DorPatch: Distributed and Occlusion-Robust Adversarial Patch to Evade Certifiable Defenses

Chaoxiang He*, Xiaojing Ma*[1], Bin B. Zhu†, Yimiao Zeng*,
Hanqing Hu*, Xiaofan Bai*, Hai Jin*, and Dongmei Zhang†
*Huazhong University of Science and Technology
†Microsoft
Emails: {hechaoxiang, lindahust}@hust.edu.cn, binzhu@microsoft.com,
{zengyimiao, huhanqing, xiaofanbai, hjin}@hust.edu.cn, dongmeiz@microsoft.com

***Abstract*—Adversarial patch attacks are among the most practical adversarial attacks. Recent efforts focus on providing a certifiable guarantee on correct predictions in the presence of white-box adversarial patch attacks. In this paper, we propose DorPatch, an effective adversarial patch attack to evade both certifiably robust defenses and empirical defenses. DorPatch employs group lasso on a patch's mask, image dropout, density regularization, and structural loss to generate a fully optimized, distributed, occlusion-robust, and inconspicuous adversarial patch that can be deployed in physical-world adversarial patch attacks. Our extensive experimental evaluation with both digital-domain and physical-world tests indicates that DorPatch can effectively evade PatchCleanser [64], the state-of-the-art certifiable defense, and empirical defenses against adversarial patch attacks. More critically, mispredicted results of adversarially patched examples generated by DorPatch can receive certification from PatchCleanser, producing a false trust in guaranteed predictions. DorPatch achieves state-of-the-art attacking performance and perceptual quality among all adversarial patch attacks. DorPatch poses a significant threat to real-world applications of DNN models and calls for developing effective defenses to thwart the attack.**

I. Introduction

Deep neural networks (DNNs) are known to be vulnerable to adversarial attacks [56]. Most adversarial attacks and defenses focus on adversarial examples with low perceptible adversarial perturbations of small L_2 or L_∞ distance added to original samples [4], [7], [11], [16], [17], [22], [66]. These attacks can perturb all image pixels, making them impractical for physical-world adversarial attacks, wherein a real-world object is modified, typically using stickers or paint, to cause misprediction. In contrast, adversarial patch attacks [6], [19], [30] are among the most practical adversarial attacks against real-world applications of DNN models [9]. These attacks craft a typically visible adversarial patch with small support (the number of pixels of the patch) instead of low perceptible perturbations. Such a patch can be printed out and stuck or painted on a real-world object to launch a physical-world attack. The number of pixels in the adversarial patch is referred to as the *patch budget* or *patch support* in this paper.

Due to the practical threat of adversarial patch attacks, defense research has been active. Early defenses [25], [45] are empirical and easily evaded by stronger adversarial patch attacks [9]. Recent defenses, like the *state-of-the-art* (SOTA) PatchCleanser [64], focus on providing certifiable guarantees for correct predictions in the presence of white-box adversarial patch attacks. Like other certifiable defenses, PatchCleanser assumes an adversarial patch is spatially bounded. It uses a mask large enough to cover the patch and applies two masking rounds to each input to make prediction and certify its robustness against adversarial patch attacks.

However, a physically realizable patch is not necessarily spatially bounded. This raises the questions: without the spatial boundedness assumption, can certifiable defenses be evaded, and can adversarially patched examples be certified by these defenses? We aim to answer these research questions in this study.

At first glance, one might intuitively think that a distributed patch like RP_2 [19], which uses a set of distributed stickers as a patch, would be able to evade PatchCleanser [64]. However, our experiments to be reported in Section VI show that this is not the case. RP_2 cannot evade PatchCleanser or make adversarially patched examples certified by it. The masking operation in PatchCleanser corrupts RP_2's distributed patches, causing them to lose their adversarial properties.

In this paper, we introduce *DorPatch* (Distributed and occlusion-robust Patch), a powerful adversarial patch attack that can evade both certifiable and empirical defenses. Our adversarial patches are physically realizable and can be used to launch attacks in the real world, like other patch attacks. We use the factorization method in SAPF [20] to decompose a patch into the Hadamard product of a binary mask and pattern (i.e., pixel values), resulting in a *Mixed Integer Programming* (MIP) problem that cannot be solved directly with common

*Chaoxiang He, Xiaojing Ma, Yimiao Zeng, Hanqing Hu, and Xiaofan Bai are with National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Hubei Engineering Research Center on Big Data Security, Hubei Key Laboratory of Distributed System Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology. Hai Jin is with National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology.
[1]Corresponding author: Xiaojing Ma (lindahust@hust.edu.cn).

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.24920
www.ndss-symposium.org

optimizers such as Stochastic Gradient Descent (SGD) or Adam [31]. We solve the MIP problem with a two-stage optimization process, which first optimizes the location and shape of the patch, and then optimizes the pixel values inside the patch.

To facilitate physical-world attacks, we enforce a minimum area for each separated part of a distributed patch to contain one or more groups and use group lasso [69] on the mask of the patch for group sparsity. We apply image dropout and density regularization to make the patch widely distributed and resilient to masking occlusions. This occlusion robustness ensures consistent predictions for a patched example after masking at different locations, enabling misprediction and certification of adversarial examples by PatchCleanser. Certification poses a more severe security threat, as it induces false trust in guaranteed predictions.

Furthermore, DorPatch applies a structural loss to ensure that patched pixels are located in more complex regions and form structures commonly seen in clean images. The combination of the structural loss and an L_2 bound makes our patches inconspicuous for physical-world attacks and almost invisible, resembling adversarial attacks that perturb all pixels, for digital-domain attacks. DorPatch can be used for both targeted and untargeted patch attacks.

We conduct extensive experiments to evaluate DorPatch, including both digital-domain evaluation and physical-world evaluation. Our empirical studies validate DorPatch's effectiveness. To the best of our knowledge, DorPatch is the first adversarial patch attack that can evade SOTA certifiable defense, PatchCleanser, and enable adversarially patched examples to be certifiable by PatchCleanser.

Certified robustness defenses generally rely on the boundedness of adversarial patches in the image or feature domain and apply masking to detect or neutralize their effects. To avoid significant degradation of the model's clean accuracy, the mask in the masking operation cannot be too large. DorPatch exploits this fundamental limitation by making an adversarial patch distributed and robust to such masking operations. We argue that all certifiable defenses that rely on the boundedness of adversarial patches are practically evadable when facing DorPatch-like attacks. We call for research efforts to develop more powerful defenses against DorPatch-like adversarial patch attacks.

This paper includes the following major contributions:

- We propose DorPatch, the first adversarial patch attack that can effectively evade PatchCleanser, a SOTA certifiable defense against adversarial patch attacks, and enable adversarially patched examples certifiable by PatchCleanser. It can also evade other certifiable defenses and empirical defenses.
- We propose an image dropout technique and density regularization to make a generated patch widely distributed and robust to masking occlusions. obustness to occlusions plays a key role both for evading PatchCleanser and for making adversarially patched examples certifiable by PatchCleanser.
- We design a structural loss function to enforce natural structures within an adversarial patch and position patch pixels in less noticeable regions for less conspicuous patches. When combined with an L_2 constraint, our adversarial patches become inconspicuous in physical-world patch attacks and nearly invisible in digital-domain patch attacks.
- Instead of applying group lasso [69] to adversarial perturbations as in existing methods [20], [67], we apply group lasso to the adversarial patch mask. We solve the *Mixed Integer Programming* (MIP) problem using an efficient two-stage optimization method. The first stage optimizes the shape and location of the adversarial patch, and the second stage optimizes the pixel values of the patch.
- We conduct extensive experiments in both the digital domain and the physical world to validate the effectiveness of DorPatch in evading both certifiable and empirical defenses. These comprehensive evaluations demonstrate the effectiveness of DorPatch against various defense mechanisms.

The DorPatch code is available at: https://github.com/CGCL-codes/DorPatch.

II. Related Work

A. Adversarial Patch Attacks

The first white-box physically realizable adversarial attack is an adversarial eyeglass frame [54], considering smoothness and printability. Wearing this frame can evade face recognition or impersonate someone. GoogleAP [6] crafts universal targeted adversarial patches, considering physical factors like location, rotation, and scale. LaVAN [30] uses a pre-defined fixed mask for localized adversarial patches in targeted and untargeted attacks. LOAP [46] generates untargeted image-specific patches, optimizing their locations.

Adversarial patches in these attacks are conspicuous. Researchers have tried to create visible yet inconspicuous patches. RP_2 [19] generates graffiti-like patches, such as stickers, for being less noticeable and more robust in physically attacking traffic sign recognition models. TNT Attack [15] optimizes in a generative model's latent space for universal, natural-looking patches. PS-GAN [38] uses a perceptual-sensitive GAN for balancing visual fidelity and attackability. IAP [3] employs a coarse-to-fine approach with multi-scale generative models for more inconspicuous patches. Both PS-GAN and IAP utilize a pre-trained model as the attention model to identify important areas of the source image for patch placement.

While the above adversarial patch attacks assume a white-box threat model, PatchAttack [68] and Meaningful Adversarial Sticker [58] operate under a black-box setting. The former generates adversarial textured patches with reinforcement learning, while the latter manipulates real stickers' positions and rotation angles for physical attacks on face recognition.

Adversarial patch attacks have also been proposed for other task DNN models, such as object detection [28], [39], [71], semantic segmentation [50], and network traffic analysis [53]. Shapeshifter [8] introduces a physical-world attack on Faster R-CNN object detector [47] by perturbing stop signs.

We focus on adversarial patch attacks against image classification models, but our proposed attack can be extended to other task DNN models. We will compare typical white-box attacks with ours on fulfillment of desired properties in Section III-C.

B. Adversarial Patch Defenses

1) Empirical Defenses: Early defenses against adversarial patch attacks rely on empirical observations. *Local Gradient Smoothing* (LGS) [45] suppresses adversarial noise by smoothing pixels based on image gradients[1], as adversarial patches often have sharp pixel value variations. *Digital Watermarking* (DW) [25] detects dense, highly sensitive regions in an input image using a sensitivity-based saliency map and masks them for prediction, as perturbed pixels in adversarial patches are highly sensitive to prediction results.

However, LGS and DW can degrade clean accuracy, as they also suppress structural edges, feature points, and sensitive regions in clean images. TaintRadar [36] and Februuss [14] address this issue by detecting and removing localized adversarial patches, with Februuss using a GAN-based inpainting method for reconstructing the removed area.

Empirical defenses like LGS and DW are shown to be ineffective against adaptive white-box adversarial patch attacks [9]. LGS can be bypassed by incorporating smoothing into patch generation, while DW can be circumvented using *Backward Pass Differentiable Approximation* (BPDA) [2] to approximate non-differentiable operations during patch crafting.

2) Certifiable Defenses: Researchers have developed certifiably robust defenses against adversarial patch attacks due to the failure of empirical defenses. The *Interval Bound Propagation* (IBP) defense [23], [44] is extended in [9] for certifiably-robust networks. However, it has shortcomings like conservative bounding of neuron activation values, expensive model training, and inability to scale to large models and high-resolution images. Randomized smoothing [10], [33], [35] is adopted in [34], [37], [49] for certifiable defenses but suffers from significant inference overhead and degraded clean accuracy.

Certifiable defenses like *Clipped BagNet* (CBN) [70], BagCert [43], and PatchGuard [63] rely on DNNs with small receptive fields, making them impractical for networks like ResNet [27] with large receptive fields.

Certifiable defenses focusing on detecting adversarial patches include *Minority Reports* (MR) [42], PatchGuard++ [65], ScaleCert [24], and PatchCleanser [64]. PatchCleanser assumes an adversarial patch is spatially bounded and applies a large enough mask that can completely cover the adversarial patch at all image locations in two rounds to an input image. If unanimous predictions are achieved for all masked images in the first round, PatchCleanser determines that the input is benign and outputs the prediction. Otherwise, it applies a second round of masking to each masked image from the first round and outputs the unanimous or majority prediction in the second round of masking. If unanimous predictions are achieved in both rounds, the input is certifiably robust to adversarial patch attacks. PatchCleanser provides state-of-the-art certifiably robust defense and is agnostic to DNN architectures. We select it to evaluate the performance of our proposed DorPatch.

[1]Image gradients differ from gradients used in model learning. The former is with respect to changes of adjacent pixel values while the latter is with respect to classification loss.

C. Adversarial Training

Adversarial training [22] is one of the most effective methods for training models robust to adversarial examples. It generates adversarial examples and applies them in training a model. Adversarial training is used in [46] and [61] to train a model robust to adversarial patch attacks. However, adversarial training incurs high computational overhead. To mitigate this, the inner optimization step is replaced with an expectation over random augmentations in [1].

D. Group Lasso and Other Related Work

Group lasso [69] is applied to sparse adversarial perturbations in StrAttack [67] and SAPF [20] for group sparsity and clustering of perturbed pixels, leading to more semantically structured adversarial perturbations with better interpretability. StrAttack optimizes with the *Alternating Direction Method of Multipliers* (ADMM) and applies a threshold to remove insignificantly perturbed pixels from the optimization result to produce an adversarial perturbation. SAPF factorizes each perturbed pixel into a product of a perturbation value and a binary pixel selection factor, resulting in an MIP problem. Due to the discrete nature of the binary selection factor, the MIP problem cannot be directly solved with conventional optimizers such as the gradient descent algorithm or the Adam optimizer [31]. SAPF applies l_p-Box ADMM [59] to reformulate the MIP problem as a continuous optimization problem and solves it with ADMM and the gradient descent algorithm in [20].

We apply group lasso to the adversarial patch mask and follow SAPF's factorization method to factorize a patch into the Hadamard product of a binary mask and pattern (i.e., pixel values) to form an MIP problem in our DorPatch attack, and solve it with a two-stage optimization method. Although our two-stage method lacks the mathematical rigor of SAPF's MIP solution, it provides similar or even better attacking performance for our DorPatch compared to SAPF's, as will be presented and further discussed in Section VI-H.

Human perceptibility is taken into consideration in [41] to generate more imperceptible adversarial examples, wherein the standard deviation in a local region is used as the perceptibility metric in generating adversarial perturbations. We also use the standard deviation as the perceptibility metric in our structural loss (Eq. 3 in Section IV-E) to place patch pixels in less perceptible regions, in addition to enforcing natural structures in adversarially patched examples.

III. Threat Model and Desirable Properties

A. Threat Model

In this paper, we assume *white-box access* to the DNN model under attack and *black-box access* to potential defenses against DorPatch. Specifically, adversaries have full access to the DNN model, including its architecture and parameters, but

no knowledge of any defense (its characteristics or settings) against DorPatch. Adversaries can modify any image pixels to craft an adversarial patch, which can be located anywhere in an image. We have removed the restriction in PatchCleanser and other certifiable robustness defenses that an adversarial patch must be spatially bounded.

B. Limitations of Existing Patch Attacks

Existing adversarial patch attacks typically employ a localized patch. The shape, location, and size of the patch in many attacks are predetermined and remain fixed during the optimization process to craft an adversarial patch. Such attacks have two major drawbacks. First, the location, shape, or size of the crafted adversarial patch may not be optimal, resulting in a less powerful adversarial attack. Second, the adversarialness is realized with adversarial pixels typically located in a small, restricted region. This strong locality has been exploited by certifiable robustness defenses, such as PatchCleanser [64], to detect and neutralize adversarial patches.

PatchCleanser relies on two assumptions: 1) The DNN model is robust to occlusion of a small-size mask at arbitrary locations of an input image, i.e., a benign image arbitrarily occluded by the mask likely yields consistent and correct predictions. 2) The adversarial patch can be fully occluded by the mask at an appropriate location. The first assumption requires that the mask should be small enough to avoid significant degradation of the model's clean accuracy. The second assumption requires the mask to be large enough to completely cover the adversarial patch.

RP_2 [19] uses a distributed graffiti-like adversarial patch, such as stickers, which may not be fully covered by a single mask in PatchCleanser while preserving clean accuracy. As will be shown in Section VI, this distributed adversarial patch is insufficient to evade PatchCleanser. The masking operation in PatchCleanser may corrupt the adversarial patch, causing it to lose its adversarialness and allowing PatchCleanser to predict correctly. If the simpler goal of causing PatchCleanser to mispredict cannot be achieved, it becomes infeasible to reach the more challenging goal of making adversarially patched examples certifiable by PatchCleanser.

C. Desired Properties of Patch Attacks

Since PatchCleanser is the SOTA certifiable robustness defense against adversarial patch attacks, we focus on it when describing DorPatch. However, the proposed method can be extended to other certifiable robustness defenses. To evade PatchCleanser and unleash its attacking power, DorPatch should possess the following desirable properties:

- **Distributed**: To evade PatchCleanser, an adversarial patch should be widely distributed to prevent being fully occluded by a small exploring mask. DorPatch achieves this through density regularization, which will be described in Section IV-B.
- **Robust to Partial Occlusions**: A distributed adversarial patch, like RP_2 [19], is not sufficient to evade PatchCleanser. To do so, the patch should be robust to partial occlusions at various locations, forcing PatchCleanser to use a large mask that degrades the model's clean accuracy. Our goal is not only to make PatchCleanser mispredict but also to create adversarial examples that are certifiably robust by PatchCleanser, fostering a false sense of trust in certified predictions. To achieve this, an adversarially patched example should produce consistent predictions in both masking rounds of PatchCleanser. DorPatch accomplishes both goals using image dropout, as will be described in Section IV-C.
- **Fully Optimized**: An adversarial patch should be fully optimized, including its shape, location, and pixel values, to achieve the most effective attack within a given patch budget. DorPatch accomplishes this goal by employing group lasso on mask (to be described in Section IV-D) and a two-stage adversarial patch generation process (to be described in Section V).
- **Inconspicuous**: An adversarial patch should be inconspicuous. To enhance the inconspicuousness of an adversarial patch and avoid being neutralized by image processing techniques, such as suppression of large image gradients [45], perturbed pixels should result in structural indistinguishability and perceptual masking should be considered when determining the locations and pixel values of perturbed pixels. Structural indistinguishability means that perturbation should result in modifications with structures frequently found in clean images, like continuous and smooth structures, rather than abrupt image changes. DorPatch achieves inconspicuousness using structural loss, which will be described in Section IV-E.

Adversarial patch attacks should fulfill all the above four desirable properties. Table I lists the fulfillment of the four desirable properties for several existing typical patch attacks and our DorPatch. We can see that among the considered patch attacks, only DorPatch possesses all four desired properties.

TABLE I. FULFILLMENT OF DESIRED PROPERTIES OF ADVERSARIAL PATCH ATTACKS

Attack\Property	Distributed	Robust to Occlusion	Inconspicuous	Location-optimized
DorPatch	✓	✓	✓	✓
LaVAN [30]				
LOAP [46]				✓
IAP [3]			✓	✓
RP_2 [19]	✓			✓

IV. Achieving Desirable Properties in DorPatch

We use the factorization method in SAPF [20] to decompose a patch into the Hadamard product of a binary mask and a pattern (i.e., pixel values), and optimize both the mask and the pattern. This results in an MIP optimization problem. In this section, we introduce several losses that enable DorPatch to achieve the desired properties outlined in the previous section. In the following description, we assume that an image-specific adversarial patch is generated. To create a universal adversarial patch, we can extend the described losses by adding an extra summation over all images to which the universal adversarial patch is applied.

A. Notation

Before describing our losses, we present the notation that will be used, which is summarized in Table II. X and x_i denote

an image and the i-th pixel in the image, respectively. M denotes the mask of a patch, which consists of 1s and 0s, where 1 indicates that a pixel is part of the patch, and 0 indicates that a pixel is not part of the patch. Consequently, M determines the location, shape, and size of the patch. Δ denotes the pattern of a patch, which represents the pixel values of the patch. Both M and Δ are determined through an optimization process in DorPatch. $X_\Delta \equiv X \oplus \Delta$ represents the resulting image after applying a patch with pattern Δ to image X according to patch mask M. $\cdot$ denotes the dot product (i.e., the summation of element-wise products). $\circ$ denotes the Hadamard product (i.e., element-wise product). $|\cdot|$ denotes the cardinality of a set. $\|\cdot\|_p$ is the L_p norm. $\mathbb{E}$ denotes the expectation operation.

TABLE II. NOTATION

Notation	Definition
X	an image
x_i	the i-th pixel in the image
M	the mask of a patch
Δ	the pattern of a patch
$X_\Delta \equiv X \oplus \Delta$	the patched image
$\cdot$	the dot product
$\circ$	the Hadamard product (i.e., element-wise product)
$\|\cdot\|$	the cardinality of a set
$\|\cdot\|_p$	the L_p norm
$\mathbb{E}$	the expectation operation

B. Density Regularization

To encourage a patch to be widely and uniformly distributed, we use a set of sampling regions, $\mathcal{A}$, which evenly divide an image into $|\mathcal{A}|$ parts. Our goal is to make the density of patch pixels in each region similar. We achieve this by minimizing the standard deviation of the number of selected pixels for the patch in each sampling region over all regions in $\mathcal{A}$. This is referred to as *density regularization*:

$$L_{den} = \sqrt{\frac{1}{|\mathcal{A}|}\sum_{\mathbf{a}\in\mathcal{A}}(M\cdot\mathbf{a} - \mathbb{E}_{\mathbf{a}\in\mathcal{A}}(M\cdot\mathbf{a}))^2} \quad (1)$$

where $\mathbf{a} \in \mathcal{A}$ is the region's mask, $M\cdot\mathbf{a}$ is the dot product of M and $\mathbf{a}$, i.e., the number of patch pixels in $\mathbf{a}$. This term encourages the mask of a generated patch to be evenly distributed among different regions in $\mathcal{A}$.

C. Image Dropout

To make a generated adversarial patch robust to partial occlusions and ensure the patched image is certifiably robust by PatchCleanser, we randomly mask out parts of the image during the patch optimization process. Specifically, we collect a set of possible occlusions, $\mathcal{B}$, such as squares of different sizes and positions. In each iteration, we generate $\mathcal{N}$ occluded images $X^i_\Delta, i \in [1,\mathcal{N}]$ from the patched image X_Δ and optimize them together. To obtain each X^i_Δ, we randomly choose n_o occlusions from $\mathcal{B}$ and remove the corresponding regions from X_Δ. This image dropout process is illustrated in Fig. 1.

The choice of n_o depends on the goal: to make PatchCleanser misclassify the patched image, we set n_o to 1. To make the patched image certifiably robust by PatchCleanser, we set n_o to 2. $\mathcal{N}$ should be large enough to cover various occlusion scenarios so that the generated patch will remain effective under PatchCleanser's removal. We find that increasing

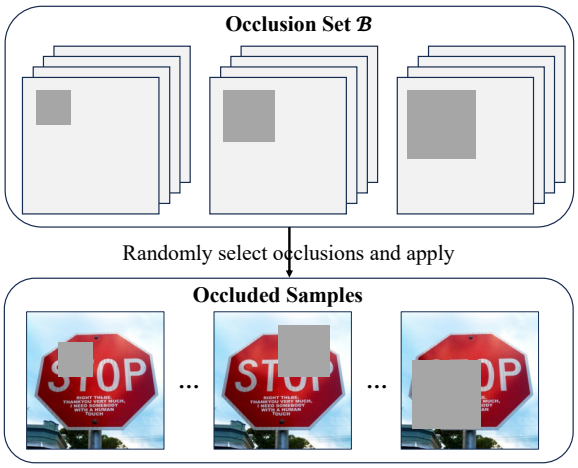


Fig. 1. An illustration of the image dropout process in one iteration of patch generation. We randomly choose n_o occlusions from the set $\mathcal{B}$ and remove the corresponding regions from the patched image X_Δ to produce an occluded image. We do this $\mathcal{N}$ times to obtain $\mathcal{N}$ occluded images $X^i_\Delta, i \in [1,\mathcal{N}]$ and optimize them together in a single batch.

$\mathcal{N}$ beyond a certain point does not improve the performance significantly. In our experiments, we use $\mathcal{N} = 128$ as the default value.

D. Group Lasso on Mask

To physically attack a target object, an attacker typically prints and cuts out a patch and then sticks it on the object. We want the patch to be easy to handle, so we design it such that each isolated part of it is large enough and has a rather regular shape. To achieve this, we only allow the patch to include or exclude whole groups of pixels, where a *group* consists of adjacent pixels and is the smallest unit for an isolated part of a patch that DorPatch generates. In our implementation, a group is a square subregion of $K \times K$ pixels, and we divide an image into non-overlapping groups. The parameter K determines the minimum area of each isolated part of the patch. Fig. 2 shows how we partition a 4×4 image into 2×2 groups.

We apply group lasso [69] to the mask M to enforce group sparsity, i.e., to minimize the number of groups in the patch:

$$L_{grp} = \sum_{l=1}^{m} \|M \circ G_l\|_2 \quad (2)$$

where G_l is the indicator matrix for the l-th group (element 1 means the corresponding pixel is in the group) and m is the total number of groups. As stated in Section IV-A, $\circ$ denotes the Hadamard product and $\|\cdot\|_2$ is the L_2-norm of a vector. By minimizing L_{grp}, we encourage many groups in M to become zero vectors, i.e., to exclude all their pixels from the patch.

E. Structural Loss

To encourage perturbed pixels to result in continuous and smooth structures, we propose the following *structural loss*:

$$L_{str} = \sum_{x_i \in X_\Delta} \frac{1}{V_i}\Big(\sum_{x_j \in N(x_i)} (x_i - x_j)^2 \cdot \min_{x_j \in N(x_i)} (x_i - x_j)^2\Big) \quad (3)$$

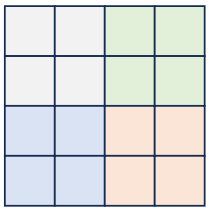


Fig. 2. An illustration of non-overlapping group partitioning: a 4×4 image is divided into 2×2 groups. Each cell represents a pixel. The cells in the same color form a group.

where $N(x)$ returns all neighborhood pixels of x over the patched image X_Δ, and V_i is the local variance of pixel values of the input image X at pixel x_i.

The inner summation in Eq. 3, i.e., $\sum_{x_j \in N(x_i)}(x_i - x_j)^2$, is the conventional total variation loss, which encourages smooth changes among neighboring pixels for each perturbed pixel. The next term in Eq. 3, i.e., $\min_{x_j \in N(x_i)}(x_i - x_j)^2$, is the minimal variance loss, which is the minimal squared pixel-value difference with adjacent pixels. It is small when a neighboring pixel has a similar value. It allows preserving a sharply changing pixel as long as at least one neighboring pixel has a similar pixel value, such as an edge pixel. The product of these two terms encourages smooth and edge-like structural modifications and suppresses outliers that deviate significantly from *all* of their neighboring pixels.

Following [41], we approximate the local perceptual masking power at a pixel x_i with its local variance V_i. We weight the product at pixel x_i by the inverse of its local variance V_i of the input image X. This weighting encourages patch pixels located in complex regions to increase inconspicuousness. Overall, the structural loss is small when the current pixel is either locally smooth, an edge point, or locally complex (i.e., exhibiting high local variance).

V. Generation of Adversarial Patches

To achieve the goal of a targeted or untargeted adversarial patch attack, we require an adversarial loss, denoted as L_{adv}. By incorporating this loss with the losses described in Section IV, we can formulate DorPatch's optimization problem as follows:

$$\begin{aligned} &\min_{M,\Delta} L_{adv} + \lambda_1 \cdot L_{grp} + \lambda_2 \cdot L_{den} + \lambda_3 \cdot L_{str} \\ &s.t.\ \ \|X_\Delta - X\|_p \leq \epsilon \end{aligned} \tag{4}$$

where λ_1, λ_2, and λ_3 are weighting parameters to balance the contributions of the different loss terms, and ϵ is the perturbation bound under the L_p norm. Solving Eq. 4 with the dropout described in Section IV-C yields the mask M and pattern Δ of the sought adversarial patch.

Mask M consists of 0s and 1s. Eq. 4 is an MIP problem, which cannot be directly optimized with common optimizers such as *Stochastic Gradient Descent* (SGD) or Adam [31]. Such an MIP problem is solved elegantly with the l_p-Box ADMM [59] in SAPF [20]. Following RP_2 [19], we solve it with a two-stage method. In the first stage, we relax the binary constraint on M and allow it to take continuous values between 0 and 1, and then optimize Eq. 4 to obtain a fractional mask M. We then threshold M to obtain a binary mask by selecting the groups with the highest values. In the second stage, we fix the binary mask M and optimize Eq. 4 to determine the optimal pixel values of the adversarial patch. The performance of using the two solutions to DorPatch will be compared in Section VI-H.

A. Stage 1: Generation of Mask M

1) Optimization Using Transparency Mask: In this stage, we optimize Eq. 4 to generate mask M. As mentioned above, M is non-differentiable since it consists of 0s and 1s. To make M differentiable, we assume M is a *transparency mask* M_T with floating point values in $[0, 1]$, which is applied to a source image X as follow:

$$X_\Delta = (1 - M_T) \circ X + M_T \circ \Delta \tag{5}$$

Transparency mask M_T controls the merging transparency when the mask is applied to an image. If M_T takes values of only 1s and 0s, then Eq. 5 is equivalent to applying patch M to image X. When M is replaced with M_T, Eq. 4 is differentiable, and common optimizers such as SGD or Adam [31] can be applied to solve Eq. 4.

In our experiments, we use SGD as the optimizer and set λ_2 and λ_3 to 10^{-3}. The parameter λ_1 is adaptively adjusted as in FIA [26] to maintain adversarialness while minimizing the group lasso of the mask. Specifically, we first set λ_1, λ_2, and λ_3 to 0 to achieve adversarialness. Once adversarialness is reached, we enable λ_2 and λ_3 to their set values and adjust λ_1 as follows: if adversarialness is maintained, we multiply λ_1 by a factor (1.2 in our experiments) to focus more on reducing group lasso; if adversarialness is lost, we divide λ_1 by another factor (1.3 in our experiments) to focus more on reaching adversarialness.

2) Selecting Important Groups to Form Mask M: The magnitude of M_T in Eq. 5 represents the merging weight of pattern Δ of the adversarial patch. A pixel with a larger magnitude has a higher contribution from the adversarial patch in the merged value of the pixel. We use the magnitude of M_T to approximate the importance of each pixel for achieving adversarialness. The importance of a group is defined as the sum of all magnitudes of M_T in the group. We compute the importance for each group, select groups in descending order of importance until the patch budget is reached, and then set the selected groups to 1s and unselected groups to 0s to form mask M.

B. Stage 2: Generation of Pixel Values

Once M is determined, we enter the second stage to continue optimizing Eq. 4 to produce pixel values of the adversarial patch to generate. Since the pixel locations in the adversarial patch are fixed in this stage, L_{grp} and L_{den} in Eq. 4 are both fixed and thus can be removed from the optimization. As a result, optimizing Eq. 4 becomes optimizing the following equation in the second stage:

$$\begin{aligned} &\min_{\Delta} L_{adv} + \lambda_3 \cdot L_{str} \\ &s.t.\ \ \|X_\Delta - X\|_p \leq \epsilon \end{aligned} \tag{6}$$

In this stage, λ_3 is adjusted dynamically in the same way as λ_1 is adjusted in the first stage.

C. Generation of Untargeted Adversarial Patch

To evade PatchCleanser and make adversarially patched examples certifiable by PatchCleanser, the two-stage generation process described above is sufficient for targeted adversarial patch attacks but insufficient for untargeted adversarial patch attacks. This is because PatchCleanser employs prediction consistency when masking out an image at different locations to make a decision. For untargeted adversarial patch attacks, predicted labels when masking different locations may be different, albeit all of them fulfill the goal of a label different from the original one. This prediction inconsistency of masking different portions of an image makes it unlikely to evade PatchCleanser.

To address this inconsistent prediction problem, DorPatch adopts a mixture process that combines both untargeted and targeted adversarial patch generations to generate adversarial patches for untargeted adversarial patch attacks. To generate an untargeted adversarial patch, DorPatch splits the first stage into two substages: an untargeted substage and a target substage. In the untargeted substage, untargeted adversarial loss L_{adv}^{unt} is used for untargeted adversarial patch optimization. When it is sufficiently trained, DorPatch switches to the target substage to ensure consistent prediction. Specifically, DorPatch selects the majority label using a majority voting of the prediction results at different masked positions in the untargeted stage as the target label. It then switches to using targeted adversarial patch loss L_{adv}^{tar} to generate an adversarial patch to the target label for the remaining training of the first stage and for the whole training of the second stage. This mixture of untargeted and targeted adversary patch generation process ensures that the optimization process drives to an easy label while ensuring prediction consistency, a basic requirement for evading certifiable defenses such as PatchCleanser.

D. Supplementary Physical-World Patch Factors

The aforementioned adversarial patch generation is sufficient for digital-domain adversarial patch attacks. However, for physical-world attacks, more factors should be taken into consideration. First, the adversarial patch should be within the victim object. This is realized by using the bounding region of the victim object to restrict selected groups of pixels in the first stage to be within the region. Second, physical factors related to printing and photoshooting should also be taken into consideration during the optimization. This is realized with the following method.

Similar to existing physical adversarial patch attacks [18], [19], we apply *Expectation Over Transformation* (EOT) [19] to make DorPatch robust to affine and perspective transformations. These transformations are used to simulate different photoshooting distances and angles. Additionally, color jitter and blurring are employed to simulate various lighting conditions and color distortion in printing and photo-shooting.

Let B be the bounding region of the victim object, and let T denote all possible combinations of transformations. Let $t \in T$ be a random sample from T, i.e., a randomly sampled combination of transformations. The same optimization of Eq. 4 described above is executed for both stages except:

1) Mask M is bounded with B: $M \leftarrow M \cdot B$;
2) Patched example X_Δ is transformed by t into $t(X_\Delta)$ before feeding into the classification model;
3) L_{adv} is calculated as the expectation over T: $L_{adv}(X_\Delta) \leftarrow \mathbb{E}_{t\sim T} L_{adv}(t(X_\Delta))$.

VI. Experimental Evaluation

In this section, we conduct experimental evaluations in both the digital domain and the physical world to assess DorPatch's performance. This includes attacking state-of-the-art certifiable defense PatchCleanser [64] and empirical defenses and evaluating the perceptual quality of patched images. We also compare DorPatch with existing adversarial patch attacks and present an ablation study to examine the impacts of different modules and parameter settings in DorPatch on its performance.

A. Attack Settings and Baseline Attacks

Most existing adversarial patch attacks evaluate their performance using untargeted attacks. To facilitate comparison, we also utilize untargeted adversarial patch attacks as the default in our evaluation of DorPatch. Our evaluation encompasses various model architectures and datasets. The specific models and datasets employed in a particular evaluation depend on the available models for that evaluation. Detailed information about the models and datasets used in each evaluation can be found in the following respective evaluation descriptions.

The attack performance results reported in this section represent the average outcomes obtained from 1,000 samples randomly selected from the test set of a dataset that are classified correctly on the evaluated model.

DorPatch Settings. For density regularization, we divide an image into 8×8 equal-size regions. In image dropout, we randomly mask out a square area with the mask size ranging from 1.5% to 12% of the entire image, sampling 128 different masks per iteration. For the group lasso loss, we set the number of pixels in each group to 7×7. As the *Carlini & Wagner* attack (C&W) [7] has proven effective in generating powerful adversarial examples, we adopt the C&W loss as the adversarial loss L_{adv} in our experiments. The values of $\lambda_i, i \in 1, 2, 3$ in Eqs. 4 and 6 were described in Section V.

In our experiments, the maximum number of iterations for each stage is set to 5,000, and the learning rate is initially set to 0.01 and is decayed as follows. We halve the learning rate when the loss value of the dynamically adjusted loss term, i.e., the group lasso in the first stage and the structural loss in the second stage, no longer decreases, and stop early when the learning rate is smaller than 0.001. At the end of optimization, we consider the generation a failure if the adversarialness is not reached.

Baseline Attacks. The following four typical and state-of-the-art adversarial patch attacks are selected as the baseline attacks in our performance evaluation.

1) **LaVAN** [30]: It crafts a localized adversarial patch placed at a pre-fixed location (randomly chosen in our experiments). Its optimization problem can be formulated as

$$\min_{\Delta} L_{adv}(X_\Delta) \tag{7}$$

2) **LOAP** [46]: It crafts a localized adversarial patch by also optimizing the location through moving the patch in different directions.
3) **IAP** [3]: It utilizes an attention technique to identify the most critical area of the image for placing the patch and generates an inconspicuous patch with Adversarial Generative Networks (GAN). Its optimization problem can be expressed as

$$\begin{aligned} &\underset{G}{argmin} \max_{D} L_{adv}(X \oplus G(z)) \\ &+ \alpha L_{GAN}(G, D) \\ &+ \beta L_{rec}(X, X \oplus G(z)) + \gamma L_{tv}(G(z)) \end{aligned} \tag{8}$$

where G and D are the generator and discriminator of the GAN. G takes in a random noise z and outputs the generated pattern, i.e., $\Delta = G(z)$. L_{GAN} and L_{rec} are the adversarial loss and the reconstruction loss for training GAN. L_{tv} is the total variance loss for smoothing the generated patch.
4) **RP**$_2$ [19]: It uses a two-stage method to generate adversarial patches. In the first stage, it employs the L_1 regularization to generate a sparse perturbation located in most vulnerable regions. The optimization problem in the first stage is as follow,

$$\min_{\Delta} L_{adv}(X_\Delta) + \lambda \|\Delta\|_1 \tag{9}$$

In the second stage, it solves the same optimization problem as in LaVAN [30] with a mask positioned on the vulnerable regions identified from the first stage. In our experiments, we set the minimum region to 7×7, the same size as a group in DorPatch. RP$_2$ is the only existing distributed patch attack.

In our experiments, each baseline attack is executed with its default setting, except for the explicitly stated settings above. For a fair comparison, we set the maximum number of iterations to 10,000 and adopt the same initial learning rate for each baseline attack. We also adopt the same learning rate decaying strategy on the adversarial loss for all baseline attacks, except for the first stage of RP$_2$. For the first stage of RP$_2$, the decaying strategy is applied with respect to the L_1 norm term.

B. Attacking Performance against PatchCleanser

We first assess the attack performance against the state-of-the-art certifiable defense, PatchCleanser. In our experiments, we utilize the open-source code and pre-trained ResNet models from its official GitHub repository [62]. These models are trained on three widely used image classification datasets: ImageNet [12], CIFAR10 [32], and CIFAR100 [32], using the *cutout* technique [13] as data augmentation for enhancing robustness to masking occlusions for higher certifiable accuracy. PatchCleanser's default settings are used in our experiments unless specified otherwise.

PatchCleanser primarily focuses on employing a single square mask to mask out a portion of an image at a time for prediction. For a given setting, PatchCleanser has a maximum size for such a localized square patch that it can protect. This maximum size is referred to as the *mask size* of PatchCleanser, which indicates a specific setting of PatchCleanser.

The *robust accuracy* of the model, which measures the model's prediction accuracy under various attacks, both with and without the defense of the primary PatchCleanser, is presented in Tables III, IV, and V for ImageNet, CIFAR10, and CIFAR100, respectively. These tables display results for different PatchCleanser mask sizes and attacks' patch budgets. Robust accuracy is important because it evaluates the model's performance in the presence of adversarial conditions, ensuring its reliability and stability. For IAP, we present experimental results only for the more practical dataset ImageNet and ignore more toy-like datasets CIFAR10 and CIFAR100 since IAP requires training a generative model for each test image, which takes a prohibitive amount of time to finish all experiments on one dataset.

The third column of these tables exhibits that all the attacks except IAP can substantially degrade the robust accuracy when no defense is applied. The robust accuracy under IAP attack is significantly higher than other attacks on ImageNet, ranging from 36.7% to 25.4% when the patch budget ranges from 3% to 12% of the image size.

However, when PatchCleanser is applied, as indicated by the fourth to sixth columns in these tables, the robust accuracy raises to a very high level, close to clean accuracy (the accuracy when there is no attack), for attacks LaVAN, LOAP, and IAP, but remains reasonably low for RP$_2$ and very low or close to 0% for DorPatch at 3% or 6% and above patch budget. By correlating with the fact that only RP$_2$ and DorPatch are distributed patch attacks among these attacks, we conclude that a distributed patch helps lower PatchCleanser's robust accuracy. Since the robust accuracy under RP$_2$ is significantly higher than that under our DorPatch attack, we conclude that our image dropout is much more effective in reducing PatchCleanser's robust accuracy than making an adversarial patch distributed. This is because image dropout makes an adversarial patch robust to PatchCleanser's masking occlusions while a only distributed patch cannot.

In addition to robust accuracy, another important performance metric is the *certified rate of patched examples* (*CRPE*), which is defined as the ratio of adversarially patched examples that are both successful in misprediction (i.e., predict to a label different from the original one for untargeted attacks or the target label for targeted attacks) and the prediction results are certifiable robust by a certifiable defense such as PatchCleanser. CRPE is important since it measures the probability that the false prediction of an adversarially patched example can be successfully certified by the certifiable defense to provide a false sense of trust.

The *certified rates of patched examples* (*CRPEs*) of the attacks are also presented in Tables III, IV, and V for ImageNet, CIFAR10, and CIFAR100, respectively. From these tables, we can see that the CRPE of each baseline attack is or close to 0%, even for a large patch budget. On the other hand, DorPach's CRPE is high even for 3% patch budget and increases with an increasing patch budget. DorPach's CRPE is comparable to and mostly higher than the certified rate of benign samples that PatchCleanser reports in [64] when the patch budget is the same as the mask size except for CIFAR10 when the patch

budget is 3%.

Note that our DorPatch is agnostic to the defense setting of PatchCleanser. We can conclude from the above experimental results that DorPatch exhibits high effectiveness in attacking PatchCleanser, in terms of both robust accuracy and CRPE.

TABLE III. ROBUST ACCURACY AND *CERTIFIED RATE OF PATCHED EXAMPLES* (CRPE) ON IMAGENET FOR ATTACKS WITH VARIOUS PATCH BUDGETS, WITH/WITHOUT PATCHCLEANSER DEFENSE (SINGLE MASK) USING DIFFERENT MASK SIZES. PB:PATCH BUDGET. MS: MASK SIZE.

PB	Attack	Robust Accuracy (in %)				CRPE (in %)		
		without Defense	PatchCleanser (MS)			PatchCleanser (MS)		
			3%	6%	12%	3%	6%	12%
3%	DorPatch	4.4	10.2	9.8	11.2	49.8	44.9	38.1
	LaVAN	6.2	89.1	90.6	86.3	0.0	0.0	0.0
	LOAP	4.7	89.5	89.9	86.4	0.0	0.0	0.0
	IAP	36.7	80.9	78.1	78.1	0.0	0.0	0.0
	RP_2	0.0	56.4	58.4	63.0	0.8	0.4	0.0
6%	DorPatch	0.8	1.2	1.2	1.2	80.9	69.7	57.6
	LaVAN	1.2	82.8	86.7	85.6	0.0	0.0	0.0
	LOAP	0.4	82.9	86.8	84.8	0.0	0.0	0.0
	IAP	27.0	71.5	71.5	71.5	0.0	0.0	0.0
	RP_2	0.0	25.8	38.1	43.2	2.7	0.4	0.4
12%	DorPatch	0.8	1.0	1.0	1.0	87.1	83.1	75.8
	LaVAN	0.0	76.2	78.1	81.6	0.0	0.0	0.0
	LOAP	0.0	77.4	78.9	78.9	0.0	0.0	0.0
	IAP	25.4	61.1	63.2	63.7	0.0	0.0	0.0
	RP_2	0.0	16.7	19.8	22.2	5.5	2.5	0.8

TABLE IV. ROBUST ACCURACY AND *CERTIFIED RATE OF PATCHED EXAMPLES* (CRPE) ON CIFAR10 FOR ATTACKS WITH VARIOUS PATCH BUDGETS, WITH/WITHOUT PATCHCLEANSER DEFENSE (SINGLE MASK) USING DIFFERENT MASK SIZES. PB:PATCH BUDGET. MS: MASK SIZE.

PB	Attack	Robust Accuracy (in %)				CRPE (in %)		
		without Defense	PatchCleanser (MS)			PatchCleanser (MS)		
			3%	6%	12%	3%	6%	12%
3%	DorPatch	7.6	13.2	13.2	12.7	40.6	37.6	33.0
	LaVAN	4.9	98	95.6	94.3	0.0	0.0	0.0
	LOAP	5.3	96.8	96.4	92.7	0.0	0.0	0.0
	RP_2	0.0	77.7	78.5	80.2	0.8	0.4	0.4
6%	DorPatch	0.0	0.0	0.0	0.0	78.8	68.2	60.6
	LaVAN	0.8	93.1	94.3	92.7	0.0	0.0	0.0
	LOAP	0.8	93.5	93.5	93.1	0.0	0.0	0.0
	RP_2	0.0	60.7	67.6	66.8	1.21	1.21	0.8
12%	DorPatch	0.0	0.0	0.0	0.0	90.9	86.4	76.3
	LaVAN	0.0	86.6	92.3	93.5	0.0	0.0	0.0
	LOAP	0.0	85.8	92.7	92.7	0.0	0.0	0.0
	RP_2	0.0	42.5	44.9	52.2	1.6	1.6	0.4

TABLE V. ROBUST ACCURACY AND *CERTIFIED RATE OF PATCHED EXAMPLES* (CRPE) ON CIFAR100 FOR ATTACKS WITH VARIOUS PATCH BUDGETS, WITH/WITHOUT PATCHCLEANSER DEFENSE (SINGLE MASK) USING DIFFERENT MASK SIZES. PB:PATCH BUDGET. MS: MASK SIZE.

PB	Attack	Robust Accuracy (in %)				CRPE (in %)		
		without Defense	PatchCleanser (MS)			PatchCleanser (MS)		
			3%	6%	12%	3%	6%	12%
3%	DorPatch	5.3	6.7	6.2	7.2	61.7	49.3	39.7
	LaVAN	1.6	61.3	62.5	62.1	0.0	0.0	0.0
	LOAP	2.3	58.2	59.4	59.0	0.0	0.0	0.0
	RP^2	0.0	40.2	43.8	43.0	0.4	0.0	0.0
6%	DorPatch	0.0	0.5	0.5	0.5	85.2	75.6	64.1
	LaVAN	0.0	55.9	57.4	59.8	0.0	0.0	0.0
	LOAP	0.4	49.2	50.4	50.4	0.0	0.0	0.0
	RP^2	0.0	29.3	29.7	33.6	0.8	0.8	0.0
12%	DorPatch	0.0	0.0	0.0	0.0	92.8	87.6	79.0
	LaVAN	0.0	41.8	48.4	54.7	0.0	0.0	0.0
	LOAP	0.0	37.5	39.8	43.8	0.0	0.0	0.0
	RP^2	0.0	16.4	19.1	19.5	1.6	1.2	0.4

TABLE VI. TRANSFORMATIONS AND THEIR PARAMETER VALUES USED IN EOT FOR DORPATCH IN OUR PHYSICAL-WORLD ATTACK

Transformation	Factor	Random Sampling Range
Color Jitter	Brightness	[0.8, 1.2]
	Contrast	[0.8, 1.2]
	Saturation	[0.8, 1.2]
	Hue	[-0.1, 0.1]
Affine	Degrees	(-5, 5)
	Horizontal Translation	Width×(-0.05, 0.05)
	Vertical Translation	Height×(- 0.05, 0.05)
	Scale	[0.8, 1.2]
Perspective	Distortion Scale	0.05
Gaussian Blur	Kernel Size	(3, 3)
	Sigma	(0.1, 2.0)

C. Perceptual Quality

Inconspicuousness is one of the desirable properties of an adversarial patch, as described in Section III-C. We have compared the inconspicuousness of different attacks. Fig. 3 shows adversarially patched examples generated by various attacks. Traditional patch attacks like LaVAN, LOAP, and RP_2, which lack any perceptual constraint in crafting adversarial patches, appear conspicuously visible and suspicious, as illustrated in Columns (b)-(d) in Fig. 3. In Column (e), IAP, by utilizing the discriminators of generative models, produces patches that harmonize with the background images at first glance. However, upon closer examination, the generated patch can still be detected due to the abnormal lines and blur effect near the eye region of the showcased samples. In contrast, as depicted in Column (f), the patches generated by our DorPatch remain inconspicuous even with the aid of masks indicating their locations in Column (g). Our DorPatch can produce the most imperceptible perturbations among all considered patch attacks.

Additionally, Column (h) in Fig. 3 shows the saliency maps of DorPatch's adversarially patched examples given by GradCAM [52], which simulate human attention. The hotter the color, the more attention it gets. From the attention maps, we can conclude that most portions of DorPatch's adversarial patch fall within low attention regions, which proves the inconspicuousness of the adversarial patch.

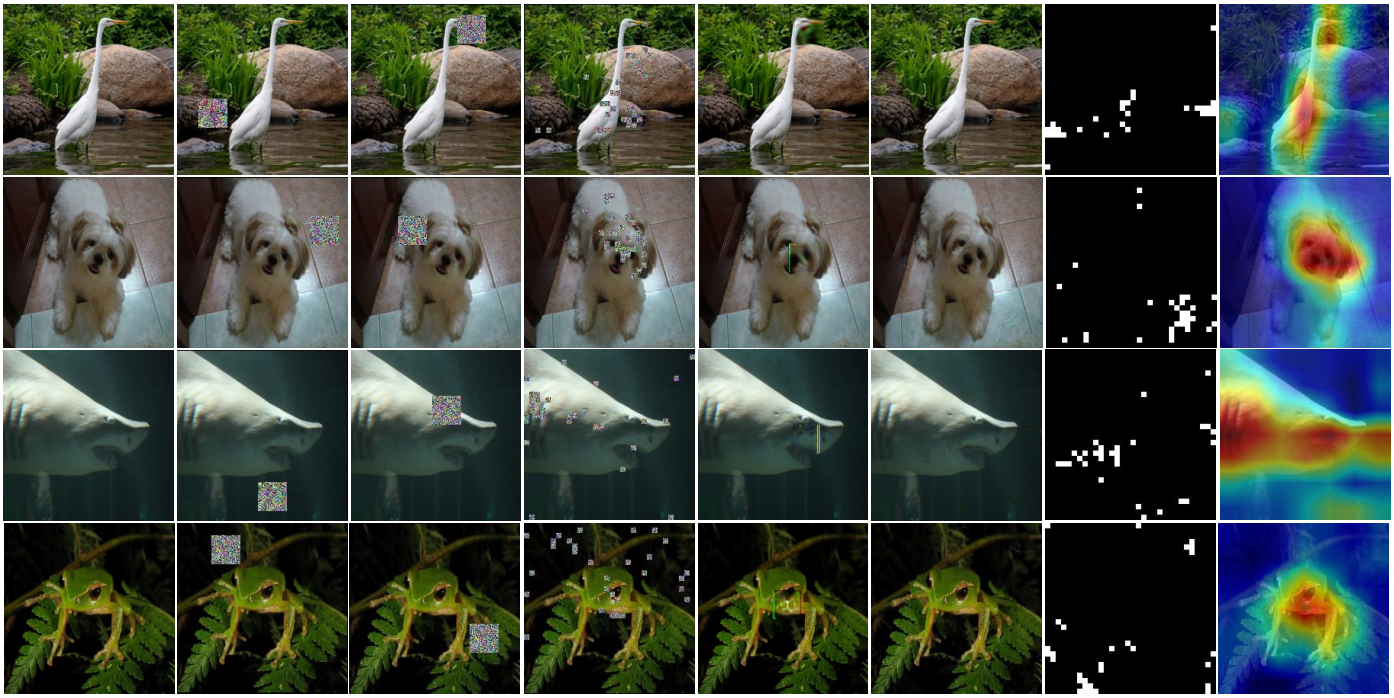
D. Physical-world Attack Performance

In this section, we evaluate DorPatch's physical-world attack performance against the PatchCleanser defense. We employ the pretrained ImageNet model from the PatchCleanser authors and set DorPatch's patch budget to 12%. Adversarial patches are generated following the method in Section V-D. The EOT [19] parameters for our experiments are listed in Table VI.

We choose IAP [3], the most inconspicuous physically realizable patch attack among all existing adversarial patch attacks, as our baseline for evaluating physical-world attack performance. Our evaluation primarily focuses on untargeted attacks.

1) Physical Experimentation Process: The physical-world experiments are conducted as follows.

We take a sample that is classified correctly and certifiable by PatchCleanser with different mask sizes from the street sign category of ImageNet, print it on photographic paper using a



(a) Origin (b) LaVAN (c) LOAP (d) RP_2 (e) IAP (f) DorPatch (g) Mask (h) Saliency Map

Fig. 3. Perceptual quality of various adversarial patch attacks. Column (g) shows the masks of our DorPatch's patched examples. Column (h) shows the saliency maps of our DorPatch's patched examples given by GradCAM [52], which simulate human attention.

Canon G580 printer, and paste it onto a wall. The resulting physical object serves as a victim object for our physical attack evaluation. In our experiments, we generate two victim objects.

After generating an adversarial patch for a victim object in the digital domain, we print it out, cut off each piece of the adversarial patch, and then paste them onto the victim object to produce an adversarially patched physical object.

We use an iPhone 13 Pro to capture sequences of photos of the physical object under various conditions, with a resolution of 1920×1080 and a frequency of 30 frames per second. These conditions include different lighting (intensity and color temperature), shooting angles (up to 30 degrees in each direction), and shooting distances of 0.4 m and 0.7 m. The camera moves at a constant speed while capturing the photo sequences.

For each frame in a sequence, we crop an image from the center with a fixed size appropriate for the shooting distance, as shown by the red squares in Fig. 4 (f). The image is then resized to the input size of the classification model and fed into the model for prediction.

For each experimental evaluation, we draw 1,000 frames uniformly from the photo sequences and use them to assess performance. The physical-world attack evaluation is illustrated in Fig. 4, and the evaluation results are presented in Table VII.

2) Perceptual Quality: As shown in Fig. 4 (c), IAP generates a square patch that significantly blurs the word "TIME", while our DorPatch creates distributed patches resembling green fluorescent stains scattered around the sign's edge area. DorPatch's patches are much less conspicuous than those of IAP.

3) Attacking Performance: Table VII shows that without any attack, the clean accuracy is high. When subjected to the IAP or DorPatch attack, the robust accuracy drops significantly for both attacks: to 0% under our DorPatch attack and 16.0% under the IAP attack. IAP's robust accuracy is much higher than ours, indicating that our DorPatch is a more effective patch attack than IAP. When the PatchCleanser defense is applied, the robust accuracy under IAP attack restores to a level close to clean accuracy, suggesting that IAP is ineffective in evading PatchCleanser. In contrast, the robust accuracy under our DorPatch attack remains very low, at 2.4% or below, demonstrating DorPatch's effectiveness in evading PatchCleanser in physical-world attacks as well.

TABLE VII. ROBUST ACCURACY (IN %) ON IMAGENET UNDER PHYSICAL-WORLD ATTACKS OF DORPATCH AND IAP WITH 12% PATCH BUDGET, WITH/WITHOUT PATCHCLEANSER DEFENSE (SINGLE MASK) UNDER DIFFERENT MASK SIZES. MS: MASK SIZE.

Attack	without Defense	PatchCleanser (MS)		
		3%	6%	12%
No Attack	100.0	100.0	100.0	100.0
IAP	16.0	91.4	90.4	66.4
DorPatch	0.0	2.4	1.4	2.1

E. Targeted Attack Performance

All the above evaluations are based on untargeted attacks for comparison with existing adversarial patch attacks since most of them focus on reporting untargeted attack performance. One would wonder: how does DorPatch perform when conducting targeted attacks, generally considered more challenging than untargeted attacks? We evaluate DorPatch's target attack performance against PatchCleanser on ImageNet in this subsection.

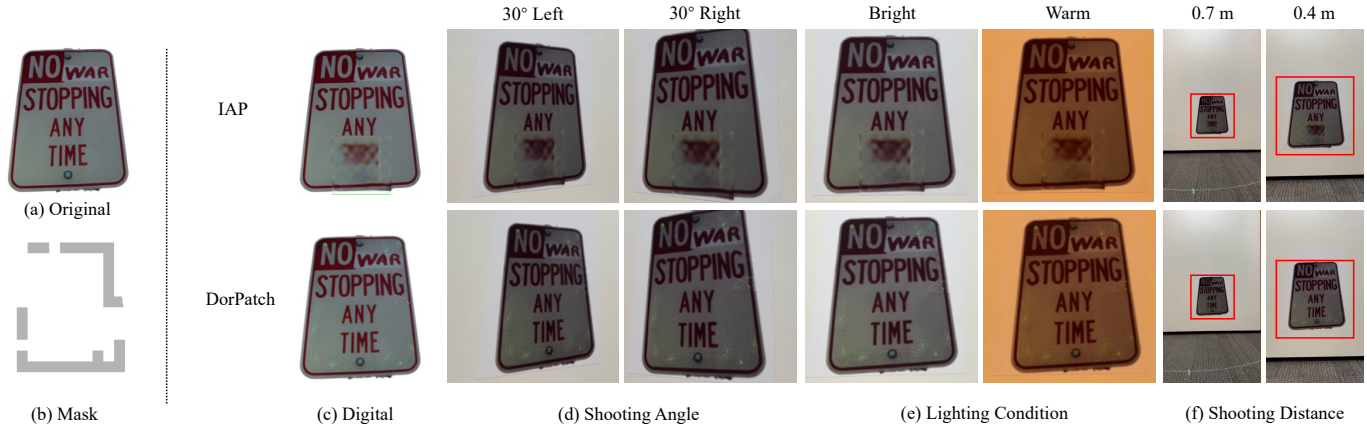


Fig. 4. Illustration of physical-world evaluation. (a) Original victim object. (b) DorPatch mask to help locate the patch. Columns (c)-(f): top row shows IAP results, bottom row shows DorPatch results. (c) Generated adversarial patched examples. (d) and (e) Captured frames and their cropped images under varying angles (up to 30°) and lighting conditions (intensity and temperature). (f) Red squares highlight cropped areas in captured frames at different shooting distances.

In our experiments, we randomly sampled 1000 correctly classified samples from the test set. For each selected sample, we randomly selected a target label different from the original prediction. These selected samples were used in evaluating DorPatch's target attack performance. The patch budget was set to 3% in our experiments.

1) Attacking Performance against PatchCleanser: By adjusting the perceptual constraint ϵ in Eqs. 4 and 6 to a reasonable level, a targeted DorPatch attack can achieve comparable attacking performance to that of an untargeted DorPatch attack. Specifically, when attacked by DorPatch as a targeted attack, the robust accuracy drops to 0.5% without defense. When PatchCleanser with a mask size of 3%, 6%, and 12% is applied, the robust accuracy drops to 1.1%, 0.5%, and 0.5%, respectively, and the CRPEs are 46.6%, 22.3%, and 29.6%, respectively.

Unlike untargeted attacks, a targeted attack is considered successful only when the generated adversarial example is misclassified into the target category. Therefore, we also report the *Attacking Success Rate* (*ASR*) for a targeted attack, which is the ratio of patched examples that successfully reach their respective target categories. For DorPatch used as a targeted attack, the ASR is 91.0% without defense. The ASR consistently remains at 91.0% when PatchCleanser with a mask size of 3%, 6%, and 12% is applied.

2) Perceptual Quality: Since IAP [3] is the most inconspicuous patch attack among all existing adversarial patch attacks, we choose it as the baseline to compare the perceptual quality of adversarial patch attacks. Fig. 5 presents a side-by-side comparison of both targeted and untargeted adversarial examples generated by DorPatch and IAP, using the same source images from ImageNet and the same target labels for targeted attacks. Comparing columns (b) with (c) and (d) with (e), we observe that targeted attacks generally result in more noticeable visual effects than their untargeted counterparts. This holds true for both DorPatch and IAP. This can be explained by the fact that targeted attacks are more challenging than untargeted attacks, and a larger patch budget is generally required to achieve adversarialness, resulting in more perceptual distortions. For DorPatch, in terms of the average L_2 norm distortion between a patched example and its source sample, DorPatch as a targeted attack is 3.88 times that of DorPatch as an untargeted attack on ImageNet.

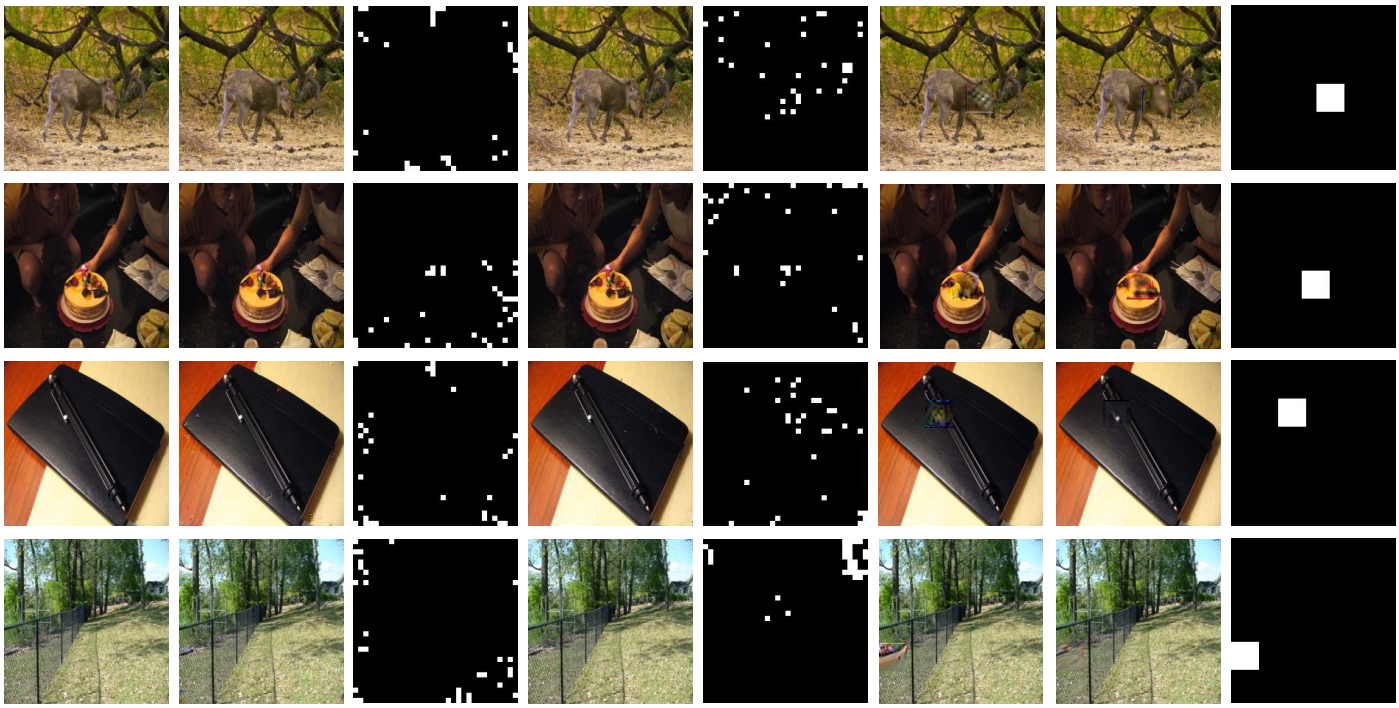
On the other hand, our DorPatch typically yields superior perceptual quality when compared to IAP. This can be mainly attributed to our designed structural loss, which encourages generated patches to be placed in complex regions and promotes the construction of continuous and smooth structures. As a result, our DorPatch's targeted attacks can even achieve perceptual quality that is comparable to or even better than IAP's untargeted attacks.

F. Attack Performance Against Empirical Defenses

We have evaluated the attack performance against the certifiable defense PatchCleanser. It is natural to question the effectiveness of the DorPatch attack against empirical defenses. In this subsection, we explore the ability of our DorPatch to evade various empirical defenses designed to counter adversarial patch attacks.

1) Evading Local Gradients Smoothing: Local Gradients Smoothing (LGS) [45] deters patch attacks by suppressing large image gradients in an image. LGS is evaluated only on ImageNet in its paper. We also evaluate LGS on ImageNet. We take the Pytorch implementation of LGS [48], with the default setting in LGS's original paper, and adopt a pre-trained DenseNet110 model from the Pytorch TorchVision model zoo [57]. The clean accuracy of the model on the test samples drops by 4.0% when it is defended by LGS.

In this evaluation, we choose LaVAN and LOAP as the baselines. The patch budget is set to 3% for all attacks. The evaluation results are shown in Table VIII. The table demonstrates that, when the model is defended by LGS, the robust accuracy under LaVAN and LOAP attacks is restored to a very high level, at 99.8% and 99.9%, respectively. However, it remains relatively low at 35.9% under the DorPatch attack. This robust accuracy indicates that LGS is more effective in defending against DorPatch than PatchCleanser, primarily because DorPatch is not designed to evade LGS. To achieve better inconspicuousness, DorPatch places adversarial patches



(a) Origin (b) DorPatch-targeted (c) DorPatch-untargeted (d) IAP -targeted (e) IAP -untargeted (f) IAP -mask

Fig. 5. Perceptual quality: targeted v.s. untargeted attacks for DorPatch and IAP. The right columns in (b)-(c) show the masks to help locate the patches.

TABLE VIII. ROBUST ACCURACY (IN %) ON IMAGENET UNDER ATTACKS WITH 3% PATCH BUDGET, WITHOUT DEFENSE AND WHEN THE MODEL IS DEFENDED BY LGS OR DW.

Defense	Attack			No Attack
	DorPatch	LaVAN	LOAP	
No Defense	3.1	7.4	4.7	100.0
LGS	35.9	99.8	99.9	96.0
DW	19.6	98.7	98.4	90.9

in complex regions with large image gradients in an image, making the patch more susceptible to LGS's suppression. We can improve DorPatch's attacking effectiveness against LGS by placing a patch in regions with smaller image gradients, slightly sacrificing perceptual quality.

2) Evading Digital Watermarking: Digital Watermarking (DW) [25] defends against patch attacks by first constructing a sensitivity-based saliency map of an input image to identify unusually dense, highly sensitive regions and then masking them out for prediction. DW is evaluated solely on ImageNet in its paper [25]. We also evaluate DW on ImageNet. Similar to LGS, we choose LaVAN and LOAP as the baselines. The patch budget is set to 3% for all attacks.

There is no official implementation available. We have implemented DW by constructing the saliency map using guided backpropagation [55] from a popular GitHub repository [21]. The resulting performance aligns closely with that reported in the original paper. We adopt a pre-trained DenseNet110 model from the Pytorch TorchVision model zoo [57] for this experiment.

The evaluation results are shown in Table VIII. When the model is defended by DW, the clean accuracy of the model on the test samples drops significantly (by 9.1%). The robust accuracy under LaVAN and LOAP attacks is restored to 98.7% and 98.4%, respectively. In contrast, DorPatch remains highly effective in bypassing DW, with the robust accuracy as low as 19.6%. This also proves our DorPatch's robustness to partial removal.

G. Attack Performance Against Adaptive Defenses

1) PatchCleanser Using Multiple Masks: PatchCleanser can be extended to defend against a distributed adversarial patch comprising multiple separated subpatches by applying multiple masks to mask out multiple regions simultaneously. It needs to search all possible combinations of subpatches to fully cover the distributed patch, i.e., each subpatch is covered by a mask. Let k be the number of distinct locations for placing a single mask on an image, and let Z be the number of subpatches. PatchCleanser needs to search for $\binom{k}{Z}$ combinations for single-round masking and $\binom{\binom{k}{Z}}{2}$ combinations for two-round masking. Each combination requires a model inference.

For the default value of $k = 36$, PatchCleanser needs 36, 630, and 7,140 model inferences for single-round masking and 630, 1.98×10^5, and 2.55×10^7 model inferences for two-round masking when Z increases from 1 to 3.

The number of model inferences explodes exponentially as Z increases. Even for $Z = 2$, two-round masking for certifying a single sample requires 1.98×10^5 model inferences, accumulating to a prohibitive number for completing all experiments on a single dataset. Consequently, we conducted experiments for single-round masking of PatchCleanser to obtain its robust accuracy when $Z = 2$ on ImageNet. The

TABLE IX. ROBUST ACCURACY (IN %) OF PATCHCLEANSER WITH DUAL MASKS ON IMAGENET UNDER DORPATCH ATTACK. VALUES IN PARENTHESES INDICATE THE ROBUST ACCURACY IMPROVEMENT FROM PATCHCLEANSER'S SINGLE PATCH SETTING.

Patch Budget	Mask Size of PatchCleanser 3%	6%	12%
3%	16.3 (+6.1)	15.6 (+5.8)	15.6 (+4.4)
6%	3.1 (+1.9)	3.5 (+2.3)	4.3 (+3.1)
9%	2.4 (+1.4)	2.4 (+1.4)	3.2 (+2.2)

TABLE X. ROBUST ACCURACY (IN %) UNDER THE DORPATCH ATTACK WITH DIFFERENT PATCH BUDGETS ON CIFAR10

Model		Patch Budget (in %)			
Arch.	Training Type	1.5	3	6	12
WRN28-4	Normal	10.8	0.6	0.7	0.1
	Adv. Trained	70.7	57.3	31.4	13.0
ResNet110	Normal	11.9	2.3	1.2	0.4
	Adv. Trained	64.7	29.6	7.0	0.9

results are presented in Table IX, with values in parentheses indicating the robust accuracy improvement from the single mask case shown in Table III. The table shows that robust accuracy slightly improves when dual masks are applied in PatchCleanser but remains low. The robust accuracy decreases when the patch budget of DorPatch increases. We can conclude that DorPatch can still effectively evade PatchCleanser using dual masks.

2) Adversarial Training: Adversarial training enhances a model's robustness by training it with adversarial examples generated using a specific method. It is widely used as an effective defense against adversarial attacks. In this subsection, we evaluate the performance of our DorPatch on adversarially trained models on CIFAR10. For this evaluation, we use pre-trained robust models from the open-sourced repositories of Hydra [29], [51] and DOA [60], [61]. Specifically, the robust WRN28-4 model from Hydra is adversarially trained using a PGD attack with 50 steps and an 8/255 L_∞ budget, while the robust ResNet110 model from DOA is trained using a rectangular occlusion attack with an 11×11 rectangle (patch budget=12%). The experimental results of our DorPatch with different patch budgets are shown in Table X.

The results demonstrate that even when the model is adversarially trained with a mismatching adversarial attack (i.e., PGD attack or the rectangular occlusion attack), it can improve adversarial robustness against our DorPatch to a certain degree, especially when the patch budget is low. For example, for a patch budget of 1.5%, the robust accuracy on WRN28-4 increases from 10.8% for normal model training to 70.7% when the model is adversarially trained. However, with increasing patch budget, the robust model can still be compromised. The robust accuracy drops to 13.0% for adversarially trained WRN28-4 and to 0.9% for adversarially trained ResNet110 when the patch budget increases to 12%.

We expect that adversarial training with adversarial examples generated with DorPatch can be more effective but the training cost is high. More adaptive defenses will be discussed in Section VII-C.

H. Performance Comparison with SAPF's Solution

In this subsection, we compare the performance of DorPatch when our two-stage solution and the L_p-Box ADMM [59] used in SAPF [20] are applied to solve DorPatch's MIP optimization problem (see Eq. 4). The L_p-Box ADMM is based on the equivalence between a discrete constraint space and the intersection of two continuous constraints: a box constraint and a L_p-sphere constraint. Using this property, SAPF replaces the non-differentiable location mask with two additional continuous variables and solves the MIP optimization problem by employing the ADMM method and the gradient descent algorithm to alternatively optimize the locations and the pixel values of the patch to generate.

TABLE XI. ROBUST ACCURACY AND CRPE OF DORPATCH WITH 12% PATCHES ON IMAGENET UNDER PATCHCLEANSER WITH VARIOUS MASK SIZES (MS)

Solution	Robust Accuracy (in %) without Defense	PatchCleanser (MS) 3%	6%	12%	CRPE (in %) PatchCleanser (MS) 3%	6%	12%
Two-stage	0.8	1.0	1.0	1.0	87.1	83.1	75.8
SAPF	1.2	2.5	2.5	2.5	58.0	37.0	16.1

Our two-stage solution optimizes the mask and the pattern separately in two stages. In the first stage, we relax the binary mask M to a transparency mask M_T with floating point values in [0, 1] and optimize Eq. 4 to obtain a fractional mask. From this fractional mask, we select the groups with the highest masking values to form the binary mask M with a given patch budget. In the second stage, we optimize the pattern Δ using Eq. (6) with the binary mask M determined in the first stage.

Table XI shows the experimental results of applying both solutions in DorPatch on ImageNet using a 12% patch budget. We can see from the table that both solutions achieve comparable low robust accuracy under PatchCleanser defense with different mask sizes. However, the L_p-Box ADMM has lower CRPEs than our two-stage solution (higher CRPE means better attacking performance). This may be attributed to the following reasons:

- The L_p-Box ADMM solution has many more variables and hyperparameters in the optimization problem, making it harder to balance the competitive relationship between the adversarialness loss and other losses and Lagrange multiplier terms.
- The dual variables introduced by ADMM are hard to optimize or tune when combined with image dropout.
- ADMM can be very slow to converge to high accuracy [5].

These results suggest that our two-stage solution, although less elegant as the L_p-Box ADMM solution, is simpler yet empirically more effective.

I. Ablation Study

It is important to know the impact of each component of our DorPatch on its performance. We conducted an ablation study to evaluate DorPatch's performance on ImageNet in attacking PatchCleanser under the single mask setting and LGS in the digital domain by removing different components of DorPatch. The patch budget was set to 3% in our experiments.

TABLE XII. ABLATION STUDY OF DENSITY REGULARIZATION AND IMAGE DROPOUT WITH PATCHCLEANSER ON IMAGENET USING A 3% PATCH BUDGET. MS: MASK SIZE.

Attack	Robust Accuracy (in %)				CRPE (in %)		
	without	PatchCleanser (MS)			PatchCleanser (MS)		
	Defense	3%	6%	12%	3%	6%	12%
DorPatch	4.4	10.2	9.8	11.2	49.8	44.9	38.1
−Density Reg.	4.7	9.0	10.9	10.9	47.5	38.5	26.9
−Img Dropout	4.7	86.7	89.8	89.1	0.0	0.0	0.0

TABLE XIII. ABLATION STUDY OF STRUCTURAL LOSS WITH LGS ON IMAGENET USING A 3% PATCH BUDGET

Configuration	Fixed λ_3		Adaptive λ_3	No L_{str}
	10^{-3}	10^{-4}		
Robust Acc. (in %)	37.9	50.3	35.9	75.5

1) Ablation of Density Regularization & Image Dropout: The attack performance after removing density regularization or image dropout is presented in Table XII when PatchCleanser is applied under the single mask setting. We observe that CRPE slightly decreases by 2.3%, 6.7%, and 11.2% for mask sizes of 3%, 6%, and 12%, respectively, when the density regularization is removed. The decline in CRPEs is more noticeable when the mask size of PatchCleanser is larger, indicating the importance of density regularization in boosting DorPatch's CRPE, especially for large mask sizes.

Table XII clearly shows that the absence of image dropout severely impairs attack effectiveness in terms of both robust accuracy and CRPE when facing the PatchCleanse defense. Without the image dropout, the generated adversarial patch cannot withstand partial occlusion, allowing PatchCleanser to successfully restore robust accuracy to above 86.7%, while CRPE drops to 0%. The experimental results highlight the importance of robustness to partial occlusions when attacking PatchCleanser.

2) Ablation of Structural Loss: We utilize LGS [45] for the ablation study of the structural loss, as it effectively impacts the structural loss by suppressing large image gradients. To investigate the effect of the structural loss in DorPatch on bypassing LGS, we vary its weighting value, i.e., λ_3 in Eq. 4 and Eq. 6, and fix the value in both stages to adjust the contribution of the structural loss in generating an adversarial patch. When λ_3 is set to 0, the structural loss is completely removed. We set the patch budget to 3% in this experiment. The evaluation results are shown in Table XIII. The table reveals that the robust accuracy increases from 35.9% to 75.5% when the structural loss is removed. When λ_3 increases from 10^{-4} to 10^{-3}, i.e., with more emphasis on the structural loss, the robust accuracy decreases from 50.3% to 37.9%.

These results demonstrate that the structural loss facilitates the construction of continuous and smooth structures, making the generated patch more resistant to neutralization by LGS.

VII. DISCUSSION

A. Limitations

DorPatch has the following limitations.

1) Black-box Transferability: We assume white-box access to the model to be attacked, and the performance results reported in Section VI are under this assumption. A question naturally arises: how effective can DorPatch be under the black-box model setting? We conduct an experiment on ImageNet with a 6% patch budget, and the result shows that the transfer success rate from a whitebox DenseNet121 model to a blackbox ResNet50 model for DorPatch as an untargeted attack is 45.3%. This can be explained by the fact that DorPatch fully optimizes the attacking effectiveness of its generated adversarial patch (i.e., to fulfill the desirable property of being fully optimized), resulting in the generated adversarial patch overfitting to the model used in generating the adversarial patch, and thus reduces the black-box transferability. We plan to employ model ensembling [40] to strengthen its black-box transferability.

2) Deployability: Another question that might arise is: how does the distributed nature of DorPatch's patch affect the difficulty of deploying a physical attack? Unlike traditional methods that use a single localized patch, DorPatch makes it harder to deploy a physical attack, because it requires cutting and pasting multiple pieces of the adversarial patch and preserving their spatial locations during the process. For example, in our physical-world evaluation in Section V-D, we took about 5 minutes to set up the attack for IAP but around 30 minutes for DorPatch, most of which was used for cutting the patch and other preparations. The actual sticking of the patch onto the target only took 1 to 2 minutes. Despite the increased difficulty, DorPatch is still practically feasible for physical attacks.

B. Restriction on Patch Budget

One may ask: instead of applying a restricting patch budget to perturb a small portion of pixels, why not simply perturb the whole surface of the victim object like ShapeShifter [8]? There are several reasons for us to restrict adversarial patches within a given patch budget. First, we want to apply the same patch budget for all attacks, DorPatch and baseline attacks, for a fair comparison. Second, perturbing the whole surface of the victim object like ShapeShifter [8] requires a much larger patch budget. Similar to adversarial examples, this can be hard to be deployed in the physical world, unless the whole image of an adversarial example is printed out and pasted on the physical object to launch a physical-world attack, which works only for static and flat objects but can be hardly deployable for uneven or live objects (e.g., human faces). DorPatch can still deploy physical-world attacks for uneven or live objects.

C. Possible Countermeasures

With its attacking power, DorPatch poses a serious threat to practical usage of DNN models. Our work calls for a great effort to develop effective countermeasures to thwart DorPatch. One countermeasure is to apply adversarial training with adversarial examples generated with DorPatch to defend against DorPatch, which should be effective but the training cost is high. More effective countermeasures should be developed.

It might be challenging to develop an effective defense against our DorPatch for general cases. For some special scenarios, such as recognizing traffic sign images containing words, we may apply OCR in combination with the classification model to classify traffic signs, which should be

able to disable DorPatch unless both the OCR model and the classification model are used by DorPatch to generate the adversarial patch.

When there are no effective countermeasures for general cases, the best protection against DorPatch is to protect the model from being accessed by adversaries, which can significantly lower the risk, since about 54.7% adversarially patched examples lose their adversarialness when attacking a black-box model, as we mentioned in Section VII-A1.

D. Spatial Boundedness of Physically Realizable Patches

To launch a physical attack, each segment has to be accurately placed on the target object, similar to how a local patch is applied. This involves some additional effort but is still physically possible. A major challenge is to preserve the spatial relation of the segments during placement. We address this in the following manner: we print the patch on adhesive paper and the patched image that includes both the patch and the target object on non-adhesive paper. We cut the patch from the adhesive paper, align it on the non-adhesive paper, and then align the non-adhesive paper on the target object. After peeling off the non-adhesive paper, the patch is attached to the target object. In this manner, attaching the patch to the target only took 1 to 2 minutes in our experiments.

We can also make the patch resilient to small deviations in segment positions during placement by introducing a minor spatial disturbance to each segment when generating a distributed patch.

We conclude that physical patches are not restricted by spatial boundaries.

VIII. Conclusion

In this paper, we present DorPatch, a novel adversarial patch attack that can evade both certifiable and empirical defenses against adversarial patch attacks, while being physically realizable for launching real-world attacks. DorPatch applies group lasso to the patch's mask, and employs image dropout, density regularization, and structural loss to generate a fully optimized, distributed, occlusion-robust, and inconspicuous adversarial patch that can fool DNN models in the physical world. We performed comprehensive experiments in both digital and physical domains to evaluate DorPatch and compare it with existing typical and state-of-the-art adversarial patch attacks. Our results show that DorPatch can effectively evade PatchCleanser, the state-of-the-art certifiable defense, and empirical defenses against adversarial patch attacks. Moreover, DorPatch can make PatchCleanser certify the wrong predictions of the adversarially perturbed examples, creating a false sense of security for the users. DorPatch achieves the best attack performance and perceptual quality among all adversarial patch attacks. DorPatch poses a serious challenge to the practical applications of DNN models and urges the development of more robust defenses against such attacks.

Acknowledgements

This work was partially supported by Hubei Province Key R&D Technology Special Innovation Project under Grant No. 2021BAA032, National Natural Science Foundation of China under Grants No. 62272175 and U1936211.

References

[1] S. Addepalli, D. Behl, G. Sriramanan, and R. V. Babu, "Efficient training methods for achieving adversarial robustness against sparse attacks," in *Proceedings of International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2021.

[2] A. Athalye, N. Carlini, and D. A. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 80. PMLR, 2018, pp. 274–283.

[3] T. Bai, J. Luo, and J. Zhao, "Inconspicuous adversarial patches for fooling image recognition systems on mobile devices," *IEEE Internet of Things Journal*, vol. 9, no. 12, pp. 9515–9524, 2022.

[4] A. J. Bose and P. Aarabi, "Adversarial attacks on face detectors using neural net based constrained optimization," in *Proceedings of International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2018, pp. 1–6.

[5] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[6] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," in *Proceedings of Conference on Neural Information Processing Systems Workshops (NeurIPS Workshops)*, 2017.

[7] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proceedings of Symposium on Security and Privacy (S&P)*. IEEE, 2017, pp. 39–57.

[8] S.-T. Chen, C. Cornelius, J. Martin, and D. H. P. Chau, "Shapeshifter: Robust physical adversarial attack on faster R-CNN object detector," in *Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, vol. 11051. Springer, 2018, pp. 52–68.

[9] P. Chiang, R. Ni, A. Abdelkader, C. Zhu, C. Studer, and T. Goldstein, "Certified defenses for adversarial patches," in *Proceedings of International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[10] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 97. PMLR, 2019, pp. 1310–1320.

[11] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 119. PMLR, 2020, pp. 2196–2205.

[12] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, "ImageNet: A large-scale hierarchical image database," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.

[13] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," *arXiv preprint arXiv:1708.04552*, 2017.

[14] B. G. Doan, E. Abbasnejad, and D. C. Ranasinghe, "Februus: Input purification defense against trojan attacks on deep neural network systems," in *Proceedings of Annual Computer Security Applications Conference (ACSAC)*. ACM, 2020, pp. 897–912.

[15] B. G. Doan, M. Xue, S. Ma, E. Abbasnejad, and D. C. Ranasinghe, "TnT attacks! universal naturalistic adversarial patches against deep neural network systems," *IEEE Transactions on Information Forensics and Security (TIFS)*, vol. 17, pp. 3816–3830, 2022.

[16] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 9185–9193.

[17] Y. Dong, T. Pang, H. Su, and J. Zhu, "Evading defenses to transferable adversarial examples by translation-invariant attacks," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 4312–4321.

[18] R. Duan, X. Ma, Y. Wang, J. Bailey, A. K. Qin, and Y. Yang, "Adversarial camouflage: Hiding physical-world attacks with natural styles," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1000–1008.

[19] 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 *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018, pp. 1625–1634.

[20] Y. Fan, B. Wu, T. Li, Y. Zhang, M. Li, Z. Li, and Y. Yang, "Sparse adversarial attack via perturbation factorization," in *Proceeding of European Conference of Computer Vision (ECCV)*, vol. 12367. Springer, 2020, pp. 35–50.

[21] J. Gildenblat, "Advanced AI explainability for pytorch," https://github.com/jacobgil/pytorch-grad-cam, 2022.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proceedings of International Conference on Learning Representations (ICLR Poster)*, 2015.

[23] S. Gowal, K. Dvijotham, R. Stanforth, R. Bunel, C. Qin, J. Uesato, R. Arandjelovic, T. Mann, and P. Kohli, "On the effectiveness of interval bound propagation for training verifiably robust models," *arXiv preprint arXiv:1810.12715*, 2018.

[24] H. Han, K. Xu, X. Hu, X. Chen, L. Liang, Z. Du, Q. Guo, Y. Wang, and Y. Chen, "Scalecert: Scalable certified defense against adversarial patches with sparse superficial layers," in *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, 2021, pp. 28 169–28 181.

[25] J. Hayes, "On visible adversarial perturbations & digital watermarking," in *Proceedings of Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops)*. IEEE, 2018, pp. 1597–1604.

[26] C. He, B. B. Zhu, X. Ma, H. Jin, and S. Hu, "Feature-indistinguishable attack to circumvent trapdoor-enabled defense," in *Proceedings of Conference on Computer and Communications Security (CCS)*. ACM, 2021, pp. 3159–3176.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016, pp. 770–778.

[28] Y.-C.-T. Hu, B.-H. Kung, D. S. Tan, J.-C. Chen, K.-L. Hua, and W.-H. Cheng, "Naturalistic physical adversarial patch for object detectors," in *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2021, pp. 7848–7857.

[29] Inspire-group, "hydra," https://github.com/inspire-group/hydra, 2021.

[30] D. Karmon, D. Zoran, and Y. Goldberg, "Lavan: Localized and visible adversarial noise," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 80. PMLR, 2018, pp. 2512–2520.

[31] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations (ICLR Poster)*. OpenReview.net, 2015.

[32] A. Krizhevsky, "Learning multiple layers of features from tiny images," *M.S. thesis, Department of Computer Science, University of Toronto*, 2009.

[33] M. Lecuyer, V. Atlidakis, R. Geambasu, D. Hsu, and S. Jana, "Certified robustness to adversarial examples with differential privacy," in *Proceedings of Symposium on Security and Privacy (S&P)*. IEEE, 2019, pp. 656–672.

[34] A. Levine and S. Feizi, "(De)Randomized smoothing for certifiable defense against patch attacks," in *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 6465–6475.

[35] B. Li, C. Chen, W. Wang, and L. Carin, "Certified adversarial robustness with additive noise," in *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, 2019, pp. 9459–9469.

[36] F. Li, X. Liu, X. Zhang, Q. Li, K. Sun, and K. Li, "Detecting localized adversarial examples: A generic approach using critical region analysis," in *Proceedings of Conference on Computer Communications (INFOCOM)*. IEEE, 2021, pp. 1–10.

[37] W. Lin, F. Sheikholeslami, J. Shi, L. Rice, and J. Z. Kolter, "Certified robustness against adversarial patch attacks via randomized cropping," in *Proceedings of International Conference on Machine Learning Workshops (ICML Workshops)*, 2021.

[38] A. Liu, X. Liu, J. Fan, Y. Ma, A. Zhang, H. Xie, and D. Tao, "Perceptual-sensitive GAN for generating adversarial patches," in *Proceedings of Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2019, pp. 1028–1035.

[39] X. Liu, H. Yang, Z. Liu, L. Song, H. Li, and Y. Chen, "DPATCH: an adversarial patch attack on object detectors," in *Proceedings of Workshop on Artificial Intelligence Safety co-located with Conference on Artificial Intelligence (SafeAI@AAAI)*, vol. 2301. CEUR-WS.org, 2019.

[40] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," in *Proceedings of International Conference on Learning Representations (ICLR Poster)*. OpenReview.net, 2017.

[41] B. Luo, Y. Liu, L. Wei, and Q. Xu, "Towards imperceptible and robust adversarial example attacks against neural networks," in *Proceedings of Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2018, pp. 1652–1659.

[42] M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. A. Wagner, "Minority reports defense: Defending against adversarial patches," in *Proceedings of Applied Cryptography and Network Security Workshops (ACNS Workshops)*, vol. 12418. Springer, 2020, pp. 564–582.

[43] J. H. Metzen and M. Yatsura, "Efficient certified defenses against patch attacks on image classifiers," in *Proceedings of International Conference on Learning Representations (ICLR)*. OpenReview.net, 2021.

[44] M. Mirman, T. Gehr, and M. Vechev, "Differentiable abstract interpretation for provably robust neural networks," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 80. PMLR, 2018, pp. 3578–3586.

[45] M. Naseer, S. Khan, and F. Porikli, "Local gradients smoothing: Defense against localized adversarial attacks," in *Proceedings of Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 1300–1307.

[46] S. Rao, D. Stutz, and B. Schiele, "Adversarial training against location-optimized adversarial patches," in *Proceeding of European Conference of Computer Vision Workshops (ECCV Workshops)*, vol. 12539. Springer, 2020, pp. 429–448.

[47] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, 2015, pp. 91–99.

[48] S. Roman, "Pytorch implementation of local gradients smoothing," https://github.com/metallurk/local_gradients_smoothing, 2020.

[49] H. Salman, S. Jain, E. Wong, and A. Madry, "Certified patch robustness via smoothed vision transformers," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022, pp. 15 116–15 126.

[50] V. Sehwag, C. Sitawarin, A. N. Bhagoji, A. Mosenia, M. Chiang, and P. Mittal, "Not all pixels are born equal: An analysis of evasion attacks under locality constraints," in *Proceedings of Conference on Computer and Communications Security (CCS)*. ACM, 2018, pp. 2285–2287.

[51] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Hydra: Pruning adversarially robust neural networks," in *Proceedings of Conference on Neural Information Processing Systems (NeurIPS)*, 2020, pp. 19 655–19 666.

[52] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 618–626.

[53] S. Shan, A. N. Bhagoji, H. Zheng, and B. Y. Zhao, "A real-time defense against website fingerprinting attacks," *arXiv preprint arXiv:2102.04291*, 2021.

[54] 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 Conference on Computer and Communications Security (CCS)*. ACM, 2016, pp. 1528–1540.

[55] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proceedings of International Conference on Learning Representations Workshops (ICLR Workshops)*, 2015.

[56] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *Proceedings of International Conference on Learning Representations (ICLR Poster)*, 2014.

[57] P. Team, "Torchvision models," https://pytorch.org/vision/stable/models.html, 2022.

[58] X. Wei, Y. Guo, and J. Yu, "Adversarial sticker: A stealthy attack method in the physical world," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 45, no. 3, pp. 2711–2725, 2023.

[59] B. Wu and B. Ghanem, "ℓp-box admm: A versatile framework for integer programming," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 41, no. 7, pp. 1695–1708, 2018.

[60] T. Wu, "phaattacks," https://github.com/tongwu2020/phattacks, 2020.

[61] T. Wu, L. Tong, and Y. Vorobeychik, "Defending against physically realizable attacks on image classification," in *Proceedings of International Conference on Learning Representations (ICLR)*. OpenReview.net, 2020.

[62] C. Xiang, "Code for PatchCleanser," https://github.com/inspire-group/PatchCleanser, 2022.

[63] C. Xiang, A. N. Bhagoji, V. Sehwag, and P. Mittal, "PatchGuard: A provably robust defense against adversarial patches via small receptive fields and masking," in *Proceedings of USENIX Security Symposium*. USENIX Association, 2021, pp. 2237–2254.

[64] C. Xiang, S. Mahloujifar, and P. Mittal, "PatchCleanser: Certifiably robust defense against adversarial patches for any image classifier," in *Proceedings of USENIX Security Symposium*. USENIX Association, 2022, pp. 2065–2082.

[65] C. Xiang and P. Mittal, "PatchGuard++: Efficient provable attack detection against adversarial patches," in *Proceedings of International Conference on Learning Representations Workshops (ICLR Workshops)*, 2021.

[66] C. Xie, Z. Zhang, Y. Zhou, S. Bai, J. Wang, Z. Ren, and A. L. Yuille, "Improving transferability of adversarial examples with input diversity," in *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019, pp. 2730–2739.

[67] K. Xu, S. Liu, P. Zhao, P. Chen, H. Zhang, Q. Fan, D. Erdogmus, Y. Wang, and X. Lin, "Structured adversarial attack: Towards general implementation and better interpretability," in *Proceedings of International Conference on Learning Representations (ICLR Poster)*. OpenReview.net, 2019.

[68] C. Yang, A. Kortylewski, C. Xie, Y. Cao, and A. Yuille, "PatchAttack: A black-box texture-based attack with reinforcement learning," in *Proceedings of European Conference of Computer Vision (ECCV)*, vol. 12371. Springer, 2020, pp. 681–698.

[69] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 68, no. 1, pp. 49–67, 2006.

[70] Z. Zhang, B. Yuan, M. McCoyd, and D. A. Wagner, "Clipped BagNet: Defending against sticker attacks with clipped bag-of-features," in *Proceedings of Symposium on Security and Privacy Workshops (S&P Workshops)*. IEEE, 2020, pp. 55–61.

[71] Y. Zhao, H. Zhu, R. Liang, Q. Shen, S. Zhang, and K. Chen, "Seeing isn't believing: Towards more robust adversarial attack against real world object detectors," in *Proceedings of Conference on Computer and Communications Security (CCS)*. ACM, 2019, pp. 1989–2004.