Saliuitl: Ensemble Salience Guided Recovery of Adversarial Patches against CNNs

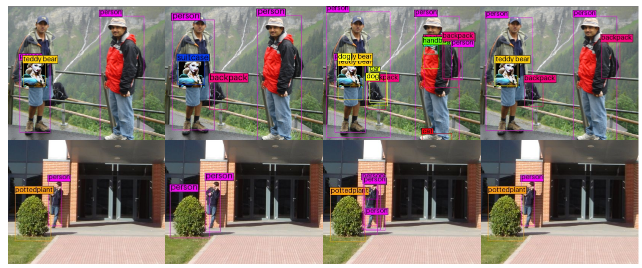
Mauricio Byrd Victorica, György Dán, Henrik Sandberg
KTH Royal Institute of Technology, Sweden
{mbv, gyuri, hsan}@kth.se

Abstract

Adversarial patches are capable of misleading computer vision systems based on convolutional neural networks. Existing recovery methods suffer of at least one of three fundamental shortcomings: no information about the presence of patches in the scene, inability to efficiently handle noncontiguous patch attacks, and a strong reliance on fixed saliency thresholds. We propose Saliuitl, a recovery method independent of the number of patches and their shape, which unlike prior works, explicitly detects patch attacks before attempting recovery. In our approach, detection is based on the attributes of a binarized feature map ensemble, which is generated by using an ensemble of saliency thresholds. If an attack is detected, Saliuitl recovers clean predictions locating patches guided by an ensemble of binarized feature maps and inpainting them. We evaluate Saliuitl on widely used object detection and image classification benchmarks from the adversarial patch literature, and our results show that compared to recent state-of-the-art defenses, Saliuitl achieves a recovery rate up to 97.81 and 42.63 percentage points higher at the same rate of lost predictions for image classification and object detection, respectively. By design, Saliuitl has low computational complexity and is robust to adaptive white-box attacks. Our code is available at `https://github.com/Saliuitl/Saliuitl/tree/main`.

1. Introduction

Convolutional neural networks (CNNs) are arguably the most widely used deep neural networks for computer vision tasks such as object detection and image classification [27]. Despite their remarkable performance, they are vulnerable to various adversarial attacks [16]. Physically realizable attacks are especially alarming, since they can be added to a physical scene rather than digitally and directly to a model's input, and have been remarkably effective in misleading CNN models [1, 2, 11, 14, 24, 36].



(a) Undefended (b) PAD (c) Obj. Seeker (d) *Saliuitl*

Figure 1. Undefended and recovered object detection outputs. Existing empirical (b) and certifiable (c) defenses struggle to balance performance on attacked (top) and clean (bottom) images, unlike *Saliuitl* (d).

Adversarial patches, commonly used to deceive computer vision models, stand out among physically realizable attacks due to their effectiveness and ease of implementation. As a consequence, various methods have been proposed in recent years to defend CNN models against patch attacks, aiming either at detecting the presence of an adversarial patch in the model's input [3, 7, 17, 28, 32, 38, 39], at recovering from the attack by removing (and potentially replacing) input regions deemed to contain an adversarial patch to suppress their effect on the model's output [18, 22, 25, 26, 33, 35, 40–42], or at modifying the model itself or its training process to improve robustness to adversarial attacks [4, 5, 43, 44].

Among these three approaches, recovery methods have the greatest promise, as they could enable the design of robust autonomous systems based on visual perception [15, 34]. Existing approaches suffer, however, from three major shortcomings. First, empirical defenses rely on fixed saliency thresholds to either detect or localize adversarial regions [18, 22, 25, 26, 33, 35, 42]. By determining an adequate threshold on a certain input characteristic (e.g., entropy [35], semantic consistency [22, 42], hidden layer activations [18]), these methods detect or localize adversarial regions as those that are above the threshold, referred to as salient. The only methods that do not rely on fixed saliency thresholds are so-called certifiable defenses, which do, however, perform an almost exhaustive search for potentially patched regions, and hence involve significant computational overhead [17, 40, 41]. Second, existing methods lack an explicit attack detection mechanism. Hence, they execute the entire recovery pipeline on all inputs without considering the likely case that the inputs are not subject to any attack [18, 22, 25, 26, 33, 35, 40, 41]. This is the case even for methods that localize adversarial regions *before* performing recovery, which can be considered an implicit attack detection mechanism [18, 22, 25, 26, 33, 35]. The consequence is a significant computational overhead in non-adversarial environments, and it may even affect performance on clean inputs, as shown in Figure 1. Third, existing detection and recovery defenses assume single-patch attacks, and make assumptions about the shape and size of a patch. Yet, in reality, an attacker is not constrained to use a single adversarial patch per image to achieve its objective; hence not adapting [17, 18, 26, 40, 41] or not evaluating explicitly [22, 25, 35] defense mechanisms in a scenario with noncontiguous adversarial regions of different shapes and sizes ignores many possible attack vectors.

We propose *Saliuitl*, an adversarial patch recovery method designed to overcome the above limitations. By performing an explicit detection stage, *Saliuitl* can maintain performance on clean inputs and is computationally efficient, which makes it applicable in time-sensitive scenarios. Moreover, instead of comparing relevant input characteristics to a fixed threshold, *Saliuitl* detects compromised inputs through the evolution of input attributes across an ensemble of thresholds. Crucially, *Saliuitl* is agnostic to the number of patches in the input, and so is its computational cost. We evaluate *Saliuitl* against state-of-the-art baselines on the tasks of object detection and image classification. Our results show that it achieves a substantial improvement in terms of recovery rate for an arbitrary number of patches and is robust to an adaptive white-box attacker with knowledge of *Saliuitl*.

The rest of the paper is presented as follows. In Section 2 we introduce our problem formulation and related works. In Section 3 we introduce *Saliuitl*, and in Section 4 we provide numerical results. Section 5 concludes the paper.

2. Preliminaries

2.1. Adversarial Patch Recovery

Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ be a dataset of input-output pairs, and let $h : \mathcal{X} \to \mathcal{Y}$ be a model. An adversarial attack is a perturbation $p \in \mathcal{P}$ that can be applied to an input $\mathbf{x}_i \in \mathcal{X}$ through a transformation $\mathcal{A}$, yielding an attacked input $\mathcal{A}(\mathbf{x}_i, p)$. Given a set of target outputs $t(\mathbf{x}_i) \not\ni h(\mathbf{x}_i)$, the attacker's objective is to find a perturbation $\hat{p}$ that, when applied to $\mathbf{x}_i$ and fed into the victim model h, results in an output $h(\mathcal{A}(\mathbf{x}_i, p)) \in t(\mathbf{x}_i)$. That is, the attack aims to find

$$\hat{p} \in \arg\max_{p \in \mathcal{P}} \ \mathbb{E}_{\mathcal{D}}[\log \Pr(h(\mathcal{A}(\mathbf{x}_i, p)) \in t(\mathbf{x}_i))].$$

Note that $t(\mathbf{x}_i)$ is a set, since attacks may be targeted or untargeted. For untargeted attacks, $t(\mathbf{x}_i) = \mathcal{Y} \setminus \{h(\mathbf{x}_i)\}$ (i.e., any output different from $h(\mathbf{x}_i)$), while for targeted attacks $t(\mathbf{x}_i) \subset \mathcal{Y} \setminus \{h(\mathbf{x}_i)\}$ (only certain outputs are attacker targets). In particular, we are interested in adversarial patch attacks [2], where the set $\mathcal{P}$ of perturbations defines the feasible locations, sizes, and shapes of the patches, and the transformation $\mathcal{A}$ replaces the corresponding pixels in $\mathbf{x}_i$ with the patch attack $p \in \mathcal{P}$. Furthermore, we focus on the common case that h is a CNN model.

Adversarial patch recovery aims at ensuring that the model output is not affected by a physically realizable adversarial patch attack. For a formal definition, consider a dataset of inputs $\mathcal{X}'$ isomorphic to $\mathcal{X}$. Each element $\mathbf{x}'_i \in \mathcal{X}'$ corresponds to exactly one element $\mathbf{x}_i \in \mathcal{X}$, and may or may not have been subject to an adversarial attack p_i. That is, $\forall \mathbf{x}'_i \in \mathcal{X}' \ \exists \mathbf{x}_i \in \mathcal{X}$ such that either $\mathbf{x}'_i = \mathbf{x}_i$ *or* $\mathbf{x}'_i = \mathcal{A}(\mathbf{x}_i, p_i)$ for some $p_i \in \mathcal{P}$ otherwise, and vice versa. A recovery method $\mathcal{R}_\phi : \mathcal{X}' \to \mathcal{Y}$ aims at obtaining the clean prediction $h(\mathbf{x}_i)$ of the victim model, without knowing whether the input $\mathbf{x}'_i \in \mathcal{X}'$ has been subject to an adversarial patch attack or not. For a recovery method $\mathcal{R}_\phi$ parameterized by $\phi \in \mathbf{\Phi}$, the objective is to find the optimal parameters

$$\hat{\phi} \in \arg\max_{\phi \in \mathbf{\Phi}} \ \mathbb{E}_{\mathcal{X}'}[\log \Pr(\mathcal{R}_\phi(\mathbf{x}'_i) = h(\mathbf{x}_i))].$$

Note that the goal is to recover $h(\mathbf{x}_i)$. Some methods try to obtain $\mathbf{x}_i$ in the process [22, 25, 35] and others do not [18, 41, 43]. The above definition of recovery captures existing defenses, as ϕ can include, e.g., parameters of the model h, saliency thresholds for methods that rely on modifying suspicious input regions, or the parameters of a robustified model for methods based on adversarial training.

2.2. Related Work

Existing defenses for adversarial patch recovery can be divided into two categories: *empirical* and *certifiable* defenses. Empirical defenses leverage salient characteristics of adversarial patches to remove potentially adversarial regions in an input to the victim model. *Themis* [18] relies on the output of a shallow victim model layer (i.e., a feature map) to localize adversarial patches. Using an importance threshold β, feature maps are binarized, and patch candidates are identified by using a sliding window in the feature map, where any window with a fraction of important neurons above a threshold θ is considered a candidate. It assumes single-patch attacks and a known upper bound on the patch size, required to define an adequate sliding window. Recovery is performed by occluding the image regions associated with patch candidates, and monitoring the effect on the model's output. *Jedi* [35] considers input re-

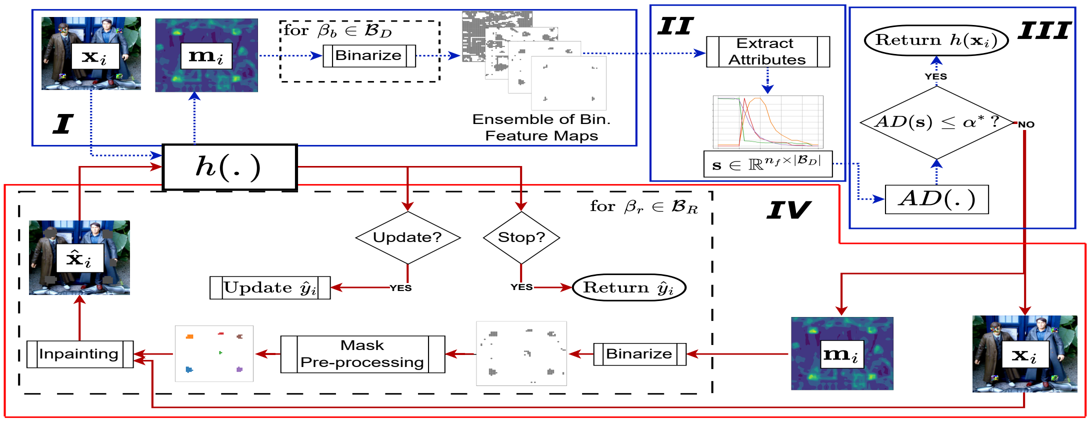


Figure 2. *Saliuitl*: During the detection stage, (I) an ensemble of binary feature maps is computed according to the threshold set $\mathcal{B}_D$, (II) attribute extraction is applied over the ensemble and the results are aggregated into the attribute vector $\mathbf{s}$, and (III) an attack detector *AD* is fed with $\mathbf{s}$. If *AD* detects adversarial patches, (IV) *Saliuitl* will perform recovery on the input $\mathbf{x}_i$ guided by the binary feature maps computed according to the threshold set $\mathcal{B}_R$.

gions with an abnormally high entropy to be potential adversarial patches, hence it generates a heat map based on image entropy, and using a threshold based on statistics for clean images, patches, and the current input, it computes a mask to occlude suspicious regions from the heatmap, which it refines using an autoencoder; masked input regions are inpainted. *PAD* [22] also computes a heatmap to determine suspicious input regions, but instead of using entropy, semantic independence and spatial heterogeneity are used to compute heatmaps, which are then fused into a single heatmap. A threshold based on the input's statistics is used to determine adversarial regions, and using an image segmentation model, a mask is generated for the input, which is used to blacken all localized suspicious regions. *NutNet* [25] proposes to use destructive training to train an autoencoder that has low reconstruction error on the distribution of clean images, and a high error on inputs outside said distribution. Using a threshold on the reconstruction error of input regions, and a second threshold on the absolute changes in pixel values on the reconstructed image, a *Dual-Mask* is computed and used to replace masked regions with gray pixels. All these *empirical* methods use fixed saliency thresholds, i.e., despite adjusting their thresholds according to the current input, these defenses define adversarial regions as salient regions in terms of some input feature, with respect to a threshold that is fixed for said input. This issue has been raised only in the context of attack detection [3].

Certifiable defenses aim to be agnostic to the content of an adversarial patch, but they rely on strong assumptions regarding the shape, size, or number of patches. *Object Seeker* [41] is a certifiable defense for evasion attacks on object detection that performs recovery by systematically splitting the input into two regions, using a set of horizontal and vertical lines, and aggregates the detection results into a final inference. *Patch Cleanser* [40] is a similar approach for image classification, where a set of masks is applied on the input to identify covered input versions whose prediction is unique and distinct, and a second set of masks is applied to such covered inputs to extract a clean prediction. Note that while these methods do not rely on the content of adversarial patches, they both assume single-patch attacks, and *Patch Cleanser* further assumes that an adequate upper bound on the patched region is known.

Compared to existing recovery approaches, *Saliuitl* is different in three fundamental ways. First, it does not use a saliency threshold to identify patch candidates, and instead detects candidate regions based on how attributes of saliency maps change depending on the saliency threshold. Second, *Saliuitl* does not assume that the shape, size or number of patches is known or is bounded. Finally, it differs from existing approaches also in that it contains an explicit detection stage, which allows to perform recovery only on inputs that are suspected to be attacked. This design choice provides a significant advantage in realistic environments that are mostly non-adversarial.

3. *Saliuitl*: Detection and Recovery in the Saliency Domain

We propose *Saliuitl* to solve the adversarial recovery problem defined in Section 2.1. *Saliuitl* consists of two consecutive stages, and is illustrated in Figure 2. In the *detection* stage, it determines whether $\mathbf{x}_i$ contains one or more adversarial patches. If it detects a patch, it moves into the *recovery* stage, where it iteratively counters the effect of the adversarial patch by converting $\mathbf{x}_i$ into a modified input $\hat{\mathbf{x}}_i$ and updating the recovered output $\hat{\mathbf{y}}_i$. The pseudo-code for *Saliuitl* is available in Algorithm 1 in the Appendix. In what follows we describe the two stages.

Detection. We feed the input $\mathbf{x}_i$ into model h and extract a feature map $\mathbf{m}_i \in \mathbb{R}^{m_x \times m_y}$ by doing channel-wise addition on the output of a shallow layer of h. Instead of localizing adversarial regions in $\mathbf{m}_i$ or $\mathbf{x}_i$ as in existing defenses, one key tenet in *Saliuitl* is to use a set of thresholds $\mathcal{B}_D$ within the range of values in $\mathbf{m}_i$. We use $\mathcal{B}_D$ to distinguish between clean and compromised images based on how salient regions in $\mathbf{m}_i$ change as the saliency threshold changes. Thus, for a set $\mathcal{B}_D$ of saliency thresholds ordered in strictly ascending order, for each threshold $\beta_b \in \mathcal{B}_D$ we compute a binary feature map $B_b \in \{0,1\}^{m_x \times m_y}$ from $\mathbf{m}_i$ by replacing entries in $\mathbf{m}_i$ below the current β_b by zeros, and all others by ones. This process yields an ensemble of binary feature maps. Note that $\mathcal{B}_D$ depends on the range of values in $\mathbf{m}_i$, and thus also on $\mathbf{x}_i$.

Next, we extract a set of attributes from each binary feature map, and aggregate the attributes across the ensemble into an ensemble attribute vector. Hence, the set $\mathcal{B}_D$ of thresholds defines an ensemble of binary feature maps, and the n_f attributes of each member of the ensemble are aggregated into an ensemble attribute vector $\mathbf{s} \in \mathbb{R}^{n_f \times |\mathcal{B}_D|}$. Observe that for each of the n_f ensemble attributes, the way the attribute changes as a function of β_b is described by a one-dimensional curve, and $\mathbf{s}$ is the concatenation of n_f curves, as illustrated in Figure 2; in Section 4.1 and in the Appendix, we provide details of how to construct $\mathbf{s}$ for a concrete choice of attributes. Next, we use an attack detector *AD* that outputs a detection score $AD(\mathbf{s}) \in [0,1]$ for the attribute vector $\mathbf{s}$. We choose to implement *AD* using a 1D CNN architecture since the attribute vector $\mathbf{s}$ can be interpreted as a n_f-channel one-dimensional signal. This simple architecture allows for a reasonably low inference time. In short, *AD* consists of two 1D convolutional layers, followed by two fully connected layers, and an output layer with a sigmoid activation function. We provide a detailed description of AD and of its training in the Appendix, as well as a discussion on alternatives to our proposed implementation.

In the design of *Saliuitl* we have intentionally sidestepped the need for determining an adequate saliency threshold to discriminate adversarial and clean regions in $\mathbf{x}_i$. The only threshold we make use of is the detection threshold $\alpha^* \in [0,1]$: $AD(\mathbf{x}_i) \geq \alpha^*$ implies the presence of patches in $\mathbf{x}_i$. α^* allows to find a good trade-off between avoiding recovery on clean images and not detecting attacked images. This first, explicit attack detection stage distinguishes *Saliuitl* from other state-of-the-art recovery methods [18, 22, 26, 35, 40, 41, 43], which perform recovery on any input to h. Performing recovery only on inputs where an attack is detected avoids the computational burden of unnecessarily recovering clean images. Moreover, unlike other recovery methods, *Saliuitl* enhances situational awareness by detecting inputs with suspicious patterns regardless of the effect they may have on the model's output.

Recovery. For an input $\mathbf{x}_i$ in which an adversarial patch has been detected, let $\hat{\mathbf{y}}_i = h(\mathbf{x}_i)$ be the initial model output. For recovery, we define $\mathcal{B}_R$, the set of recovery saliency thresholds within the range of values in $\mathbf{m}_i$. $\mathcal{B}_R$ may overlap with the set $\mathcal{B}_D$ used for detection. Importantly, $\mathcal{B}_R$ is in strictly descending order. For each threshold $\beta_r \in \mathcal{B}_R$, we compute a binary feature map $B_r \in \{0,1\}^{m_x \times m_y}$ from $\mathbf{m}_i$ and use it to localize input regions that correspond to adversarial patches. Since recovery takes place after an attack is detected, we may simply consider that all the ones in B_r (i.e., important neurons in $\mathbf{m}_i$) correspond to an adversarial patch, but we may also preprocess B_r to remove outliers. Thus, to recover from the detected patch attack we derive $\hat{\mathbf{x}}_i$ from $\mathbf{x}_i$, by inpainting the pixels corresponding to potentially adversarial feature map regions. We then monitor how the output $h(\hat{\mathbf{x}}_i)$ differs from $\hat{\mathbf{y}}_i$, and check if the inpainting has contributed to countering the effect of an attack, that is, if new objects are detected in the case of object detection and if the predicted label changed in the case of image classification. If this is the case then $\hat{\mathbf{y}}_i$ is updated accordingly (see the Appendix for details). Note that the strictly descending order of $\mathcal{B}_R$ ensures that the inpainted area increases monotonically at every iteration, since more neurons will be considered salient as the value of β_r decreases. After computing $h(\hat{\mathbf{x}}_i)$ and possibly updating $\hat{\mathbf{y}}_i$, we stop if any of the stopping conditions is satisfied. For object detection, the stopping condition is (i) if the total amount of important neurons to be used for inpainting in the current β_r suffice to occlude at least half of all the pixels in $\mathbf{x}_i$. For image classification, besides (i), an additional condition is (ii) if the predicted label is different from that without inpainting, i.e., if $\hat{\mathbf{y}}_i$ has changed. If any of the stopping conditions is met or if *Saliuitl* has iterated through all thresholds in $\mathcal{B}_R$, the recovered output $\hat{\mathbf{y}}_i$ is returned. Note that update and stopping conditions depend on the victim model's task.

Three key design choices set *Saliuitl* apart from prior works. First, *Saliuitl* has an explicit attack detection stage, and thus only performs recovery on images potentially containing adversarial patches. In practice, most images are not expected to be adversarially manipulated, allowing notable savings in computational burden and gains on clean performance. Second, *Saliuitl* does not need an explicit threshold of input features to discriminate adversarial images from clean ones, as it detects attacks based on how the extracted attributes change in an image as one varies the definition of *important* neurons, i.e., the saliency threshold. This design choice makes *Saliuitl* robust to adaptive attacks, as shown in Section 4.3. Finally, instead of assuming a specific shape for the detected patches (mostly rectangular single patches in the literature), it leverages the evolution of binary feature maps across decreasing saliency thresholds to directly remove relevant candidate regions of arbitrary shapes.

Figure 3. Top (from left to right): single, double, triangular, and multi-object patch attacks on object detection. Bottom (from left to right): single, double, quadruple, and triangular patch attacks on image classification.

4. Numerical Results

4.1. Experimental Setup

Benchmarks and baselines. We use four commonly used datasets in the literature on adversarial patches. For the object detection task we use INRIA [8] and Pascal VOC [13], and for the image classification task we use ImageNet [10] and CIFAR-10 [23]. We compare *Saliuitl* against a variety of baseline defenses, covering robust models [43], empirical recovery [18, 22, 25, 35], and certifiable recovery [40, 41]. The appendix contains evaluations on additional baselines.

Attacks and Victim Models. To facilitate the generation of challenging patch attack scenarios, we employ white-box patch attacks on representative CNN models. For evaluation, we consider the *Princess* evasion attack by [24] on YOLOv2 [31] for object detection, and the untargeted misclassification attack from the official implemenation of [39] on ResNet-50 [19] for image classification. We also consider the evasion attack by [36] on object detection during training and to generate adaptive attacks. We create single, multiple (i.e., noncontiguous), and triangular patch versions of these attacks as illustrated in Figure 3; we provide detailed descriptions for each attack scenario in the Appendix.

Evaluation metrics. A central notion in our evaluation is that of *effective* attacks. For object detection, we say that an attack p_i applied to input $\mathbf{x}_i$ is effective if there is at least one object in the clean predicted output $h(\mathbf{x}_i)$ that has an intersection over union (IoU) below 0.5 with each object in the attacked predicted output $h(\mathcal{A}(\mathbf{x}_i, p_i))$. For image classification, we say that an attack p_i applied to $\mathbf{x}_i$ is effective if the adversarial class prediction $h(\mathcal{A}(\mathbf{x}_i, p_i))$ is different from the clean prediction $h(\mathbf{x}_i)$. Since the attack on image classification relies on the ground truth labels for optimization, we consider only correctly classified examples for image classification, i.e., inputs $\mathbf{x}_i$ for which $h(\mathbf{x}_i)$ equals the ground truth label.

Each attack scenario is based on a dataset of clean images $\mathcal{X}_0$. We apply the corresponding adversarial patch attack on each element of $\mathcal{X}_0$, yielding the attacked dataset $\mathcal{X}'_0$, which is partitioned into effectively attacked images $\mathcal{X}'_1$, and ineffectively attacked images $\mathcal{X}'_2$, i.e., $\mathcal{X}'_0 = \mathcal{X}'_1 \cup \mathcal{X}'_2$. Note that each clean image corresponds to exactly one attacked image and vice versa, so that $|\mathcal{X}_0| = |\mathcal{X}'_1 \cup \mathcal{X}'_2|$. Moreover, based on $\mathcal{X}'_1$ and $\mathcal{X}'_2$, we define corresponding clean partitions $\mathcal{X}_1$ and $\mathcal{X}_2$, so that $\mathcal{X}_0 = \mathcal{X}_1 \cup \mathcal{X}_2$. Each element in $\mathcal{X}'_j$ is generated by applying the scenario's patch attack to exactly one element in $\mathcal{X}_j$.

For a defense method $\mathcal{R}_\phi$, we say the *recovery rate* is the fraction of effectively attacked inputs (i.e., images in $\mathcal{X}'_1$) recovered by $\mathcal{R}_\phi$. In addition, we say the *lost prediction rate* is the fraction of clean inputs (i.e., images in $\mathcal{X}_1$) for which applying $\mathcal{R}_\phi$ results in an incorrect output. Finally, we say the *inflicted attack rate* is the fraction of ineffectively attacked inputs (i.e., images in $\mathcal{X}'_2$) for which applying $\mathcal{R}_\phi$ results in an incorrect output. Object detection models are commonly evaluated using the mean average precision (mAP), which is the area under the precision-recall curve of detected objects (see the Appendix for further details). We define adversarial normalized mean average precision (adversarial nmAP) as the mAP for effectively attacked inputs (i.e., images in $\mathcal{X}'_1$) after applying $\mathcal{R}_\phi$, divided by the mAP for clean inputs (i.e., images in $\mathcal{X}_1$) in the absence of any recovery method. Similarly, we introduce clean nmAP as the mAP for clean inputs (i.e., images in $\mathcal{X}_1$) after applying $\mathcal{R}_\phi$, divided by the mAP for clean inputs (i.e., images in $\mathcal{X}_1$) in the absence of any recovery method. Note that for each input, we consider the ground truth to be the undefended clean predicted output $h(\mathbf{x}_i)$. Finally, for an attack detector $\mathcal{F} : \mathcal{X} \to \{0, 1\}$, which outputs binary labels indicating whether or not an input contains an adversarial patch, we introduce *attack detection accuracy* as the fraction of correctly labeled attacked inputs (i.e., images in $\mathcal{X}'_0$). Note that the ground truth label is 0 for all inputs in $\mathcal{X}_0$, and 1 for all inputs in $\mathcal{X}'_0$, i.e., all attacked inputs should be labeled 1, regardless of attack effectiveness.

***Saliuitl* Implementation Details.** We now describe the concrete choices for attribute extraction, mask preprocessing, and inpainting that we use in our evaluation. We perform extensive ablation experiments on alternative choices for these components in the Appendix.

In the detection stage, for each input $\mathbf{x}_i$, we get $\mathbf{m}_i$ from the first max pooling layer and use $\mathcal{B}_D = \{\frac{x \cdot \max(\mathbf{m}_i)}{20}\}_{x=0}^{19}$. We extract $n_f = 4$ attributes from each binary feature map B_b as in [3]. The first attribute is the amount of active neurons (i.e., the number of ones) in B_b. The other three attributes characterize the spatial distribution of important neurons in B_b and are obtained through density-based clustering using DBSCAN [12] with parameters $\epsilon = 1$ and $n_{min} = 4$. The second attribute is the number of identified clusters. The last two attributes are the mean and standard deviation of the average intra-cluster distance, i.e., the average distance between points in each cluster c identified in

Table 1. Best Recovery Rate (*RR*) - Lost Prediction Rate (*LPR*) trade-offs for single (1), double (2), quadruple (4), triangular (T), and multi-object (MO) patch attack scenarios. Best and second best methods per row are in bold and underlined, respectively, according to ***RR*** ***minus LPR***. *Certifiable* stands for *Object Seeker* in INRIA/VOC and *Patch Cleanser* in ImageNet/CIFAR-10.

Attack	***Saliuitl*** RR/LPR	**NutNet** RR/LPR	**PAD** RR/LPR	**Jedi** RR/LPR	**Themis (50)** RR/LPR	**Certifiable** RR/LPR	**FNS**
INRIA-1	0.5909/0.0152	0.3182/0.1515	0.3182 /0.3030	0.3030/0.4545	0.2273/0.0 (0.2879/0.0)	**0.6515/0.0**	0.1515/0.0606
INRIA-2	0.3871/0.0	0.0968/0.0484	0.1774/0.2742	0.1935/0.4516	0.0645/0.0 (0.0806/0.0161)	**0.5968/0.0**	0.1129/0.0484
INRIA-T	0.4737/0.0526	0.0351/0.0	0.3158/0.2982	0.1930/0.2982	0.0877/0.0 (0.1754/0.0)	**0.6491/0.0**	0.1053/0.0351
INRIA-MO	**0.4749/0.0**	0.1285/0.0726	0.2235/0.2514	0.1732/0.3631	0.0950/0.0056 (0.0838/0.0)	0.4190/0.0	0.0670/0.0223
VOC-1	0.5404/0.0293	0.1637/0.1544	0.2938/0.2573	0.2423/0.4546	0.1594/0.0143 (0.2159/0.0122)	**0.6483/0.0057**	0.1944/0.1315
VOC-2	0.5376/0.0125	0.1027/ 0.1074	0.2429/0.2798	0.2006/0.5000	0.0533/0.0086 (0.0886/0.0196)	**0.6034/0.0196**	0.1332/0.0799
VOC-T	0.4244/0.0348	0.0174/0.0816	0.1503/0.2906	0.1613/0.5041	0.0596/0.0101 (0.0852/0.0055)	**0.6425/0.0055**	0.0495/0.0449
VOC-MO	0.3955/0.0095	0.0067/0.0597	0.1456/0.1555	0.1432/0.2853	0.0392/0.0060 (0.0576/0.0148)	**0.4284/0.0046**	0.1018/0.0601
ImageNet-1	0.8869/0.0071	0.9090/0.0681	0.8838/0.2073	0.81/0.099	**0.9286/0.0039** (0.8663/0.0020)	0.5950/0.5720	0.8416/0.0549
ImageNet-2	**0.8535/0.0061**	0.9090/0.0920	0.8620/0.1996	0.6747/0.1078	0.1786/0.0051 (0.1743/0.0065)	0.0988/0.0948	0.8124/ 0.0651
ImageNet-4	**0.8436/0.0086**	0.9043/0.0898	0.8220/0.1961	0.6027/0.0923	0.0950/0.0067 (0.0856/0.0021)	0.0627/0.0642	0.7887/0.0820
ImageNet-T	**0.5065/0.0612**	0.5023/0.2631	0.5165/0.4607	0.4198/0.3334	0.4423/0.0497 (0.4098/0.0346)	0.1964/0.1973	0.3819/0.1491
CIFAR-1	**0.9738/0.0008**	0.9641/0.0025	0.9615/0.0178	0.8113/0.0133	0.8652/0.0046 (0.7975/0.0059)	0.4630/0.4260	0.8316/0.0440
CIFAR-2	**0.9789/0.0006**	0.9617/0.0032	0.9141/0.0140	0.6315/0.0245	0.4553/0.0083 (0.4310/0.0056)	0.1917/0.1700	0.8411/0.0292
CIFAR-4	**0.9747/0.0**	0.9501/0.0019	0.8239/0.0271	0.2540/0.0224	0.2516/0.0047 (0.2568/0.0009)	0.1131/0.1120	0.8258/0.0300
CIFAR-T	**0.8566/0.0**	0.6339/0.0207	0.5423/0.0604	0.4629/0.0777	0.4577/0.0104 (0.4542/0.0052)	0.3212/0.3230	0.6373/0.1347

B_b. In the recovery stage we set $\mathcal{B}_R = \mathcal{B}_D$, apply DBSCAN to each binary feature map B_r, and leverage the DBSCAN results for mask preprocessing, using only neurons within the detected clusters in B_r to determine potentially adversarial regions; we use biharmonic inpainting [6, 9] to compute $\hat{\mathbf{x}}_i$. A detailed description is available in the Appendix. We do not use a fixed attack detection threshold α^*, but instead we obtain results for $\alpha^* \in \{\frac{x}{100}\}_{x=0}^{99}$ and report the best results for each attack scenario. In the Appendix, we provide further details on the training and implementation of our detector network *AD*; we evaluate *Saliuitl* with attacks that are not used for training.

Baseline Parameters. To allow for a fair comparison, we vary the parameters of the baseline methods in order to compare *Saliuitl* to their best performing settings. Sensible ranges for the parameters were determined based on the respective papers and their official implementations (available for all baselines except *Themis*). Note that *Themis* assumes a known bound on the patch size; to assess the impact of this unrealistic assumption we also report results for *Themis-50*, where the bound used by *Themis* is half the true bound. Further details on our implementations for the baselines can be found in the Appendix. Note that Object Seeker is not applicable to image classification and Patch Cleanser is not applicable to object detection.

4.2. Results

Object Detection. We first evaluate the recovery performance of *Saliuitl* for object detection. The first eight rows in Table 1 show the best recovery performance achieved by *Saliuitl* and the baselines, expressed as the best tradeoff between recovery and lost prediction rates achieved by each approach. We observe that *Saliuitl* outperforms all non-certifiable defenses, and it even outperforms *Object Seeker* (the certifiable defense) for multi-object patches in the INRIA dataset. We illustrate the detailed tradeoff between recovery rate and lost predictions in Figures 5 and 6 in the Appendix. Our results indicate that for single patches *Saliuitl* achieves a favorable tradeoff between recovery rate and lost predictions; at a very low lost prediction rate, it achieves a recovery rate that can only be attained by *Object Seeker* among all baselines. We can also conclude that most methods perform significantly worse when there are multiple patches on the image. Interestingly, even though *Object Seeker* was designed for single patches it performs well for double patches as well, although its performance decreases significantly for the multi-object patch. We conjecture that the double patch locations in our chosen attack model allow *Object Seeker* to retain its functionality in this scenario. Moreover, with the exception of *Object Seeker*, all methods experience a notable drop in performance for the triangular patch. *Saliuitl*'s advantage over other non-certifiable defenses in terms of recovery and lost prediction rates becomes emphasized in the triangular patch scenario. Recall that *Saliuitl* makes no assumptions regarding the shape of the patch. We conjecture that the performance drop experienced by *Saliuitl* is due to the square kernels used by CNN models to compute feature maps at intermediate layers, including the shallow feature maps leveraged by *Saliuitl* to perform detection and recovery.

We further assess the object detection performance attained by each method using the nmAP in Table 2, which shows the best tradeoff between clean and adversarial nmAP for each method, and in Figures 7 and 8 in the Appendix, which show the detailed tradeoff for all methods. *Saliuitl* outperforms all baseline defenses, regardless of the number or shape of the patches. Among all baselines, only *FNS* in the single and triangular patch scenarios is capable of reaching *Saliuitl*'s adversarial nmAP, but at a notably lower clean nmAP. Similarly, while *Object Seeker* and

Table 2. Best adversarial-clean nmAP trade-offs for object detection. For single (1), double (2), triangular (T), and multi-object (MO) patch attack scenarios. Best and second best methods per row are in bold and underlined, respectively, according to adv. *plus* clean nmAP.

Attack	*Saliuitl* Adv./Clean	NutNet Adv./Clean	PAD Adv./Clean	Jedi Adv./Clean	Themis (50) Adv./Clean	Object Seeker Adv./Clean	FNS Adv./Clean
INRIA-1	**0.6897/0.9998**	0.6267/0.8836	0.5168/0.6251	0.5726/0.6389	0.6258/1.0 (0.6405/1.0)	0.6040/1.0	0.7337/0.9249
INRIA-2	**0.5998/1.0**	0.5228/0.9526	0.5354/0.7457	0.3962/0.6880	0.5469/1.0 (0.5509/1.0)	0.5035/1.0	0.5462/0.9297
INRIA-T	**0.7034/0.9987**	0.5765/0.9519	0.6007/0.7333	0.5559/0.6884	0.5965/1.0 (0.6238/1.0)	0.5700/1.0	0.5803/0.9307
INRIA-MO	**0.4737/0.9999**	0.2962/0.8931	0.2330/0.6771	0.3384/0.6462	0.3017/0.9939 (0.3077/0.9970)	0.2485/0.9641	0.2993/0.9101
VOC-1	**0.5094/0.9950**	0.4593/0.9184	0.4623/0.7312	0.3870/0.5930	0.4735/1.0 (0.4732/1.0)	0.4626/0.9598	0.4888/0.9585
VOC-2	**0.5088/0.9940**	0.4262/0.9262	0.4485/0.7295	0.3972/0.5670	0.4329/1.0 (0.4356/1.0)	0.4221/0.9677	0.4332/0.9448
VOC-T	**0.5043/0.9942**	0.4654/0.9177	0.4604/0.7259	0.4243/0.5842	0.4969/1.0 (0.5040/0.9935)	0.4842/0.9756	0.4995/0.9609
VOC-MO	**0.3563/0.9877**	0.2279/0.9269	0.2567/0.7385	0.2303/0.6048	0.2318/0.9744 (0.2318/0.9744)	0.2238/0.8939	0.2467/0.9229

Themis are often able to achieve no reduction in the clean nmAP, they are unable to tradeoff performance in clean images to attain *Saliuitl*'s adversarial nmAP. Note also that in terms of nmAP, *Object Seeker* is dominated by *Saliuitl* in all scenarios. As illustrated in Figure 1, the success of *Object Seeker* in terms of recovery rate and lost predictions comes at the cost of a significant amount of duplicate predictions or predictions of non-existing objects.

In a realistic deployment context, it is reasonable to assume that the vast majority of inputs fed into a model would not contain patch attacks. Hence, maintaining a low rate of lost predictions is extremely important. In Figures 5 and 6 in the Appendix we highlight the recovery rate achieved by each method, for different thresholds on the rate of lost predictions and inflicted attacks, for which a low rate is also desirable. In all cases *Saliuitl* is able to maintain a very low rate of inflicted attacks compared to other methods. Naturally, reducing lost predictions takes precedence over reducing inflicted attacks in a realistic scenario. For Pascal VOC (Figure 6), *Saliuitl* can achieve a substantial recovery rate at a lower rate of lost predictions than *Object Seeker* in all scenarios, although the latter can attain a higher recovery rate. Only *Themis* and *Themis-50* achieve a non-zero recovery rate at a zero lost prediction rate for all attacks, but this comes at the cost of both a very low recovery rate and a significant inflicted attack rate. In most scenarios *Saliuitl* is the only approach that attains a high recovery rate at a low rate of inflicted attacks, with the exception of *Object Seeker* for the multi-object patch scenario.

Key Advantages: *Saliuitl* (i) dominates all non-certifiable baselines, (ii) adapts better to triangular and multiple patches, and (iii) compared with *Object Seeker*, it performs better in terms of nmAP and more consistently across datasets for fixed thresholds on lost predictions and inflicted attacks.

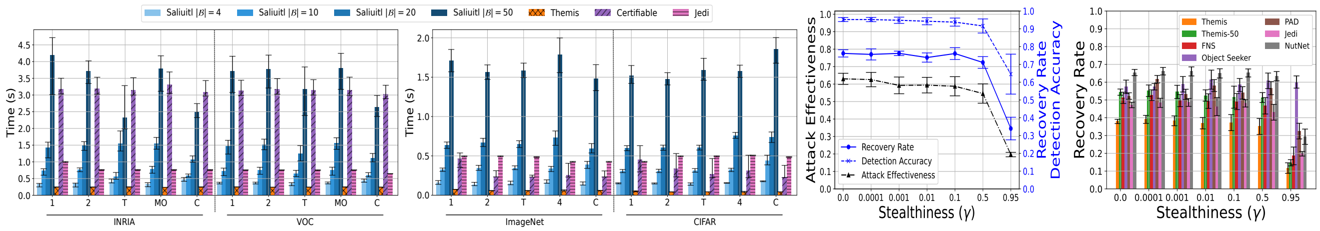
Image Classification. We next consider corresponding results for the image classification task. The last eight rows of Table 1 show the recovery performance of *Saliuitl* and the baselines, and show that *Saliuitl* outperforms all baselines, with the exception of *Themis* for single patch attacks on ImageNet, however, the fact that *Saliuitl* achieves a better performance than *Themis-50* shows that this result is sensitive to the strong assumption that the patch size is known by *Themis*; this is also coherent with the fact that, for triangular and multiple patches, *Saliuitl* outperforms *Themis*, which despite knowing the bound on the total patch size, fails to accurately locate the adversarial regions due to its usage of single square windows to find patch candidates.

Figures 9 and 10 in the Appendix show the detailed tradeoff between recovery rate and lost predictions. The figures show that *Saliuitl* significantly outperforms all baselines in image classification scenarios, with the exception of *Themis* for single patch attacks on ImageNet, which is in fact the sole instance of a method outperforming *Saliuitl* for image classification. Among other baselines, *NutNet* and *PAD* are the only methods capable of attaining *Saliuitl*'s high recovery rate, but this comes at the cost of a larger amount of lost predictions. Importantly, except for *Saliuitl*, *NutNet*, *PAD*, and *FNS*, all other methods perform notably worse for multiple patches. Moreover, *Saliuitl* is the defense method that best adjusts to triangular patches.

Figures 9 and 10 in the Appendix show the recovery rates obtained for different thresholds on the rate of lost predictions and inflicted attacks. For ImageNet (Figure 9), *Saliuitl* outperforms most baselines for single-patch attacks, except for *Themis* and *Themis-50*, providing a high recovery rate at the lowest rate of lost predictions across all other methods; it is also able to surpass *Themis-50*'s recovery rate at the cost of an increase in lost predictions. While *FNS* and *NutNet* can attain a lower inflicted attack rate, this comes at the cost of a higher lost prediction rate, a lower recovery rate, or both. The performance of *Saliuitl* is superior for double, quadruple, and triangular patch attacks, for which no method can attain *Saliuitl*'s high recovery rate for the same amount of lost predictions. For CIFAR-10 (Figure 10), in most scenarios *Saliuitl* achieves a recovery rate unattainable by all other methods at zero lost predictions and inflicted attacks, except for the triangular patch scenario, where *Saliuitl* experiences a non negligible inflicted attack rate.

Key Advantages: *Saliuitl* (i) essentially dominates all baselines and (ii) adapts better to multiple and triangular patch scenarios.

Computational Efficiency Adversarial patches are especially relevant for time-sensitive applications, hence com-



(a) Computational Cost (b) Adaptive - *Saliuitl* (c) Adaptive - Baselines

Figure 4. (a): *Saliuitl*'s cost compared to the baselines on all datasets; error bars denote the first and third quartiles over the dataset. (b)-(c): adaptive attack effectiveness, *Saliuitl*'s attack detection accuracy and recovery rate (b) and the baselines' recovery rate (c) vs. adaptive attack stealthiness; error bars represent one standard deviation over five generated patches for each value of γ.

putational efficiency is a crucial aspect, often ignored by recovery methods [22, 35, 41]. Useful defenses should be able to operate at a reasonable speed [18, 25, 43].

To assess the efficiency of *Saliuitl* compared to state-of-the-art approaches, we evaluate the computational cost of *Themis*, *Jedi*, *Patch Cleanser* and *Object Seeker*. We omit *FNS* and *NutNet* since their added computational cost at inference time is negligible, and we omit *PAD* due to its notably higher cost compared to all other methods. *Patch Cleanser* and *Object Seeker* provide reference points on the cost of certifiable defenses, and *Themis*, owing to its efficiency-oriented design, makes for a strong baseline. These evaluations were done on a system with a 2x Intel Xeon Gold 6130 CPU core with an NVIDIA T4 GPU.

We show our results for all datasets in Figure 4(a); we vary *Saliuitl*'s ensemble size and set the baselines to their default parameters. For simplicity, for each input $\mathbf{x}_i$ we use $\mathcal{B}_D = \mathcal{B}_R = \mathcal{B}_B = \{\frac{x \cdot \max(\mathbf{m}_i)}{|\mathcal{B}_B|}\}_{x=0}^{|\mathcal{B}_B|-1}$, and we set $\alpha^* = 0.5$. Our results show that a smaller ensemble size allows *Saliuitl* to approach the low computational cost of *Themis*, while larger ensemble sizes might result in a computational cost even above that of certifiable defenses. We observe that the cost of *Saliuitl* increases linearly with ensemble size. In the Appendix, we provide an ablation study for the sizes of the sets $\mathcal{B}_D$ and $\mathcal{B}_R$. We find that *Saliuitl* is able to tradeoff computational cost to outperform state-of-the-art defenses while maintaining a relatively low execution time that is insensitive to the number of patches.

4.3. Adaptive attacks

Finally, we evaluate *Saliuitl* against an adaptive attacker with complete knowledge of *Saliuitl*. Unlike prior evaluations [18, 35, 43], where adaptive attacks were constrained to use only the input features used by the defender to localize or filter adversarial patches, we can directly incorporate our attack detector *AD* into the adaptive attack optimization process, allowing for a stronger adaptive attacker that can balance the original attack objective with the objective of bypassing *Saliuitl* [3]. The adaptive attack aims to find

$$\hat{p} \in \arg\max_{p \in \mathcal{P}} \; \mathbb{E}_{\mathcal{D}}[(1-\gamma) \cdot \log \Pr(h(\mathcal{A}(\mathbf{x}_i, p)) \in t(\mathbf{x}_i)) + \gamma \cdot AD(\mathcal{A}(\mathbf{x}_i, p))],$$

where $\gamma \in [0, 1]$ is the weight of *stealthiness*, i.e., how much the attacker penalizes being detected by *Saliuitl*, and with some abuse of notation, $AD(\cdot)$ denotes the detection score *AD* assigns to a given input. Using this new attack objective, we generate adaptive single-patch attacks for object detection on INRIA following [36], using stealthiness values $\gamma \in \{0.0, 0.0001, 0.001, 0.01, 0.1, 0.5, 0.95\}$.

In Figure 4(b), we report *Saliuitl*'s attack detection accuracy and recovery rate, and the attacker's effectiveness for different values of γ. We note that while an adaptive attacker can reduce *Saliuitl*'s detection accuracy, and in turn, its recovery rate, this comes at the cost of a significant drop in attack effectiveness, showing that *Saliuitl* is robust against adaptive attacks. Figure 4(c) shows the recovery rate for the baselines when defending against the adaptive attack trained against *Saliuitl*. We observe that despite our adaptive attacker lacking any knowledge about the baselines, it can decrease the recovery rate of most baselines, including *PAD* and *NutNet*, which are supposed to be patch-agnostic. The exception is *Object Seeker*, because as a certifiable method, it does not rely on the patch content and can only benefit from the drop in attack effectiveness experienced by the adaptive attacker.

5. Conclusion

We introduce *Saliuitl*, a two-stage patch attack recovery method that does not detect attacks based on a single threshold, does not assume the shape or number adversarial patches, and performs attack detection explicitly before moving onto recovery. *Saliuitl* achieves substantial performance gains over existing state-of-the-art defenses and enjoys a favorable efficiency-performance trade-off. *Saliuitl* is also resilient to adaptive white-box attacks. For future work, we consider extending *Saliuitl* to other architectures (e.g. visual transformers), and a further exploration of the recovery of non-rectangular adversarial patches, since our results show that these are particularly challenging attacks for *Saliuitl* and existing defenses.

Acknowledgments

This work was partly funded by the KTH Railway Group and by the Swedish Transport Administration through Railway Excellence Area 6. We acknowledge the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725, for computational and storage resources, and for awarding this project access to the LUMI supercomputer, owned by the EuroHPC Joint Undertaking and hosted by CSC (Finland) and the LUMI consortium.

References

[1] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. In *Proc. of International Conference on Machine Learning (ICML)*, 2017. 1

[2] Tom B. Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *ArXiv*, abs/1712.09665, 2017. 1, 2

[3] Mauricio Byrd Victorica, György Dán, and Henrik Sandberg. SpaNN: Detecting multiple adversarial patches on CNNs by spanning saliency thresholds. In *Proc. of the IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, 2025. 1, 3, 5, 8

[4] Junzhe Cai, Shuiyan Chen, Heng Li, Beihao Xia, Zimin Mao, and Wei Yuan. HARP: Let Object Detector Undergo Hyperplasia to Counter Adversarial Patches. In *Proc. of the ACM Intl. Conf. on Multimedia*, 2023. 1

[5] Ping-Han Chiang, Chi-Shen Chan, and Shan-Hung Wu. Adversarial pixel masking: A defense against physical attacks for pre-trained object detectors. In *Proceedings of the 29th ACM International Conference on Multimedia*, 2021. 1

[6] Charles K. Chui and H.N. Mhaskar. MRA contextual-recovery extension of smooth functions on manifolds. *Applied and Computational Harmonic Analysis*, 28, 2010. 6

[7] Kenneth T. Co, Luis Muñoz-González, Leslie Kanthan, and Emil C. Lupu. Real-time detection of practical universal adversarial perturbations. *ArXiv*, abs/2105.07334, 2021. 1

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 5

[9] Steven Benjamin Damelin and Nguyen S. Hoang. On surface completion and image inpainting by biharmonic functions: Numerical aspects. *Int. J. Math. Math. Sci.*, 2017. 6

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 5

[11] Bao Gia Doan, Minhui Xue, Shiqing Ma, Ehsan Abbasnejad, and Damith C. Ranasinghe. TnT attacks! universal naturalistic adversarial patches against deep neural network systems. *IEEE Transactions on Information Forensics and Security (TIFS)*, 2022. 1, 2

[12] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996. 5, 4

[13] M. Everingham, L. Van Gool, and C.K.I. Williams. The PASCAL visual object classes (VOC) challenge. In *International Journal of Computer Vision*, 2010. 5, 2

[14] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1

[15] Friedrich Fraundorfer, Lionel Heng, Dominik Honegger, Gim Hee Lee, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Vision-based autonomous mapping and exploration using a quadrotor MAV. In *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012. 1

[16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *Proc. of International Conference on Learning Representations (ICLR)*, 2015. 1

[17] Husheng Han, Kaidi Xu, Xing Hu, Xiaobing Chen, Ling Liang, Zidong Du, Qi Guo, Yanzhi Wang, and Yunji Chen. ScaleCert: Scalable certified defense against adversarial patches with sparse superficial layers. In *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2021. 1, 2

[18] Husheng Han, Xing Hu, Yifan Hao, Kaidi Xu, Pucheng Dang, Ying Wang, Yongwei Zhao, Zidong Du, Qi Guo, Yanzhi Wang, Xishan Zhang, and Tianshi Chen. Real-time robust video object detection system against physical-world adversarial attacks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCADICS)*, 2023. 1, 2, 4, 5, 8

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[20] Yu-Chih-Tuan Hu, Jun-Cheng Chen, Bo-Han Kung, Kai-Lung Hua, and Daniel Stanley Tan. Naturalistic physical adversarial patch for object detectors. In *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 27

[21] Hao Huang, Ziyan Chen, Huanran Chen, Yongtao Wang, and Kevin Zhang. T-SEA: Transfer-based self-ensemble attack on object detection. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 27

[22] Lihua Jing, Rui Wang, Wenqi Ren, Xin Dong, and Cong Zou. PAD: Patch-agnostic defense against adversarial patch attacks. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2, 3, 4, 5, 8

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

[24] Shuohao Lin, Ernie Chu, Che-Hsien Lin, Jun-Cheng Chen, and Jia-Ching Wang. Diffusion to confusion: Naturalistic

adversarial patch generation based on diffusion model for object detector. *ArXiv*, abs/2307.08076, 2023. 1, 5, 2, 3, 4

[25] Zijin Lin, Yue Zhao, Kai Chen, and Jinwen He. I don't know you, but I can catch you: Real-time defense against diverse adversarial patches for object detectors. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2024. 1, 2, 3, 5, 8

[26] Jiang Liu, Alexander Levine, Chun Pong Lau, Rama Chellappa, and Soheil Feizi. Segment and complete: Defending object detectors against adversarial patch attacks with robust patch detection. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 4, 27

[27] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[28] Michael McCoyd, Won Park, Steven Chen, Neil Shah, Ryan Roggenkemper, Minjune Hwang, Jason Xinyu Liu, and David Wagner. Minority reports defense: Defending against adversarial patches. In *Proc. of Applied Cryptography and Network Security Workshops*, 2020. 1

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2019. 5

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011. 4, 5, 15

[31] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[32] Giulio Rossolini, Federico Nesti, Fabio Brau, Alessandro Biondi, and Giorgio Buttazzo. Defending from physically-realizable adversarial attacks through internal over-activation analysis. In *Proc. of the AAAI Conference on Artificial Intelligence*, 2023. 1

[33] Abhijith Sharma, Phil Munz, and Apurva Narayan. Assist is just as important as the goal: Image resurfacing to aid model's robust prediction. In *Proc. of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023. 1, 2, 27

[34] B. Sinopoli, M. Micheli, G. Donato, and T.J. Koo. Vision based navigation for an unmanned aerial vehicle. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, 2001. 1

[35] Bilel Tarchoun, Anouar Ben Khalifa, Mohamed Ali Mahjoub, Nael B. Abu-Ghazaleh, and Ihsen Alouani. Jedi: Entropy-based localization and removal of adversarial patches. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 4, 5, 8

[36] Simen Thys, Wiebe Van Ranst, and Toon Goedemé. Fooling automated surveillance cameras: Adversarial patches to attack person detection. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019. 1, 5, 8, 2, 3, 4

[37] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in Python. *PeerJ*, 2014. 4, 19, 22

[38] Siyang Wu, Jiakai Wang, Jiejie Zhao, Yazhe Wang, and Xianglong Liu. NAPGuard: Towards detecting naturalistic adversarial patches. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 2

[39] Chong Xiang and Prateek Mittal. PatchGuard++: Efficient provable attack detection against adversarial patches. In *Proc. of International Conference on Learning Representations Workshops (ICLRW)*, 2021. 1, 5, 2

[40] Chong Xiang, Saeed Mahloujifar, and Prateek Mittal. PatchCleanser: Certifiably robust defense against adversarial patches for any image classifier. In *Proc. of USENIX Security Symposium*, 2022. 1, 2, 3, 4, 5

[41] Chong Xiang, Alexander Valtchanov, Saeed Mahloujifar, and Prateek Mittal. ObjectSeeker: Certifiably robust object detection against patch hiding attacks via patch-agnostic masking. In *Proc. of IEEE Symposium on Security and Privacy*, 2023. 2, 3, 4, 5, 8

[42] Zirui Xu, Fuxun Yu, Chenchen Liu, and Xiang Chen. LanCeX: A versatile and lightweight defense method against condensed adversarial attacks in image and audio recognition. *ACM Trans. Embed. Comput. Syst.*, 2022. 1

[43] Cheng Yu, Jiansheng Chen, Yu Wang, Youze Xue, and Huimin Ma. Improving adversarial robustness against universal patch attacks through feature norm suppressing. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2023. 1, 2, 4, 5, 8

[44] Yanjing Zhang, Jianming Cui, and Ming Liu. Research on adversarial patch attack defense method for traffic sign detection. In *Cyber Security*. Springer Nature, 2022. 1