(M, \delta, F, \mathcal{P})$ 
8:
9:   // Attack (Perturbation Boosting)
10:   $A, \delta \leftarrow \text{BOOST}(x, \delta, M, F, y_{tar})$ 

```

for the optimal perturbation given the discovered mask. This two-step problem can be viewed as heuristically solving the general framework presented in Algorithm 1 with a single iteration, with mask generation corresponding to pixel selection and removal and perturbation boosting becoming the internal attack. This solution then forms the basis for our primary, recommended hard-label instantiation of GRAPHITE, as shown in Algorithm 4.

Step #1: Mask Generation. We first find an optimized mask that, when filled in with the target image x_{tar} , results in at least a specified level of transform-robustness, tr_{lo} , a hyperparameter. We constrain the unmasked perturbation to be fixed at $\delta_{tar} = x_{tar} - x$ and solve the optimization objective from (1) under that constraint:

$$\begin{aligned} \underset{M}{\operatorname{argmin}} \quad & \lambda \cdot \|M\|_0 - \mathbb{E}_{t \sim T} \left[F \left(t(x + M \cdot \delta_{tar}) \right) = y_{tar} \right] \\ \text{s.t.} \quad & \mathbb{E}_{t \sim T} \left[F \left(t(x + M \cdot \delta_{tar}) \right) = y_{tar} \right] \geq tr_{lo} \end{aligned} \quad (6)$$

We show how to solve this formulation heuristically using an algorithm described in Section 5.2.2. Note that δ_{tar} only applies within the resulting mask M .

Step #2: Perturbation Boosting. Equipped with the mask M found by the Mask Generation Optimization Problem in Step #1, we then aim to maximize transform-robustness for the given mask by changing the perturbation δ within the masked region:

$$\underset{\delta}{\operatorname{argmax}} \quad \mathbb{E}_{t \sim T} \left[F \left(t(x + M \cdot \delta) \right) = y_{tar} \right] \quad (7)$$

By using transform-robustness as a measure of physical robustness, we can leverage this function as an optimization goal we can pursue even in the hard-label case (described in Section 5.2.3). In Section 5.2.3, we also provide empirical evidence that this reformulation based on transform-robustness is approximately continuous. We do this by showing that the objective has relatively low Lipschitz constants even when approximating the expectation in (7) by averaging over n transformations. We additionally refer to this process as *boosting* the transform-robustness, as it aims to make transform-robustness as high as possible.

5.2.2. Mask Generation Algorithm. For the first step of our hard-label GRAPHITE algorithm, we need to find a

Algorithm 5 Mask Generation. Let $TR(M, \delta)$ be the estimate of transform-robustness for a mask M and perturbation δ , and λ_1 be a hyperparameter, let $\mathcal{P}$ be the set of all patches, let M be the mask covering the whole object, and let $\delta_{tar} = x_{tar} - x$.

Input: Victim Image x , Target Image x_{tar} , Target Label y_{tar} , Optional Max Mask Size m_{max}

Output: Mask M with lowest mask score $J(M, \delta, tr_{lo})$

```

1: function  $J(M, \delta, tr_{lo})$ 
2:   if  $TR(M, \delta) < tr_{lo}$  then
3:     return  $\infty$ 
4:   return  $\lambda_1 \cdot \|M\|_0 + (1 - TR(M, \delta_{tar}))$ 
5:
6: // HEATMAP ESTIMATION
7: for  $\rho$  in  $\mathcal{P}$  do
8:    $tr_{\rho} \leftarrow TR(M - \rho, \delta_{tar})$ 
9: Sort  $\mathcal{P}$  from highest  $tr_{\rho}$  to lowest
10:
11: // COARSE REDUCTION
12:  $i \leftarrow$  Binary search for highest index such that
    $TR(\bigcup_{i=pivot}^{|\mathcal{P}|} \mathcal{P}_i), \delta_{tar}) \geq tr_{hi}$ 
13:  $L_c \leftarrow \{\mathcal{P}[i], \mathcal{P}[i+1], \dots, \mathcal{P}[|\mathcal{P}|-1]\}$ 
14:  $M \leftarrow \bigcup_{i=pivot}^{|\mathcal{P}|} \mathcal{P}_i$ 
15:
16: // FINE REDUCTION
17: for  $\rho$  in  $L_c$  do
18:   if  $J(M - \rho, \delta_{tar}, tr_{lo}) < J(M, \delta_{tar}, tr_{lo})$  then
19:      $M \leftarrow M - \rho$ 
20:   if  $m_{max}$  is provided and  $\|M\|_0 < m_{max}$  then
21:     break
22: if  $m_{max}$  is provided and  $\|M\|_0 > m_{max}$  then
23:   Increase  $\lambda_1$ ; Goto 24
24: return  $M$ 

```

candidate mask that has good initial transform-robustness and small size. Boosting can take care of further improving the transform-robustness as long as it samples enough variance at nearby points to make useful gradient estimates with RGF. We first set the perturbation to the target image’s pixels within the mask. In other words, the initial attack is $x + M \cdot (x_{tar} - x)$.

We generate masks using a three-stage process:

- 1) Heatmap Estimation
- 2) Coarse-grained Mask Reduction
- 3) Fine-grained Mask Reduction

The algorithm steps are shown in Algorithm 5. We first initialize the mask to the entire victim object, which overlays the target object over the victim. We then choose a group of pixels called a *patch* as a configurable input into the algorithm. We then collect all patches of a given shape (e.g. 4×4 squares) that overlap with the victim object. These patches serve two purposes: (1) it helps us estimate a “heatmap” (i.e., which regions of pixels contribute more to transform-robustness) and (2) it is used to remove groups of pixels from the mask to reduce its size as the algorithm optimizes the mask objective function (6). The shape size such as 4×4 square (rather than a single pixel) helps ensure that the attack image is not too pixelated

and thus more likely to be implementable as a physical perturbation such as via stickers.

This process can be visualized by imagining a thin overlay of the target being placed on top of the victim, and then slowly cutting regions out of the overlay, exposing the victim object underneath.

Heatmap Estimation. We generate a “heatmap” (similar to a saliency map) over the victim image to begin pixel selection and removal. This could be any process that generates an ordering of pixel patches. We mainly focus on one strategy that we call a “target-based heatmap” strategy, which orders patches by the transform-robustness with respect to the target.

Specifically, for each patch p that overlaps with the victim object, we measure the transform-robustness of the mask M and perturbation δ_{tar} , where M includes every pixel of the victim object except for p . In other words, we take the original target image overlay, cut out p , and measure the resulting transform-robustness. If the transform-robustness drops a lot relative to the transform-robustness with the patch after removing p , those target pixels are important to causing a target prediction. This enables us to identify candidate regions to remove from the mask. We output the sorted list of patches in decreasing order of transform-robustness.

The size of the patches matters in the hard-label setting. If we pick just a single pixel as our patch, the transform-robustness would likely be the same with or without it, making the heatmap useless. If we make the patch too large, we lose the ability to make more general mask shapes. For 32×32 noise, we empirically found that a 4×4 square patch worked well.

The “target-based heatmap” approach can successfully evaluate patches with varying degrees of transform-robustness, leading to a useful ordering. One could imagine a “victim-based heatmap” where we add small sections of the target onto the victim, calculating the transform-robustness with respect to the victim, but we found this approach to be sub-optimal. Our intuition is that this is a less direct measure in the targeted case - classifications of the target versus any other non-victim labels are treated the same. Note that in a “victim-based” setting we cannot practically compute transform-robustness with respect to the target since the target signal is typically too small to get anything other than $\approx 0\%$ transform-robustness. We try a random heatmap strategy in Section 6.3, where we simply order the patches randomly and pass the list on to fine-grained reduction.

Coarse-grained Mask Reduction. Using the sorted list of patches from heatmap estimation, we begin reducing the size of the mask. As an optimization to save queries, we first do a coarse-grained reduction that binary searches for a pivot in the patch list. We find the point $pivot$ in the ordered list of patches $\mathcal{P}$ such that the bitwise unions of patches from that point ($\bigcup_{i=pivot}^{|\mathcal{P}|} \mathcal{P}_i$) yields a mask of transform-robustness $\geq tr_{hi}$, where tr_{hi} is a hyper-parameter specifying a high transform-robustness threshold that must be reached. If tr_{hi} cannot be reached, we simply include all patches. Coarse-grained reduction outputs L_c , the list of patches from $pivot$ to the end.

Fine-grained Mask Reduction. We then use a greedy fine-grained reduction algorithm to further improve the objective function of (6). Fine-grained reduction takes L_c

and evaluates each patch in sorted order. It removes the patch if that improves the objective function, keeping it otherwise. Empirically, we found that it was important to add the restriction that transform-robustness does not cross below some minimum threshold, tr_{lo} to ensure success in boosting. The end result is the final mask defined by the union of the retained elements of L_c . Because the objective function rewards small masks and high transform-robustness, the final mask balances both well.

We can optionally specify to GRAPHITE that we desire a maximum mask size m_{max} . If this option is activated, we can simply break early once we drop below m_{max} . If the first iteration fails to get below m_{max} , then the value of λ (the weight of the mask size term in the joint optimization problem in (1)) can be increased until an iteration gets below m_{max} .

Let n be the number of transforms used for estimating transform-robustness. Heatmap estimation uses $n \cdot |\mathcal{P}|$ queries, coarse-grained reduction uses $n \cdot \log |\mathcal{P}|$ queries, and fine-grained reduction uses $n \cdot |L_c|$ queries.

5.2.3. Perturbation Boosting Algorithm.. Given a resulting image x and a mask M from the previous stage, transform-robustness boosting, or simply *boosting*, aims to find the perturbation δ that maximizes transform-robustness. We propose a *transform-robustness-based* reformulation to use with RGF [32], [33] to find the perturbation within the mask M with maximum transform-robustness, which works as follows.

The perturbation δ is first initialized to $M \cdot (x_{tar} - x)$ and then we proceed to maximize the probability that a perturbation remains robust to physical-world transforms with the RGF [32], [33] method for gradient estimation using q random samples for each gradient estimation. Explicitly, let u be random Gaussian unit vectors within the allowable range of the mask and let β be a nonzero smoothing parameter. Then, we set δ to $\delta - \eta \cdot \hat{g}$ where η is the step size and $\hat{g}$ is the gradient, calculated as follows:

$$\hat{g} = \frac{TR(M, \delta + \beta u) - TR(M, \delta)}{\beta} \cdot u \quad (8)$$

where $TR(M, \delta)$ refers to the estimated transform-robustness for a mask M and a perturbation δ .

Finally, we introduce the notion of a query budget which limits the number of queries used in a particular stage of the algorithm. These parameters can be tuned to emphasize better mask generation or better boosting without a budget of queries if we want to limit the number of attack queries.

Suitability of Transform-Robustness Reformulation. To demonstrate that our reformulation based on transform-robustness is approximately continuous for successful RGF optimization, we show empirically that our objective has a low local Lipschitz constant. We execute boosting on attack examples from Stop sign to Speed Limit 30 km/hr, Stop sign to Pedestrians, and Stop sign to Turn Left Ahead with $n = 1000$ transformations and approximate the local Lipschitz constant every time we compute $TR(M, \delta + \beta \cdot u)$. The approximate local Lipschitz constant is given by $\frac{|TR(M, \delta + \beta \cdot u) - TR(M, \delta)|}{|\beta \cdot u|}$. We found that the maximum observed local Lipschitz constant was 0.056. We include a histogram of observed local Lipschitz constants in Fig. 2.

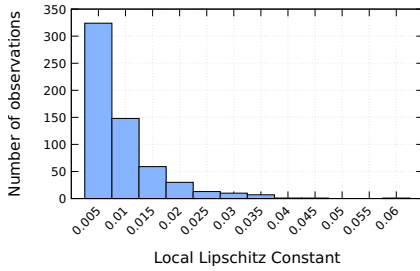


Figure 2. A histogram of our approximate local Lipschitz constant observations on transform-robustness. This graph shows that transform-robustness does not vary too much per amount of change with a sufficient number of transformations.

6. Experiments

In this section, we demonstrate the digital and physical viability of our black-box GRAPHITE instantiation presented in Algorithm 4 and Section 5.2, henceforth referred to as simply GRAPHITE unless otherwise specified. We first report digital transform-robustness results on GTSRB [40] and CIFAR-10 [41]. Then, we print stickers on a smaller subset of attacks, and conduct real-world field tests on physical traffic signs to show that our results carry over to the real-world. Finally, we explore our pipeline’s ability to adapt to attacker priorities/constraints, allowing for a trade-off between transform-robustness, mask size, and query budget.

6.1. Experimental Setup

This section discusses our experimental setup for running GRAPHITE. We will publicly release our code and data prior to paper publication.

Datasets and Classifiers. We use the classifier from RP_2 [18], [42] trained on an augmented GTSRB dataset [40]. As with RP_2 [18], we replace the German Stop signs with U.S. Stop signs from the LISA dataset [43]. As a validation set, we take out the last 10% from each class in the training set. We augment the dataset with random rotation, translation, and shear, following Eykholt *et al.* [18]. Our network, GTSRB-Net, has a 97.656% test set accuracy.

As in our white-box experiments in Section 4, we use Internet images outside of the dataset plus our own Stop sign picture since we had a physical sign available for initialization to demonstrate that GRAPHITE does not rely on having training set images to initialize from. In our experiments we assume that object boundaries are available but note they could be obtained automatically through an object segmentation network [44].

GTSRB Attack Details. We set the size of our input images to be 244×244 . During the attack, we generate 32×32 perturbations and then upsample the perturbations to the resolution of the input image when they are added. Reducing the dimensionality of the perturbation space makes RGF more efficient [20] and can help encourage blockier perturbations that can be sensed by a camera in the physical-world.

For field testing, we print stickers and place them on a $30'' \times 30''$ Stop sign and place it at stationary positions

TABLE 5. HYPERPARAMETERS FOR BLACK-BOX GRAPHITE EXPERIMENTS. ALL RANGED PARAMETERS SAMPLED UNIFORMLY.

Parameter	Value
RGF smoothing parameter (β)	1
RGF step size (η)	500
Boosting query budget (b)	20k
# Samples for RGF gradient sampling (q)	10
Rotation range about the y axis	$(-50^\circ, 50^\circ)$
Base focal length (f)	3 ft.
Crop sizes and offsets	$(-3.125\%, 3.125\%)$
Lighting gamma value (γ)	(1, 3.5) and (1, 1/3.5)
Gaussian kernels for blurring	[1, 3, 5, 7]
Joint optimization mask size weight (λ)	5
# of transforms (n)	100
Transform-robustness high threshold for coarse red. (tr_{hi})	85%
Transform-robustness low threshold for fine red. (tr_{lo})	65%
Max mask size option (m_{max})	Not set

TABLE 6. QUANTITATIVE RESULTS OF DIGITAL GTSRB ATTACKS. AVERAGE AND STANDARD DEVIATION OF TRANSFORM-ROBUSTNESS, # OF PIXELS OUT OF $32 \times 32 = 1024$ IN THE MASK (I.E., ℓ_0 DISTANCE), NUMBER OF QUERIES REPORTED FOR THE GIVEN VICTIM TO ALL OTHER TARGETS.

Victim	Stop	SL 30 km/hr	Avg. of all 43 victims
TR (%)	80.8 ± 7.02	84.5 ± 10.6	77.8 ± 17.5
Mask Size	116 ± 38.0	139 ± 71.7	170.3 ± 122
# Queries	$133k \pm 8.92k$	$134k \pm 15.2k$	$126k \pm 18.9k$

to test how robust our attacks are to different viewing conditions. We take five pictures of the perturbed Stop sign at 14 different locations for a total of 70 pictures per set. To test lighting conditions, we take one set of images in outdoor light and two sets indoors, one with indoor lights on and one without. To compare against baseline Stop sign accuracy, we also take five pictures of a clean Stop sign at each of the same 14 locations. The 14 locations were chosen based on RP_2 evaluation [18].

To gather crops, we use the original YOLOv3 [45] object detector network trained on MS COCO [46] to predict bounding boxes for the Stop sign. We take the output bounding boxes, crop the sign, resize to 32×32 , and feed through our network for classification. Further hyperparameters are specified in Table 5.

6.2. Experimental Results

We first report digital results for GTSRB and CIFAR-10. We then report physical transform-robustness field tests results on GTSRB¹. Finally, we report physical drive-by results on GTSRB.

Digital Transform-Robustness Results (GTSRB). We report results for all 1,806 possible GTSRB victim target pairs in Table 6. On average, we observe masks with an ℓ_0 distance of 170.3 (16.6% w.r.t. the 32×32 image area) and 77.8% transform-robustness.

In Table 7 we provide results from attacks between all victim target pairs of a subset of 10 varied GTSRB

1. CIFAR-10 objects such as airplanes or birds would have been difficult for applying physical perturbations.

TABLE 7. SAMPLE OF DIGITAL TARGETED ATTACKS ON GTSRB-NET. RAN WITH $tr_{lo} = 65\%$, $tr_{hi} = 85\%$, $n = 100$ TRANSFORMS. FOR CELLS WITH SAME VICTIM AND TARGET, WE REPORT THE % OF TRANSFORMS THE ORIGINAL LABEL IS PREDICTED. MASKS SIZE IS REPORTED IN TERMS OF # OF PIXELS IN 32×32 (I.E., ℓ_0 IN 32×32 SPACE).

Victim	Target									
Transform-robustness	92%	86%	85%	79%	80%	82%	80%	90%	77%	71%
Mask Size	0	88	123	105	51	96	131	92	138	97
Transform-robustness	77%	100%	88%	94%	94%	89%	88%	99%	61%	93%
Mask Size	161	0	18	140	74	152	175	45	210	156
Transform-robustness	79%	97%	88%	87%	88%	91%	95%	97%	62%	65%
Mask Size	171	13	0	228	132	173	162	41	213	122
Transform-robustness	50%	73%	52%	95%	86%	79%	100%	0%	79%	88%
Mask Size	398	398	398	0	198	254	28	398	205	190
Transform-robustness	73%	85%	71%	85%	100%	81%	71%	80%	71%	71%
Mask Size	111	72	120	92	0	122	93	74	263	148
Transform-robustness	0%	58%	7%	4%	92%	100%	46%	87%	83%	57%
Mask Size	436	436	436	436	81	0	436	157	198	436
Transform-robustness	56%	84%	70%	90%	80%	69%	100%	0%	78%	77%
Mask Size	398	398	398	23	133	225	0	398	133	161
Transform-robustness	74%	91%	73%	86%	77%	78%	76%	99%	72%	69%
Mask Size	172	43	75	92	62	106	133	0	218	114
Transform-robustness	72%	83%	82%	74%	91%	92%	78%	90%	73%	79%
Mask Size	142	123	141	159	137	137	175	90	0	182
Transform-robustness	79%	89%	89%	81%	82%	67%	82%	91%	73%	99%
Mask Size	42	85	79	106	106	153	117	62	242	0

signs. We include the final output image, the transform-robustness, and the mask size. GRAPHITE generally finds small, highly transform-robust perturbations on a variety of attacks. Examples that tend to not perform well are attacks on a triangle sign victim, as overcoming the difference in shape is difficult. Furthermore, it is not large enough to fit the entire target object on the victim. This means that in reduction the starting transform-robustness may be low to begin with, making it hard to prune patches from the mask. Resizing the target sign to fit within the victim sign may alleviate this. Generally, the attack quality varies depending on the victim target pair, which is consistent with prior work that finds that the distortion varies drastically depending on the attack pair [14].

Digital Transform-Robustness Results (CIFAR-10).

We test on 500 random victim-target pairs from CIFAR-10 [41] on Wide ResNet 34-10 [47], commonly used to evaluate CIFAR-10 attacks and defenses [38], [48], [49]. We observe that even natural images are not robust to the same set of transformations used in GTSRB, so we reduce the perspective transform to $[-30, 30]$ at a constant focal length, reduce γ to $(1, 2)$ and $(1, 1/2)$, and remove the blur. We achieve an average transform-robustness of 87.43% and mask size of 125.5 with 141.9k queries. We include example attacks in Appendix D.

Physical Transform-Robustness Field Tests (GTSRB). We also conduct field experiments to confirm that our results carry over to the physical world. This consists

TABLE 8. GTSRB FIELD TEST RESULTS. PHYSICAL ROBUSTNESS RESULTS ARE CALCULATED OVER 5 PICTURES EACH AT THE FOLLOWING SPOTS: 5 FT $\times$ $\{0^\circ, 15^\circ, 30^\circ, 45^\circ\}$, 10 FT $\times$ $\{0^\circ, 15^\circ, 30^\circ\}$, 15 FT $\times$ $\{0^\circ, 15^\circ\}$, 20 FT $\times$ $\{0^\circ, 15^\circ\}$, 25 FT, 30 FT, 40 FT. EACH EXAMPLE WAS TESTED 3 TIMES: OUTDOORS, INDOORS WITH INDOOR LIGHTS TURNED OFF, AND INDOORS WITH INDOOR LIGHTS TURNED ON.

Victim	Target	Digital GRAPHITE attack	Physical GRAPHITE attack (outdoors)	Dig. TR (100 xforms)	Phys. TR (Indoors, lights off)	Phys. TR (Indoors, lights on)	Phys. TR (Outdoors)
				86%	92.9%	94.3%	100%
				79%	97.1%	85.7%	100%

TABLE 9. EXTENDED GTSRB FIELD TEST RESULTS, INCLUDING THE MEASURED PHYSICAL ROBUSTNESS OF THE VIDEO FRAMES. EVERY 10TH FRAME OF THE VIDEOS WERE ANALYZED.

Target	# Frames Analyzed	Attack Success Rate (Transform-robustness)
Speed Limit 30 km/hr	40	97.5%
Pedestrians	39	82.1%

of printing stickers and placing them on real-world traffic signs and then evaluating images at differing conditions. We evaluate GRAPHITE on targeted Stop sign attacks to Speed Limit 30 km/hr and Pedestrians at different viewing angles and lighting.

Table 8 shows the results of the field tests for GTSRB. On average, they took 131k queries. The Speed Limit 30 km/hr attack was successful over all three lighting conditions with at least 92.9% physical robustness. The Pedestrians attack also did well, with at least 85.7% success rate over each lighting condition. Overall, these results suggest that transform-robustness translates reasonably well into the physical world and improvements in transformations could further improve the translation. Baseline Stop sign tests were at least 95.7% in each lighting condition.

Physical Drive-by Tests (GTSRB). In this section, we provide extended field test results for GTSRB under additional imaging conditions, in the form of drive-by tests. In particular, we record videos while driving towards the sign in a private lot, simulating a realistic driving environment in an allowable fashion². We test each of the two attacks from Table 8. As in prior work by Eykholt et al. [18], we analyze every 10th frame from the video. We use the same YOLOv3 [45] detector as the other GTSRB results to crop our frames. Some frames, particularly ones at the beginning of the video when we were farthest away could not be cropped with YOLOv3 and were thus cropped manually in a similar fashion.

We include results for these two stop sign attacks in Table 9. The Speed Limit 30 km/hr attack was the most successful, with 97.5% physical robustness, which is in line with the field test results in Table 8 and above the

digital transform-robustness. The Pedestrians attack was also successful with an 82.1% physical robustness rate, right around its digital transform-robustness of 79%.

Overall, these results confirm the field test results reported in Table 8 in an even more realistic driving setting. Example images are included in Appendix E.

6.3. Variations and Tuning of GRAPHITE

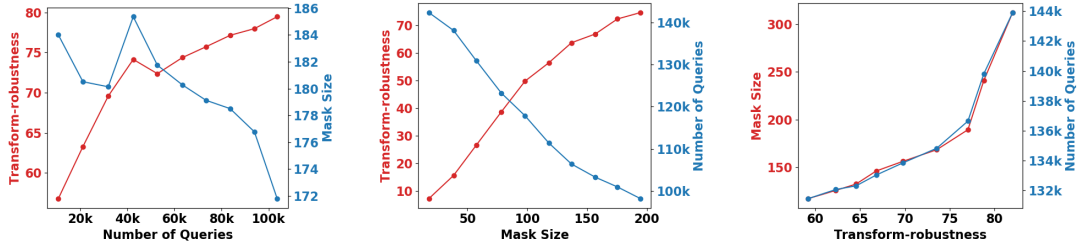
We now explore GRAPHITE’s ability to trade-off transform-robustness, mask size, and query count as well as an examination of heatmap estimation strategies. These tradeoffs allow an attacker to design their attack given the relative importance of certain attack constraints, such as a need to keep the perceptibility of such attacks low or a need to keep the query cost under a certain budget. We also examine GRAPHITE’s output without the restriction that the noise be contained within the victim object and a strategy with multiple mask generation / boosting rounds.

6.3.1. Reducing the Number of Queries. To further reduce the number of queries, we consider two adjustments: 1) reducing the number of transforms and 2) replacing the target-based heatmap strategy with a random heatmap strategy that simply orders patches randomly. We also test a “minimum query” setting, which uses both adjustments to the extreme: both the random heatmap and only one transform.

Tuning the Number of Transformations. We first begin by decreasing n (the number of transformations) that we use in mask generation and boosting. This makes every transform-robustness measurement run faster. We test with $n = 10, 20, 30, 40, 50, 60, 70, 80, 90, 100$ transforms. Regardless of the value of n , we measure the final transform-robustness of each setting with 100 transformations to ensure the numbers are comparative. We test the 90 non-identity victim-target pairs from Table 7. The results of these tests are shown in Fig. 3a. Since the number of queries scales monotonically with the number of transforms, we plot the 10 points using the number of queries on the x axis, and with the transform-robustness and mask size on the two y axes.

As more transforms are used in mask generation and boosting, we find that the quality of the generated attack increases, i.e., the size of the mask decreases

2. We follow local laws and regulations for safety in this test.



(a) Effects of tuning the number of transforms, on 90 victim-target pairs. Generally, as the number of queries and transformations are increased, the mask size decreases and the transform-robustness increases.

(b) Effects of tuning the max mask size option. As m_{max} increases, the transform-robustness increases and the query count decreases.

(c) Effects of tuning tr_{lo} . Generally, as tr_{lo} increases, the mask size increases as we reject more mask removals and the query count increases some as fewer overlapping patches are removed early.

Figure 3. Effects of trading off query count, mask size, and transform-robustness in our hard-label GRAPHITE pipeline.

TABLE 10. COMPARISON OF THE ORIGINAL TARGET-BASED HEATMAP VS. A RANDOM HEATMAP STRATEGY. THE NUMBER OF QUERIES CAN BE DRASTICALLY DECREASED WITH, ON AVERAGE, SOME INCREASE IN MASK SIZE AND SOME DECREASE IN TRANSFORM-ROBUSTNESS. AVERAGE AND STANDARD DEVIATION ARE PRESENTED.

Strategy	Orig. Strategy with Target Heatmap	Random Heatmap	Min Query Setting
TR (%)	77.8±17.5	76.4±17.1	19.9±21.2
Mask Size	170.3±122	175.9±121	112.993±69.7
# Queries	126k±18.9k	62.1k±12.5	572±105

and transform-robustness increases. Intuitively, this makes sense - as the number of transformations are increased, the objective becomes smoother, and the estimates become more accurate. This also explains why the data at points with fewer transformations are noisy within the relative scale of the plot, as the sampled transformations may represent the true distribution poorly. This performance benefit comes at the trade-off of increased query count and thus, if given a specific query budget, an attacker could tune this parameter based on their query restrictions.

Random Heatmap. For this variation, we consider replacing the target-based heatmap estimation process with a random heatmap strategy that simply orders the patches randomly, processing the patches in fine-grained reduction in an arbitrary order (note that we remove coarse-grained reduction, as its use of binary search makes it no longer well-defined). This significantly reduces the queries by saving an iteration over patch transform-robustness estimations. While we lose the ability to binary search, we save several thousand queries with this change. The objective function of (6) rejects the removal of extremely bad choice of patches but can suffer from poor removal choices early on that negatively influence the final result.

The results of this experiment are shown in Table 10, run over all 1,806 victim-target pairs. The random heatmap strategy performs quite similarly to the original target-based heatmap strategy with only about half of the queries, with the average number of queries dropping from 126k to 62.1k. The loss in attack quality is a drop from 77.8% to 76.4% transform-robustness and an increase from 170.3 to 175.9 pixels in terms of mask size. So,

random heatmap is a viable strategy when query count is a concern. On average, however, the target-based heatmap still outperforms in terms of transform-robustness and mask generation.

Min Queries Setting. We now combine both previous experiments by running GRAPHITE with the random heatmap and with only one transformation, to see if we can get a result that could possibly cause an accident in the right conditions with the minimum possible queries.

The results are shown in Table 10, run over all victim-target pairs. We find that, on average, GRAPHITE can find attacks with 19.9% transform-robustness using only 572 queries. This includes 1,665 / 1,806 victim-target pairs with greater than 0% transform-robustness. The mask size is also small, with an average of 112.993 pixels (11.0% of the image), with fewer transformations for mask generation to have to optimize for. On just the 1665 non-zero transform-robustness examples, the average mask size was 107.897 pixels (10.5% of the image) and the average query count was 566.253 queries. This suggests that risky examples could be generated in as few as a couple hundred queries but acquiring more robustness to a variety of transformations requires more queries.

6.3.2. Reducing the Mask Size. To reduce the mask size, we enable the m_{max} option and vary it to the desired maximum mask size. This option can help achieve a desired mask size limit or make a less perceptible attack. We also change $tr_{lo} = 0$ to ensure that the maximum mask size can be reached, even if it means choosing a mask with a transform-robustness of 0.

We test with m_{max} values of 20, 41, 61, 82, 102, 123, 143, 164, 184, and 205, which corresponds to 2%, 4%, ..., 20% of the image area. Since mask size increases monotonically as m_{max} increases, we plot the actual average mask size on the x axis. The results of these tests are shown in Fig. 3b. The results show an increase in transform-robustness as the maximum mask size increases (and thus the area to perturb increases) and a decrease in query count, as the algorithm can get below the maximum size quicker and exit earlier.

6.3.3. Increasing transform-robustness. To adjust GRAPHITE based on a desired transform-robustness level, we can tune the value of tr_{lo} . By increasing tr_{lo} ,

TABLE 11. COMPARISON OF THE ORIGINAL VICTIM-OBJECT CONSTRAINED ATTACK AND AN ATTACK WHERE THE PERTURBATION CAN BE PLACED ANYWHERE IN THE IMAGE. AVERAGE AND STANDARD DEVIATION ARE PRESENTED.

Strategy	Orig. Strategy, Restricted to Victim Object	Area Unrestricted
TR. (%)	77.8±17.5	86.7±8.81
Mask Size	170.3±122	147.1±77.8
# Queries	126k±18.9k	149k±20.6

we can achieve higher levels of transform-robustness but potentially at the cost of larger mask size. This is because tr_{lo} acts like a floor of acceptable transform-robustness while performing mask generation and guarantees a higher transform-robustness point on boosting. So, on average, a higher tr_{lo} should result in higher transform-robustness.

We test with tr_{lo} values of 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. We fix tr_{hi} to 1.0 for all values of tr_{lo} so that it can remain constant across the tests. The average observed transform-robustness increased monotonically with tr_{lo} , so we plot the actual transform-robustness on the x axis. The results of these tests are shown in Fig. 3c. The results show an increase in both mask size and number of queries as transform-robustness increases, as expected. As such, this parameter can help raise transform-robustness if the attack is not as successful as need be in the physical-world. Conversely, lowering tr_{lo} may help find smaller, less perceptible attacks.

6.3.4. Removing the Victim Object Boundary Restrictions. To examine GRAPHITE’s performance in a setting closer to standard patch attacks, we remove the restriction that the perturbation be contained to the victim object.

The results show that GRAPHITE can perform even better if it is not concerned with fitting it in within the boundary of the victim object. The average transform-robustness increases from 77.8% to 86.7% and the mask size drops from 170.3 pixels ($\approx 16.6\%$ of the image) to 147.1 pixels ($\approx 14.4\%$ of the image). With more valid patches to evaluate, the query count raises slightly from 126k to 149k. Thus, if the threat model does not need to remain strictly in victim object, smaller attacks can be found with higher transform-robustness.

6.3.5. Alternating Repeatedly between Mask Generation and Boosting. We now apply multiple rounds of mask generation and boosting as an approximation of a joint optimization approach, testing the 90 non-identity victim-target pairs from Table 7 on two rounds. We find that the average mask size improves from 172.0 to 138.9 pixels (a 19.3% improvement) but transform-robustness decreases from 75.3% to 74.2% and query count increases from 129.5k to 213.3k queries (a 64.7% increase). While this strategy can yield smaller masks, the query count increases faster than the performance improvements.

7. Discussion

We now discuss various countermeasures, how GRAPHITE performs against a defense called PatchGuard [29], alternative defense approaches, transformation limitations, license plate attacks, and ethical concerns.

7.1. Countermeasures

To the best of our knowledge, there are no defenses that directly aim to mitigate the effects of physical attacks that are realized through arbitrarily shaped perturbations. However, there is some recent work that attempt to address patch attacks (a similar attack where the perturbation is instead restricted to a single patch) [29], [39], such as PatchGuard [29], which we explore below. We discuss a few other possible strategies as well.

7.1.1. PatchGuard. Recent work has, however, attempted to address patch attacks (a similar attack where the perturbation is instead restricted to a single patch) [29], [39]. In this section, we investigate whether such defenses are sufficient to defend against attacks from GRAPHITE. To this end, we select PatchGuard [29], a state-of-the-art provably robust defense against patch attacks.

At a high-level, PatchGuard operates by leveraging a CNN with a “small receptive field”, i.e., a CNN where each feature is only influenced by small regions in the input image. This in turn suggests that patch attack perturbations (which occupy small regions in the input image) influence only a few features and are thus forced to produce abnormally large “suspicious” feature values to elicit misclassification (which can then be masked out). GRAPHITE, however, generates arbitrarily shaped perturbations. As such, even if the model has a small receptive field, many features can be influenced by the spread-out perturbations. Intuitively, this suggests that PatchGuard is largely ineffective at defending against GRAPHITE.

Results. We evaluate this intuition by launching several GRAPHITE attacks against PatchGuard. Specifically, we sample victim-target pairs from the CIFAR-10 dataset, and craft GRAPHITE attacks against the authors’ CIFAR-10 BagNet CNN implementation of the PatchGuard defense. We employ transforms adapted for the 32×32 CIFAR-10 image size and pare down specific transforms so that average transform-robustness of CIFAR-10 images themselves is over 50% (rotation about y axis is reduced to between -10° and 10° , lighting variation is removed, and blurring is reduced to kernels of [0,3]). Across 100 randomly sampled victim-target pairs, we are able to obtain an average transform-robustness of 68% using 155.8k queries with an average mask size of 193.81 pixels. This includes 33 examples with less than 102 pixels (10% of the image) with 77% transform-robustness.

Discussion. The above results suggest that current state-of-the-art defenses against patch attacks are not effective against GRAPHITE, and that future work is necessary to gain robustness against such attacks. One such possible modification to the PatchGuard defense is to mask out the top n regions (rather than that top-1 region as in the standard PatchGuard defense) that contribute to misclassification. One challenge is that it is unknown to the defender how many regions of perturbations GRAPHITE introduces and thus what is the appropriate value (or perhaps, thresholding) of n that would suffice. With higher values of n , the risk of false positives could increase as well. A future study would need to explore the defenses in depth and experimentally evaluate them against adaptive GRAPHITE-generated attacks against models that incor-

TABLE 12. ALPR FIELD TEST RESULTS. TRANSFORM-ROBUSTNESS RESULTS CALCULATED OVER 5 PICTURES EACH AT: {5 FT, 10 FT, 15 FT, 20 FT} $\times$ {0°, 15°}. LICENSE PLATES ARE EXPIRED PLATES WE ACQUIRED FOR THE PURPOSE OF TESTING THESE ATTACKS. ALL IMAGES CROPPED TO SHOW THE PLATE AREA IN LARGER DETAIL.

Digital GRAPHITE attack	Physical GRAPHITE attack	Phys. transform-robustness	Avg. Lev. Dist.
		82.5%	3.175
		67.5%	1.175

porate the above defense as part of their classification pipeline. We leave this to future work.

7.1.2. Alternative Defense Approaches. Our white-box or black-box GRAPHITE algorithm could potentially be used in *adversarial training* to gain robustness to such attacks. In particular, attacks could be tuned (Section 6.3) or in other ways to be more efficient and the parameterization of the GRAPHITE style pipeline enables the generation of multiple attacks per image. We leave exploration of the viability of such an approach to future work. A model of the types of shapes and sizes of perturbations that should be considered may also help.

It may also be possible to look for anomalous query behavior that is suspicious of a possible hard-label attack [50]. One difficulty of such a defense is that GRAPHITE’s transformation sampling naturally applies query blinding (e.g. the application of pre-processing transformation functions to hide the query sequence) as a byproduct. Prior work has shown that query blinding weakens this type of defense [50]. We leave in-depth analysis of the viability and anticipated next steps from the attacker and the defender to future work.

7.2. Transformations

One benefit of the GRAPHITE framework is its ability to accept arbitrary, parameterizable transformations in the hard-label setting, even if non-differentiable. We choose a set of transformations (perspective transformation, gamma correction, and blurring) inspired by prior work [16] but we could easily use new transformations that model different effects in the future.

One such case where this may be useful is with printing and lighting error. We noticed a limitation in gamma correction to model sun glare and printing deficiencies, particularly when the color blue is involved. See Appendix F for more details. Future improvement in physics-based modeling could improve our ability to robustify our attacks against a broader range of effects.

7.3. ALPR Attack

To demonstrate GRAPHITE’s ability to generalize to a real-world system in a different domain, we attack an

Automated License Plate Recognition (ALPR) system. We print license plate border stickers and attack expired license plates in physical field tests. The results are shown in Table 12. Details are included in Appendix G.

7.4. Ethical Concerns

While the algorithms described could be misused to discover vulnerabilities in real systems for potential attacks, it is important to provide system designers tools to understand potential vulnerabilities to hard-label physical attacks before deployment. Our hope is that the ability to discover these attacks quickly can guide future defense design as in Section 7.1. As perhaps the most practical threat model, hard-label physical attacks could represent the first stage of attacks that we should defend against - these are the attacks that an adversary could practically pull off in the real-world with the least amount of model access for the model to be useful.

8. Conclusion

We proposed GRAPHITE, a general framework for generating practical, real-world attacks on computer vision systems that satisfy three key requirements: that they are automatic, physical, and only require hard-label access to a machine learning classifier. GRAPHITE’s attacks are both automatically generated (e.g., without requiring specification of mask shapes or their location) and highly query-efficient compared to state-of-the-art in both white-box and black-box hard-label settings. In black-box hard-label settings, GRAPHITE is able to generate attacks that are orders of magnitude more efficient in terms of number of queries to the model than other state-of-the-art black-box hard-label attacks. As future direction for research, we plan to explore the use of GRAPHITE framework in adversarial training to help address an open problem - training models to better defend against robust physical perturbation attacks.

Acknowledgements

This material is based on work supported by DARPA under agreement number 885000, Air Force Grant FA9550-18-1-0166, the National Science Foundation (NSF) Grants 1646392, 2039445, CCF-FMitF-1836978, SaTC-Frontiers-1804648 and CCF-1652140, and ARO grant number W911NF-17-1-0405. Earlene Fernandes is supported by the University of Wisconsin-Madison Office of the Vice Chancellor for Research and Graduate Education with funding from the Wisconsin Alumni Research Foundation. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of our research sponsors. We thank Nelson Manohar for his help in early development, and Renuka Kumar, Washington Garcia, David Fouhey, Pin-Yu Chen, and Varun Chandrasekaran for their feedback.

References

- [1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [3] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.
- [4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [5] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.
- [6] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3354–3361.
- [7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
- [8] OpenPilot, "OpenPilot on the Comma Two," <https://github.com/commaai/openpilot>, 2020.
- [9] H. Bou-Ammar, H. Voos, and W. Ertel, "Controller design for quadrotor uavs using reinforcement learning," in *Control Applications (CCA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2130–2135.
- [10] C. Mostegel, M. Rumpler, F. Fraundorfer, and H. Bischof, "Uav-based autonomous image acquisition with multi-view stereo quality assurance by confidence prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2016, pp. 1–10.
- [11] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.
- [12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 39–57.
- [13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
- [14] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.
- [15] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.
- [16] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," *arXiv preprint arXiv:1707.07397*, 2017.
- [17] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *arXiv preprint arXiv:1712.09665*, 2017.
- [18] K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song, "Robust physical-world attacks on deep learning visual classification," in *Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [19] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, "Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16, 2016, p. 1528–1540.
- [20] M. Cheng, T. Le, P.-Y. Chen, J. Yi, H. Zhang, and C.-J. Hsieh, "Query-efficient hard-label black-box attack: An optimization-based approach," in *International Conference on Learning Representations*, 2019.
- [21] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," *arXiv preprint arXiv:1712.04248*, 2017.
- [22] T. K. S. Lab, "Experimental security analysis of Tesla AutoPilot," https://keenlab.tencent.com/en/whitepapers/Experimental_Security_Research_of_Tesla_Autopilot.pdf.
- [23] M. Lee and Z. Kolter, "On physical adversarial patches for object detection," *arXiv preprint arXiv:1906.11897*, 2019.
- [24] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen, "Dpatch: An adversarial patch attack on object detectors," *arXiv preprint arXiv:1806.02299*, 2018.
- [25] A. Fawzi and P. Frossard, "Measuring the effect of nuisance variables on classifiers," in *British Machine Vision Conference (BMVC)*, no. CONF, 2016.
- [26] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille, "Patchattack: A black-box texture-based attack with reinforcement learning," in *European Conference on Computer Vision*. Springer, 2020, pp. 681–698.
- [27] M. Cheng, S. Singh, P.-Y. Chen, S. Liu, and C.-J. Hsieh, "Signopt: A query-efficient hard-label adversarial attack," *arXiv preprint arXiv:1909.10773*, 2019.
- [28] J. Chen, M. I. Jordan, and M. J. Wainwright, "Hopskipjumpattack: A query-efficient decision-based attack," *arXiv preprint arXiv:1904.02144*, 2019.
- [29] C. Xiang, A. N. Bhagoji, V. Schwag, and P. Mittal, "PatchGuard: A Provably Robust Defense against Adversarial Patches via Small Receptive Fields and Masking," in *USENIX Security Symposium*, 2021.
- [30] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," *arXiv preprint arXiv:1607.02533*, 2016.
- [31] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin, "Black-box adversarial attacks with limited queries and information," in *Proceedings of the 35th International Conference on Machine Learning*, 2018, pp. 2137–2146.
- [32] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2341–2368, 2013.
- [33] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Foundations of Computational Mathematics*, vol. 17, no. 2, pp. 527–566, 2017.
- [34] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.
- [35] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *CoRR*, vol. abs/1611.02770, 2016. [Online]. Available: <http://arxiv.org/abs/1611.02770>
- [36] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM, 2017, pp. 15–26.
- [37] N. Narodytska and S. Kasiviswanathan, "Simple black-box adversarial attacks on deep neural networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, July 2017, pp. 1310–1318.
- [38] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
- [39] P.-y. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studer, and T. Goldstein, "Certified defenses for adversarial patches," *arXiv preprint arXiv:2003.06693*, 2020.
- [40] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural networks*, vol. 32, pp. 323–332, 2012.

- [41] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” *CiteSeer*, 2009.
- [42] V. Yadav, “p2-traffic signs,” <https://github.com/vxy10/p2-TrafficSigns>, 2016.
- [43] 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*, vol. 13, no. 4, pp. 1484–1497, 2012.
- [44] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [45] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [46] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft coco: Common objects in context,” in *European conference on computer vision*. Springer, 2014, pp. 740–755.
- [47] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.
- [48] H. Zhang, Y. Yu, J. Jiao, E. P. Xing, L. E. Ghaoui, and M. I. Jordan, “Theoretically principled trade-off between robustness and accuracy,” *arXiv preprint arXiv:1901.08573*, 2019.
- [49] H. Zheng, Z. Zhang, J. Gu, H. Lee, and A. Prakash, “Efficient adversarial training with transferable adversarial examples,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1181–1190.
- [50] S. Chen, N. Carlini, and D. Wagner, “Stateful detection of black-box adversarial attacks,” in *Proceedings of the 1st ACM Workshop on Security and Privacy on Artificial Intelligence*, 2020, pp. 30–39.

Appendix A. NP-Completeness of Mask Generation

We now explore the theoretical properties of mask generation and prove its NP-Completeness. Let an $n \times n$ square grid be represented as G_n , which is a graph (V_n, E_n) (V_n has vertices (i, j) , where $0 \leq i \leq n$ and $0 \leq j \leq n$ and for each (i, j) , $\{(i, j+1), (i, j-1), (i+1, j), (i-1, j)\} \cap V_n$ is in the set of edges E_n). A *mask* M is a sub-graph of the grid G_n that corresponds to a contiguous region of squares. Let $C(G_n)$ be the set of masks corresponding to the grid G_n . Let $\mu : C(G_n) \rightarrow \mathbb{R}^+$ be a *monotonic scoring function* ($M \subseteq M'$ implies that $\mu(M) \leq \mu(M')$). The mask generation problem can be stated as follows: Given r and threshold t , find a mask M of size $\leq r$ (the size of the mask is number of squares in it) such that $\mu(M) \geq t$. We call this problem $\text{MASK}_P(n, \mu, r, t)$.

Simply enumerating masks is not feasible because the number of masks could be exponential. We provide a simple argument. Consider a $k \times k$ sub-grid of G_n . Consider columns that are odd numbered (i.e. of the form $(i, \star)$, where i is odd). Now any choice of one square for the even columns gives us a contiguous mask, so there are $\geq 2^{\frac{k(k+1)}{2}}$ masks. There are $(n-k)^2$ $k \times k$ sub-grids in G_n . So a lower bound on masks of size k is at least $(n-k)^2 2^{\frac{k(k+1)}{2}}$. Next we prove that our problem is actually NP-complete.

The Set Cover. Given a universe $\mathcal{U}$ and a family $\mathcal{S}$ of subsets of $\mathcal{U}$, a cover is a subfamily $\mathcal{C} \subseteq \mathcal{P}(\mathcal{U})$ of sets whose union is $\mathcal{U}$. In the set-covering decision problem, the input is a triple $(\mathcal{U}, \mathcal{S}, k)$ (k is an integer), and the question is *whether there is a set covering of size k or*

less. In the set covering optimization problem, the input is a pair $(\mathcal{U}, \mathcal{S})$ and the task is to find a set covering that uses the fewest sets. The set-covering decision problem is known to be NP-complete.

Theorem 1. Problem MASK_P is NP-complete.

Proof. Our reduction is from the decision set-cover problem. Assume we are given an instance of the set-cover problem $(\mathcal{U}, \mathcal{S}, k)$. Let $n = \max(|\mathcal{U}|, |\mathcal{S}|)$. We create a $n \times n$ grid G_n . Let $C(G_n)$ be the set of masks of G_n . We construct a scoring function μ as follows: Let M be a mask. Let $I = \{i | (i, 0) \in M\}$, and $\mathcal{S}_I = \{S_j | j \in I \wedge S_j \in \mathcal{S}\}$. $\mu(M) = 1$ if and only if the following condition holds (otherwise $\mu(M) = 0$): $|I| \leq k$ and $\cup_{j \in I} S_j = \mathcal{U}$. It is easy to see that $\text{MASK}_P(n, \mu, n^2, 1)$ has a satisfying solution iff the instance of the set cover problem has a solution. This proves that the problem is NP-hard. Given a solution to the problem $\text{MASK}_P(n, \mu, r, t)$, it is easy to check that it is a valid solution in polynomial time, so the problem is NP. Therefore, $\text{MASK}_P(n, \mu, r, t)$ is NP-complete. $\square$

Appendix B. Transformation Details

Prior work by Eykholt *et al.* [18] and Athalye *et al.* [16] model environmental effects to create physical-world attacks in the white-box setting. These transformations account for varying conditions such as the distance and angle of the camera, lighting conditions, etc. Based on this work, we build a more principled set of transformations using classical computer vision techniques. To this end, we group these effects into 3 main classes of transformations:

- 1) **Geometric transformations:** These transformations refer to shape-based changes including rotation, translation and zoom. For planar objects, all three effects can be captured in a single perspective transformation through a homography matrix. Homography matrices relate two planar views under different perspectives. Geometrically, to convert points from one image plane to another, one can break down the operation into a rotation and translation matrix R , perspective projection onto a plane (P), and an affine transformation from plane to pixels (A). In the planar case, this boils down to a 3×3 homography matrix H :

$$x_{out} = APRx_{in} = Hx_{in} \quad (9)$$

We use these homographies to simulate rotation around the y axis and different viewing distances for given ranges of values. Once we pick values for each of the parameters uniformly, we construct the homography matrix to compute the transformation.

After performing the perspective transform, we random crop to the tightest square crop that includes the whole object $\pm c\%$ of the resultant image size to adjust for cropping errors. We also add random offsets for the crop, given as two

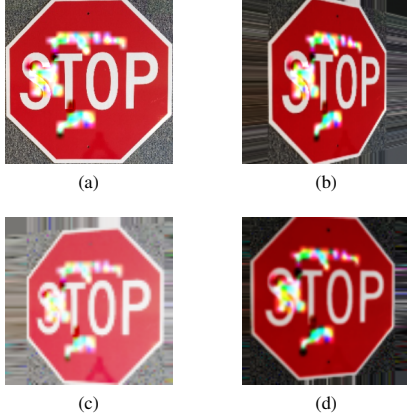


Figure 4. Examples of different transformed images. The upper-left image is the original, and the rest are three examples of transformed versions with perspective, lighting, and blurring transforms.

more parameters. Then, we resize the square to the original resolution.

- 2) **Radiometric transformations:** These are appearance-based transformations with effects such as lighting-based changes. One technique to perform brightness adjustments is gamma correction, which applies a nonlinear function. Separately, printers apply nonlinear functions to their colorspace as well. Gamma correction is reflective of nonlinear human sight perception. To model these radiometric-based changes, we model gamma correction under gamma values between $\frac{1}{\gamma}$ and γ , with half coming from $[\frac{1}{\gamma}, 1]$ and half coming from $[1, \gamma]$ in expectation where γ is the maximum gamma value allowed. Assuming the image ranges from $[0, 1]$, this is defined as the following:

$$x_{out} = x_{in}^{\gamma} \quad (10)$$

Note that one limitation of gamma correction is that if the color consists of RGB values of 0 or 255, the color does not change regardless of the gamma value.

- 3) **Filtering transformations:** These transformations model changes related to the camera focus. We model Gaussian blurring of different kernel sizes to measure the effects of the target object being out-of-focus. As a side benefit, this may help deal with printer error as some precision in color values is lost in printing. To maximize this printer side benefit, we blur just the perturbation, and let the perspective transform take care of minor out of focus variation in the rest of the image.

We define a single transformation to be a composite function that includes one of each type of modeled transformation. In our case with those listed above, we would have a perspective transform followed by a cropping operation, gamma correction, and a Gaussian blur convolution. Examples of transformed images are shown in Figure 4.



Figure 5. Example targeted, white-box automatic, physically-realizable attacks between: Stop, Speed Limit 30 km/hr, Pedestrians, Turn Left Ahead.

Appendix C. Additional White-box Details and Results

In this section, we discuss additional details on hyperparameters for the white-box experiment in Section 4. We also include example digital results in Figure 5. We set $z = 4$ (the number of patches to remove at a time) and $tr_{min} = 80\%$ (the stopping criteria of the minimum transform-robustness the EoT PGD attack must find). We choose values of the patch size p and step size s such that when rounded to the nearest int, the patches consists of the areas that 4×4 patches would occupy in the original 32×32 input resolution of GTSRBNet at a stride of 2. We compute transform-robustness with 100 transforms. PGD is performed with a step-size of $2/255$ and a max of 50 iterations per round of patch removal. The perturbation is restricted to be within the pre-defined boundaries of the victim traffic sign (if desired, object boundaries could be detected automatically with a segmentation network). Finally, we apply a random start on PGD with noise between $[-8/255, 8/255]$. Details on the specifics of GTSRBNet and the transformations, which mirrors that of the hard-label attack, can be found in Section 6.1 and Appendix B.

Appendix D. Additional CIFAR-10 Results

We include example images from our CIFAR-10 digital results in Table 13.

Appendix E. Drive-by Test Images

We include example images from our drive-by experiments in Table 14.

TABLE 13. RANDOM SAMPLE OF DIGITAL TARGETED ATTACKS ON CIFAR-10. RAN WITH $tr_{lo} = 65\%$, $tr_{hi} = 85\%$, $n = 100$ TRANSFORMS. FOR CELLS WITH SAME VICTIM AND TARGET, WE REPORT THE % OF TRANSFORMS THE ORIGINAL LABEL IS PREDICTED. MASKS SIZE IS REPORTED IN TERMS OF # OF PIXELS IN 32×32 (I.E., ℓ_0 IN 32×32 SPACE). THE ACTUAL VICTIM AND TARGET IMAGES USED IN EACH CELL ARE RANDOMLY SELECTED FROM THE POOL OF IMAGES OF THAT CLASS, WITH THE REFERENCE IMAGE IN THE TOP AND LEFT BEING USED FROM THE DIAGONAL IMAGES.

Victim	Target									
										
 Transform-robustness Mask Size	 89% 0	 96% 111	 86% 21	 91% 63	 78% 73	 91% 75	 84% 67	 84% 113	 73% 166	 75% 93
 Transform-robustness Mask Size	 100% 128	 78% 0	 82% 451	 93% 121	 73% 256	 89% 97	 90% 273	 92% 92	 49% 146	 100% 46
 Transform-robustness Mask Size	 96% 66	 78% 233	 18% 0	 98% 12	 92% 21	 93% 101	 100% 53	 100% 88	 95% 170	 79% 90
 Transform-robustness Mask Size	 100% 53	 92% 235	 98% 74	 97% 0	 99% 67	 96% 115	 98% 109	 78% 122	 67% 167	 79% 74
 Transform-robustness Mask Size	 98% 39	 97% 114	 96% 102	 100% 92	 97% 0	 61% 34	 91% 45	 82% 149	 46% 216	 98% 67
 Transform robustness Mask Size	 100% 61	 98% 167	 89% 33	 99% 68	 95% 49	 100% 0	 93% 105	 76% 77	 79% 610	 95% 125
 Transform robustness Mask Size	 98% 48	 99% 116	 100% 44	 83% 80	 81% 88	 97% 29	 94% 0	 98% 63	 51% 414	 60% 138
 Transform-robustness Mask Size	 80% 197	 99% 121	 84% 152	 89% 89	 100% 122	 98% 24	 67% 264	 77% 0	 80% 105	 100% 54
 Transform-robustness Mask Size	 100% 58	 57% 120	 99% 61	 95% 79	 91% 201	 70% 19	 100% 961	 91% 63	 96% 0	 100% 73
 Transform-robustness Mask Size	 94% 90	 95% 66	 95% 271	 100% 49	 73% 237	 72% 139	 93% 77	 82% 74	 24% 251	 100% 0

Appendix F. Printing and Lighting Error

We noticed a limitation of gamma correction’s ability to model sun glare and printing errors while testing a Stop Sign to Turn Left Ahead attack. Of particular interest was the discrepancy between the modeled attack’s blues in the digital form and the captured attack’s blues in the physical form.

Examples images for the initial field tests are shown in Table 15 and example images for drive-by tests are

shown in Table 16. In the drive-by tests, 42 frames were analyzed and the transform-robustness was 19%.

We can see that the Turn Left Ahead attack is less successful than the Speed Limit 30 km/hr or Pedestrians attack. However, we can easily see that the blue over the “P” is very different in the digital and physical versions, and we hypothesize that it plays a big factor in the performance drop.

To test this hypothesis, we take the outdoor images and digitally darken them. We find that by darkening them we are able to raise the attack success rate from 0% to

TABLE 14. SAMPLE OF GTSRB DRIVE-BY TEST PICTURES. TOP: SPEED LIMIT 30 KM/HR ATTACK. BOTTOM: PEDESTRIANS ATTACK.



52.9%. Likewise, we are also able to increase the attack success rate for the drive-by images by digitally increasing the contrast and decreasing the brightness. This raises the attack success rate from 19% to 47.6%, including successful attacks on the last 11 frames. This suggests that much of the inaccuracy can be attributed to the modeling and error of the blue color.

We can trace this to two limitations printing and lighting error. Printing blue was harder than other colors. In a related issue, a second limitation was the inaccuracies of modeling sun glare, as shown by the increase of attack success in indoor lighting settings (Table 15).

Examining the problem more closely, the color in the blue spot over the “P” includes many instances of the tuple $(0, 0, 255)$. While our transforms adjust for lighting with gamma correction, its exponential nature always leaves a tuple like $(0, 0, 255)$ as $(0, 0, 255)$. This suggests a limitation of using gamma correction as a method for modeling lighting changes (especially impact of sunlight) for colors at the extreme ends of the RGB spectrum.

A better model to address lighting changes at extreme ends of the spectrum in EoT transforms may increase robustness. More accurate modeling of printing error in the transformations could help mitigate this effect as well. We add that we found it difficult to improve the printing quality by adding the NPS term as in RP_2 [18] to GRAPHITE, as oftentimes the limited color palette either made too difficult to find good attacks. Finally, while point source lighting and further sun glare modeling could be conceptually done, this would require knowledge of physical surface properties (reflectivity, etc.) and weather/environmental conditions beyond what was available with the current data.

Appendix G. ALPR Attack

This section includes details on attacking an Automated License Plate Recognition (ALPR) system with GRAPHITE.

Dataset and Classifiers. For ALPR, we use OpenALPR version 2.3.0, the latest freely available version. We treat this command line tool as a complete black-box. While this particular tool provides confidence scores,

others may not, so we do not use the scores during our attack. We initialize with the same image but with a gray rectangle filled in with $(127, 127, 127)$ over the plate.

ALPR Attack Details. To attack ALPR systems, we imagine printing a license plate holder sticker to cause the ALPR system to fail to detect your license plate number correctly (in an untargeted fashion). In this case, we could attack with just the boosting stage from the (known) border mask consisting of the license plate holder, but we found this typically yielded a result with poor initial transform-robustness. To overcome that, we alternate between mask generation and boosting in multiple rounds as initially discussed in Section 6.3.5. We essentially want to reduce the mask inside of the plate border to zero, but cannot initially apply boosting on just the border, so we find it useful to slowly remove the inside of the plate until we have a sufficiently transform-robust border.

To test our ALPR attack, we print out the license plate holder stickers and place it on expired license plates we acquired for purposes of field testing. We ran ALPR on stationary pictures of the car taken in a driveway and took five pictures of the car at 5’, 10’, 15’, and 20’ away at both 0 and 15 degree angles. We generate our attack on images of height 500. The perturbation was generated at a height of 250 and enlarged to fit over the whole image.

As with GTSRB, we use the original author’s YOLOv3 [45] object detector network trained on MS COCO [46] to predict bounding boxes for the car. We take the output bounding boxes, crop the sign accordingly, and send the crops to the black-box ALPR pipeline.

Hyperparameters. Like our GTSRB attack, we test over $n = 100$ transformations to compute transform-robustness and take $q = 10$ gradient samples for RGF sampling [32]. For transformations, we model rotations about the y axis with homography matrices, lighting changes with gamma correction, and focus changes with Gaussian blurring.

For our ALPR attack, we set rotation to be between -15° and 15° and fix the base focal length $f = 10$ ft. We set the Gaussian kernels to sizes 1, 3, and 5, and let the remaining parameters match the GTSRB attack. We used 3 iterations of mask generation and boosting. The patch size for mask generation was 8×8 , then 4×4 , and then in the last iteration the mask was fixed to just the border. The stride for the patches was the width divided by 2. We additionally added in the backtracking line search to adaptively select the step size as in OPT-attack [20]. We set $tr_{lo} = 20\%$ and $tr_{hi} = 60\%$. We use $n = 10$ transformations in mask generation and $n = 50$ transforms in boosting. We set λ in (6) to 25. We do not utilize the m_{max} option.

Physical Transform-Robustness Field Tests: We also evaluate ALPR license plate holder attacks on two plates and cars. In total, we evaluated over 700 physical images.

Table 12 shows ALPR field test results. These attacks took an average of 12950 queries. We found these attacks to have physical success as well. The Washington plate attack success rate was 82.5%. The transform-robustness (digital) for this attack was 80%. 100% of unperturbed, baseline images correctly predicted the license plate. For the Michigan plate attack, the attack succeeded in 67.5% of images while the transform-robustness for this attack was 86%. 82.5% of unperturbed, baseline images correctly

TABLE 15. GTSRB FIELD TEST RESULTS FOR VICTIM `STOP` SIGN AND TARGET `TURN LEFT AHEAD`. PHYSICAL ROBUSTNESS RESULTS ARE CALCULATED OVER 5 PICTURES EACH AT THE FOLLOWING SPOTS: 5 FT $\times$ {0°, 15°, 30°, 45°}, 10 FT $\times$ {0°, 15°, 30°}, 15 FT $\times$ {0°, 15°}, 20 FT $\times$ {0°, 15°}, 25 FT, 30 FT, 40 FT. THE ATTACK WAS TESTED 3 TIMES: OUTDOORS, INDOORS WITH INDOOR LIGHTS TURNED OFF, AND INDOORS WITH INDOOR LIGHTS TURNED ON.

Victim	Target	Digital GRAPHITE attack	Physical GRAPHITE attack (outdoors)	Dig. TR (100 xforms)	Phys. TR (Indoors, lights off)	Phys. TR (Indoors, lights on)	Phys. TR (Outdoors)
				80%	55.7%	55.7%	0%

TABLE 16. SAMPLE OF GTSRB DRIVE-BY TEST PICTURES FOR `TURN LEFT AHEAD` ATTACK.



predicted the license plate. The average Levenshtein distance, which calculates the number of additions, subtractions, and substitutions required to change one string to another, was 2.175 (including correct predictions). These results also suggest that transform-robustness translates well to physical-world robustness.