This work has been accepted for publication in the IEEE Conference on Secure and Trustworthy Machine Learning (SaTML). The final version will be available on IEEE Xplore.

SpaNN: Detecting Multiple Adversarial Patches on CNNs by Spanning Saliency Thresholds

Mauricio Byrd Victorica
KTH Royal Institute of Technology
Stockholm, Sweden
mbv@kth.se

György Dán
KTH Royal Institute of Technology
Stockholm, Sweden
gyuri@kth.se

Henrik Sandberg
KTH Royal Institute of Technology
Stockholm, Sweden
hsan@kth.se

***Abstract*—State-of-the-art convolutional neural network models for object detection and image classification are vulnerable to physically realizable adversarial perturbations, such as patch attacks. Existing defenses have focused, implicitly or explicitly, on single-patch attacks, leaving their sensitivity to the number of patches as an open question or rendering them computationally infeasible or inefficient against attacks consisting of multiple patches in the worst cases. In this work, we propose SpaNN, an attack detector whose computational complexity is independent of the expected number of adversarial patches. The key novelty of the proposed detector is that it builds an ensemble of binarized feature maps by applying a set of saliency thresholds to the neural activations of the first convolutional layer of the victim model. It then performs clustering on the ensemble and uses the cluster features as the input to a classifier for attack detection. Contrary to existing detectors, SpaNN does not rely on a fixed saliency threshold for identifying adversarial regions, which makes it robust against white box adversarial attacks. We evaluate SpaNN on four widely used data sets for object detection and classification, and our results show that SpaNN outperforms state-of-the-art defenses by up to 11 and 27 percentage points in the case of object detection and the case of image classification, respectively. Our code is available at https://github.com/gerkbyrd/SpaNN .**

***Index Terms*—Convolutional neural networks, adversarial machine learning, adversarial patch attacks.**

I. Introduction

Deep learning models achieve state-of-the-art performance on computer vision tasks, but they are vulnerable to adversarial attacks, i.e., input perturbations crafted to change the model's output [1]–[3]. One class of adversarial attacks is digital attacks, which involve imperceptible perturbations of the input image, often bounded in some ℓ_p norm. Several digital attack generation methods have been proposed in the past decade [1], [4]–[6], followed by corresponding defense schemes [7]–[10]. These attacks assume the adversary has direct access to the pixels of the input image provided to the model.

In more recent years, focus has shifted towards physically realizable attacks [11]. They differ from digital attacks in that they are spatially constrained, and they typically involve applying a printable patch containing an adversarial pattern to an object in the physical scene. For instance, an adversarial patch can be applied in the form of a sticker [2], [12], a printed pattern on clothing [3], [12], or a projected image [13]. Unlike digital attacks, patch attacks do not assume access to the digital images in the deep learning model's processing pipeline, and instead manipulate physical objects in the scene, which makes their implementation more feasible and eliminates the need to access the victim model's input directly.

Existing defenses against patch attacks either aim at detecting adversarial patches [14]–[21] or at recovering from patch attacks by localizing the patches and removing them [20], [22]–[30]. The approach they follow for detecting patches is based on the patches' impact on statistical properties of the input data, e.g., by computing gradients in the pixel domain [27], by detecting unusually high activations in feature maps [15], [22], or by detecting high entropy regions in pixel space [23]. As a result, existing approaches for detecting patch attacks against convolutional neural networks (CNNs) suffer from two main limitations. First, most methods, explicitly or implicitly, assume a single patch per object [21], [23], [29], or even a single patch per image [22], [24], making them vulnerable to attacks deviating from such assumptions. Second, they are based on one or more detection thresholds that are compared to image statistics in one or more feature spaces (e.g., thresholds on image entropy [23] or internal layers' neural activations [15], [17], [22]), and hence they require parameter tuning and adjustments to changes in the defended model or the input data distribution.

In this paper, we propose *SpaNN*, a novel patch attack detection method that overcomes limitations of existing defenses. *SpaNN* achieves superior detection performance owing to two key ideas. First, detection in *SpaNN* is based on how the spatial patterns of important neurons in a shallow feature map *change* as the definition of *important neurons* changes. Second, the pattern changes used to distinguish attacked images from clean images are independent of the number of patches in the image. These two design choices make it possible for *SpaNN* to detect attacks regardless of the number of patches, while making it robust to adaptive attacks that maximize impact subject to remaining undetected. Our main contributions are as follows:

i) We propose *SpaNN*, an approach for detecting multiple adversarial patches based on the clustering analysis of an ensemble of binarized saliency maps in feature space.
ii) We evaluate the proposed detection method on various object detection and image classification tasks and show that *SpaNN* achieves an effective attack detection accuracy of at least 86.13% for object detection and 96.64% for image classification, for any number of patches.

iii) We compare *SpaNN* to various baselines and show that it achieves state-of-the-art performance on single-patch detection, and establishes the new state-of-the-art for multiple-patch attacks.
iv) In further experiments, we show that the computational cost of *SpaNN* is independent of the number of patches, and evaluate its effectiveness against an adaptive attacker.

The rest of the paper is organized as follows. We discuss related works in Section II. We introduce the relevant background in Section III, and present *SpaNN* in Section IV. We present numerical results in Section V and we conclude the paper in Section VI.

II. Related Work

Mechanisms to detect patch attacks against image classification and object detection CNN models have been proposed in recent works [15], [21]–[24]. In *Themis* [22], a sliding window is applied on the feature map produced by the first convolutional layer of a CNN model, to identify "patch candidates", i.e., relatively dense areas in terms of neural activity. Attacks are then detected based on the effect that occluding the patch candidates has on the model's output. *Jedi* [23] computes an entropy heat map for the input image using a threshold that is adjusted for each input image and then uses filtering and post-processing to keep only non-sparse high-entropy areas in the heat map. An autoencoder is used to construct patch masks corresponding to high-entropy areas, and the masks are applied to the input image before feeding it to the CNN model. *Z-Mask* [15] is a similar method to *Jedi*, where two over-activation heat maps are computed using *Spatial Pooling Refinement* (one focusing on the defended model's shallow layers and the other on its deep layers). The heat maps are processed using two MLPs and simple aggregations, resulting in a mask for over-activated input areas and a scalar measure of over-activation. If the scalar measure is above a given threshold, then the mask is applied to the input image.

NAPGuard [21] trains a modified YOLOv5 object detector to detect only the "patch" class; to achieve good performance, the loss function used during training encourages the detector to accurately detect the high-frequency aggressive features of adversarial patches, and a low-pass filter is used at inference time to suppress natural features and facilitate the detection of patches. *PAD* [29] analyses images through a sliding window to generate semantic independence and spatial heterogeneity heatmaps. After fusing the two heatmaps, the regions which may contain adversarial patches are determined with respect to a threshold that depends on the statistics across the image; *PAD* then relies on the Segment Anything (SAM) [31] image segmentation model to produce adequate patch masks.

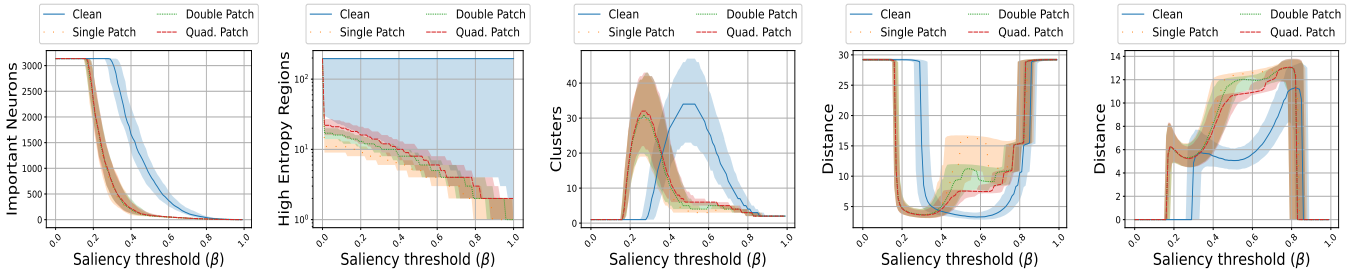
Slightly different from the above approaches, certifiable methods aim at providing formal guarantees for a given attack model [14], [19], [24]. *Object Seeker* [24] is a certifiable recovery method specific to object detection, and relies on a two-step process consisting of (i) patch-agnostic masking, where horizontal and vertical lines are used to split the image into two parts at k interpolations on each axis, and (ii) pruning, where the objects detected in masked images are filtered, merged, and subsequently pruned to obtain a robust final inference. In short, a filtered set of masked bounding boxes containing the ones dissimilar enough from the originals is clustered, and a representative from each cluster is selected. The final output consists of the pruned new boxes and those detected on the original image.

Closely related to our work is *ViP* [19], which is a certifiable detection and recovery method for patch attacks on Vision Transformer models (ViTs) for image classification, explicitly addressing the double-patch attack scenario. As with other methods, *ViP* relies on applying a set of masks to the input image and analyzing the corresponding set of predictions, assuming that at least one mask occludes the patch attack. An attack is detected if any two predictions are inconsistent, and clean predictions can be recovered by majority voting. To mask double patches, *ViP* uses *generalized windows*, which cover disjoint input regions, guaranteeing the occlusion of any two patches of known size. ViT models are leveraged to implement adequate *base classifiers*, which predict labels using a subset of the tokenized input image. Each mask thus corresponds to the unused input regions of a base classifier. Since it focuses on ViT models instead of CNNs, we do not consider *ViP* as a relevant baseline for our work. We note that there is no need to restrict ourselves to attack detection methods, since attack recovery methods also perform detection internally. Moreover, most state-of-the-art adversarial patch defenses focus on recovery, and usually, some form of attack detection is at their core [22]–[24], [29].

Fixed saliency thresholds. In *Themis*, a neural activation threshold β determines what neurons should be considered important. Important neurons are used to construct a binarized feature map, and if the number of important neurons in a given area exceeds a second threshold θ, then the area becomes a *patch candidate* [22]. In *Jedi*, entropy heat maps are constructed using a dynamic threshold, which is computed based partly on the input image, partly on pre-computed statistics for clean images, and partly on hyper-parameters chosen empirically [23]. Similarly, the over-activation heat maps in *Z-Mask* are constructed using pre-computed statistics for activation values of clean images [15]. Even *Object Seeker*, which aims to be agnostic to the attack model, tunes the victim model's confidence threshold to detect bounding boxes on masked images adequately [24].

A main shortcoming of these methods is that the optimal threshold values depend on either the data set, the model under attack, or the attack formulation.

Dealing with multiple patches. *Themis* operates under the assumption that at most a single patch can be present in any given image [22]. *Object Seeker* is also formulated for a single-patch attack, and while a proof of concept for two patches is presented, it requires expensive computations and, unlike the regular method, is not patch-agnostic [24]. While in principle most recent works apply to attacks that place multiple-patches *per object*, their evaluation does not address this scenario [15],



(a) Number of important neurons. (b) Number of high entropy regions. (c) Number of clusters. (d) Average mean intra-cluster distance. (e) Standard deviation of mean intra-cluster distance.

Fig. 1: Input characteristics vs. saliency threshold β. Lines represent the median for each quantity, and shaded regions show the first and third quartiles.

[21], [23], [29]. Focusing on single-patch attacks is a common limitation among defense methods for CNN models and for object detection models in general.

III. Preliminaries

In what follows we define the attack model and formulate the attack detection problem. We then motivate our approach by illustrating the relationship between saliency thresholds and input characteristics induced by adversarial patches.

A. Adversarial Patches and Detection Problem

For a machine learning model h (e.g., used for image classification, object detection, etc.), and a set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) : \mathbf{x}_i \in \mathcal{X}, \mathbf{y}_i \in \mathcal{Y}\}$ of input-output pairs, we define the attacker's target output $t(\mathbf{x}_i)$ as the inference the model should output given input $\mathbf{x}_i$. The adversarial attack model is defined by the set of perturbations $\mathcal{P}$ that the attacker can choose from and by the transformation function $\mathcal{A}$, which is used to apply a perturbation $p \in \mathcal{P}$ to input $\mathbf{x}_i$. Hence, for a model h the attacker aims to find a perturbation that minimizes the loss function

$$\mathcal{L}(p) = -\mathbb{E}_{\mathcal{D}}[\log \Pr(h(\mathcal{A}(\mathbf{x}_i, p)) \in t(\mathbf{x}_i))],$$

i.e., it aims to find $\hat{p} \in \arg\min_{p \in \mathcal{P}} \mathcal{L}(p)$. For a targeted attack, the attacker's target output $t(\mathbf{x}_i)$ is a particular $\mathbf{y}'_i \neq \mathbf{y}_i$, i.e., $t(\mathbf{x}_i) = \mathbf{y}'_i$. For an untargeted attack, $t(\mathbf{x}_i)$ is any output different from the clean output $\mathbf{y}_i$, i.e., $t(\mathbf{x}_i) \in \mathcal{Y} \backslash \{\mathbf{y}_i\}$. Adversarial patches are the most explored physically realizable attack model in the literature [2], [3], [12], [32], [33]. In the case of adversarial patches, $\mathcal{P}$ defines the number, size, shape, location, and pixel value range of adversarial patches, while $\mathcal{A}$ is the replacement operation where the corresponding pixels in the input $\mathbf{x}_i$ are replaced by the patch p.

Now consider a set $\mathcal{D} = \{(\mathbf{x}_i, z_i) : \mathbf{x}_i \in \mathcal{X}, z_i \in \{0, 1\}\}$ of input-label pairs, where the label indicates whether or not the input has been subject to an adversarial attack. An attack detector $\mathcal{F}_\phi$ parametrized by $\phi \in \Phi$ should thus predict the label for each input $\mathbf{x}_i \in \mathcal{X}$, and the objective is to find detector parameters that minimize the loss function

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathcal{D}}[\log \Pr(\mathcal{F}_\phi(\mathbf{x}_i) \neq z_i)],$$

i.e., the goal is to find $\hat{\phi} \in \arg\min_{\phi \in \Phi} \mathcal{L}(\phi)$. $\mathcal{F}_\phi$ and Φ vary widely between attack detection methods proposed in the literature, but most mechanisms proposed to detect patch attacks include a saliency threshold $\beta \in \mathbb{R}$ among their parameters ϕ [22], [23], [29]. The saliency threshold is compared to features computed from $\mathbf{x}_i$, e.g., entropy [23] or the neural activations in a hidden layer [22], and as such its choice has a significant impact on $\mathcal{F}_\phi$, as we show next.

B. Input Characteristics Across Thresholds

To illustrate the dependence of the attack detection results on the choice of the threshold that is used to detect regions containing patch attacks, we selected a random subset of 3,334 images from the ImageNet validation set and created three attacked versions for each image, using one, two, and four adversarial patches. We also extracted feature maps from all clean and attacked images using the ResNet-50 CNN model [34].

We then computed the summary statistics on neural activation and entropy, used for detection by *Themis* and *Jedi*, respectively. For neural activation, for each feature map M we extracted, we compute its maximum neural activation $\max(M)$, and for a threshold $\beta \in [0, 1]$ we calculate the number of neurons with activation above (or equal to) $\beta \cdot \max(M)$, i.e., the number of important neurons [22]. For entropy, we split each image into multiple regions using a sliding window and calculated the entropy H of each region. We then compute the maximum entropy H_{max} among all regions in the image, and for a threshold $\beta \in [0, 1]$ we calculate the number of regions with activation above (or equal to) $\beta \cdot H_{max}$, i.e., the number of high entropy regions [23].

In Figures 1(a)-(b) we present summary statistics for the number of important neurons and high entropy regions, across all clean and attacked inputs. The results show that the choice of β determines the ability to discriminate between attacked and clean images based on these features, i.e., a detector based on one of these summary statistics must set β to a value where the curve for clean images does not overlap with curves for patched images. Once a specific β is chosen, an attacker might adapt its attack to generate inputs that are similar to clean

inputs for a particular choice of β. Hence, choosing a single value of β makes an attack detector brittle.

Thus, instead of computing features for a particular saliency threshold β, we propose to base detection on how a carefully selected set of features changes as a function of β. A simple choice would be to use the previously considered features, as Figures 1(a)-(b) show that the shapes of the curves as a function of the saliency threshold β are substantially different for attacked and clean images regardless of the number of patches. Nonetheless, features better than these can be constructed by considering the spatial distribution of important neurons in the feature map, inspired by the observation that adversarial patches affect localized areas of the feature map [14], [22]. For designing new features, we binarized each feature map M using different values of the importance threshold β, i.e., for each M and β we computed a binary version of M, replacing with zeros the elements with values below $\beta \cdot \max(M)$ and replacing all other elements with ones. We performed clustering on the binarized feature maps using DBSCAN, and for each β and each M, we computed the number of clusters, the mean average intra-cluster distance, and the mean standard deviation of intra-cluster distances. We show summary statistics of these quantities as a function of the threshold β in Figures 1 (c) - (e). We observe that for each quantity, there is a notable difference between the curves corresponding to clean and patched images for any number of patches. This observation is the basis of the detection scheme we propose next. Importantly, our approach does not rely on any single saliency threshold to distinguish attacked images from clean ones, which makes it less vulnerable to evasion attacks.

IV. Clustering-based Attack Detection from Binarized Feature Map Ensembles

Our proposed approach consists of three steps, performed on input $\mathbf{x}_i$ to a given CNN model h: (i) computing an ensemble of binarized feature maps, (ii) executing a clustering algorithm for each element in the ensemble, and (iii) classifying $\mathbf{x}_i$ as benign or adversarial based on the clustering results across the ensemble. Our method is illustrated in Figure 2.

A. Computing Ensembles of Binarized Feature Maps

Given a CNN model h and an input $\mathbf{x}_i$, we sum the output of intermediate layer ℓ across channels to obtain the feature map $M = h_\ell(\mathbf{x}_i)$; hence M is two-dimensional. For a set $\mathcal{B} = (\beta_1, \ldots, \beta_B)$ of threshold values, the key tenet of the proposed attack detector is to compute a binarized feature map B_b from M for each threshold $\beta_b \in \mathcal{B}$. For threshold β_b the binarized feature map B_b has the same dimensions as M, and binary entries $B_{bij} = \mathbb{1}_{M_{ij} \geq \beta_b \cdot \max(M)}$. We thus obtain a total of $|\mathcal{B}|$ binarized feature maps. Note that any threshold β_b must be between 0 and 1.

B. Clustering of Binarized Feature Maps

The second step is to characterize the spatial distribution of nonzero entries in each binarized feature map. We do so by clustering each binarized feature map B_b using DBSCAN [35]. DBSCAN aims to find areas of high density in data space, in terms of the Euclidean distance ϵ between data points. Clusters are formed according to core samples, data points with at least $w_{\min}$ neighbors at a distance lower than ϵ. Any data points not neighboring or being a core sample are discarded as outliers.

There are three main reasons for choosing DBSCAN. First, it is a density-based approach, aligning with the notion that adversarial patches result in dense localized areas of important neurons. Second, DBSCAN does not require tuning hyperparameters such as the number of clusters. Third, it is widely available, easy to implement, and computationally efficient.

C. Construction of Clustering Features and Classification.

Given the clustering results, we compute for each threshold β_b the number n_c of clusters, the mean average intra-cluster distance $\overline{d_{ic}}$ (i.e., the mean distance between points in a cluster, averaged over all clusters obtained for B_b), the standard deviation $\sigma(d_{ic})$ of the average intra-cluster distance, and the number of important neurons n_{imp}, i.e., the number of non-zero elements. We thus obtain $4B$ quantities, which we use to construct a clustering feature vector $s \in \mathbb{R}^{4 \times B}$, where the rows of s are one-dimensional curves of length B corresponding to each clustering metric. We preprocess s by normalizing it over its second dimension, and then re-scaling and centering it around zero so that each row of s has its values between -1 and 1. s is used as input to *AD*, a 4-channel one-dimensional CNN, taking each row of s as a separate channel; *AD* has an intentionally simple architecture, as it can be trained quickly, is less prone to overfitting, and allows fast inference. The parameters of *AD* are as follows.

- 1D convolutional layer: 4 input channels, kernel size = 2, stride = 1, 12 output channels. Followed by 1D average pooling, 1D batch-norm, and ReLU activation function.
- 1D convolutional layer: 12 input channels, kernel size = 2, stride = 1, 12 output channels. Followed by 1D average pooling, 1D batch-norm, ReLU activation function, and a flattening operation to pass a single-dimensional input to the next layer.
- Fully connected layer: 144-dimensional input, 576 units, followed by ReLU activation function.
- Fully connected layer: 576-dimensional input, 576 units, followed by ReLU activation function.
- Output layer: 576-dimensional input, 1 unit, followed by a sigmoid activation function.

The output of *AD* is the detection score (between 0 and 1), which is used for identifying whether or not there is an adversarial patch in input $\mathbf{x}_i$. The pseudocode of the proposed attack detector is shown in Algorithm 1.

V. Numerical Results

We use our clustering-based approach to detect single and multiple patch attacks against commonly used models performing object detection and image classification. We evaluate multiple datasets widely used in the literature on patch attacks, and compare against relevant state-of-the-art baselines.

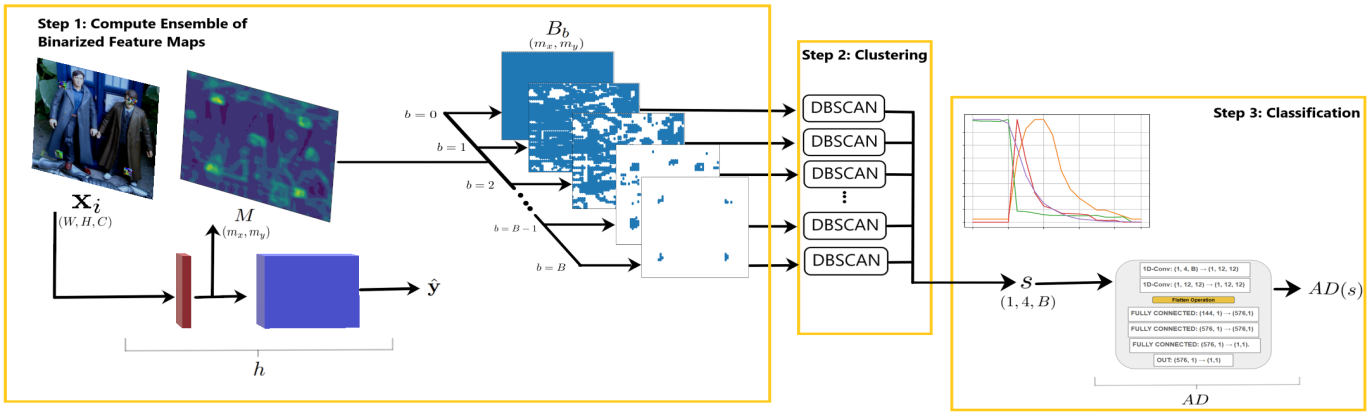


Fig. 2: *SpaNN*: For any input $\mathbf{x}_i$, after extracting a feature map M from a shallow layer of the victim model h, a binarized feature map B_b is obtained for each threshold β_b in the set $\mathcal{B}$. DBSCAN is applied to each element in the ensemble, and the resulting clustering feature vector s is fed to the neural network AD, which outputs an attack detection score $AD(s)$.

(a) Single patch. (b) Double patch. (c) Single patch. (d) Double patch. (e) Quad. patch.

Fig. 3: Single and multiple patches for object detection (a-b) and for image classification (c-e).

A. Experimental Setup

Models. We use YOLOv2 [36] to perform object detection for three reasons. First, YOLOv2 is widely available and is computationally efficient. Second, the attack model we considered to train *SpaNN* was developed for and evaluated on YOLOv2 [32]. Third, the baseline attack detection schemes were initially evaluated on YOLOv2 [22] or on later versions of YOLO [23], [24]; YOLOv2 is a representative architecture of later versions, and in general, of state-of-the-art one-stage CNN-based object detectors. For image classification, we use the widely used ResNet-50 [34] model, representative of state-of-the-art CNN models for image classification.

Data. We use the INRIA Person [37] (614 training and 288 test images) and Pascal VOC 2007 [38] (4947 training and 4953 test images) datasets for object detection. For image classification, we use the ImageNet [39] validation set (50,000 images) and the CIFAR-10 [40] test set (10,000 images). We focus mainly on INRIA and ImageNet, and report additional results for Pascal VOC and CIFAR-10 in the appendix.

Patch Attack Models. We use state of the art patch attacks against object detection and image classification, as follows. *For object detection* adversarial patches are *created* following the attack model presented by Thys et. al [32] during training and when optimizing defense-aware adaptive patches. A 300 $\times$ 300 pixel patch is optimized to minimize the objectness score of the model under attack for a given object (i.e., the patch attack aims to make objects "disappear"). For the evaluation we use a patch not used during training: the diffusion-based naturalistic *DM-NAP-Princess* patch [33]. This patch is readily available and is more challenging to detect than most other patches available in the GAP dataset [21], which was constructed to evaluate patch attack detection methods. We refer to the appendix for further evaluations on other attacks from the GAP dataset. We *apply* all adversarial patches

Algorithm 1 *SpaNN*.

Require: Model h, Attack detector AD, set of thresholds $\mathcal{B} \in [0,1]^B$, input data $\mathcal{X}$
  for $\mathbf{x}_i \in \mathcal{X}$ **do**
    $M, \hat{\mathbf{y}} \leftarrow h(\mathbf{x}_i)$ ▷ $M \in \mathbb{R}^{m_x \times m_y}$ is a feature map
    $s := \{\}$ ▷ Empty sequence to be filled
    for $\beta_b \in \mathcal{B}$ **do**
      $t \leftarrow \beta_b \cdot \max(M)$ ▷ Importance threshold
      $B_b := M \geq t$ ▷ $B_{bij} := \mathbb{1}(M_{ij} \geq t)$
      $n_{imp_b} \leftarrow \sum_{i,j} B_{bij}$
      $n_{cb}, \overline{d_{icb}}, \sigma(d_{ic})_b \leftarrow \text{Clustering}(B_b)$
      $s \leftarrow s \cup \{n_{cb}, \overline{d_{icb}}, \sigma(d_{ic})_b, n_{imp_b}\}$
    end for
    $s \leftarrow \text{Preprocess}(s)$
    return $AD(s)$ ▷ Attack detection model output
  end for

following Thys et al. [32]. For a single-patch attack on an object in an image, the square patch is re-scaled to occupy 20% of the total area of the bounding box the model outputs for the given object, and it is placed in the center of said bounding box. For an attack with two patches on an object, we re-scale each patch to occupy 10% of the attacked object's bounding box, and place the patches diagonally reflected from each other w.r.t. the center of the bounding box, as illustrated in Figures 3 (a)-(b). The patches differ in location but have the same pixel content, shape, and size. Note that unlike previous works, we consider multiple patches on the same object.

We say that a patch attack is *effective* if at least one of the detected objects in the clean inference $h(\mathbf{x}_i)$ has an overlap of no more than 50% with each detected object in the model's inference on the perturbed input $h(\mathcal{A}(\mathbf{x}_i, p))$. A true positive occurs when a patch attack $\mathcal{A}(\mathbf{x}_i, p)$ is detected by the detector. A false alarm (false positive) occurs when an attack is detected for a clean image $\mathbf{x}_i$.

For image classification, adversarial attacks are created following the open source implementation of the *PatchGuard++* defense [14]. For a region in pixel space corresponding to a single patch, with a fixed size of 32×32 pixels and a randomly chosen location, the pixels within the region are optimized to maximize the cross-entropy loss corresponding to the attacked model's prediction of the correct label. For multiple patches, besides the single-patch region, new regions are added symmetrically reflected within the complete image area, as illustrated in Figures 3(c)-(e). Note that in contrast to the patches used for object detection, each patch attack on image classification is optimized for a specific image, i.e., patches in the test set are not used during training.

We say that a patch attack is *effective* if the classifier model inference $h(\mathcal{A}(\mathbf{x}_i, p))$ is different from the ground truth $\mathbf{y}_i$. A true positive occurs when a perturbed image $\mathcal{A}(\mathbf{x}_i, p)$ is detected as attacked. A false alarm (false positive) occurs when an attack is detected for a clean image $\mathbf{x}_i$. Given a dataset $\mathcal{X}$ of clean and attacked images, we define the attack detection accuracy as the fraction of correct inferences over $\mathcal{X}$, considering both true positives (*TP*) and true negatives (*TN*). Note that these quantities can be computed over effective attacks, non-effective attacks, or both; in the sequel, we always state the type of attacks considered. We define the attack detection rate as the fraction of detected attacks, i.e., the recall over $\mathcal{X}$ including both effective and non-effective attacks.

B. Attack Detector Parameters

DBSCAN parameters. For evaluation, we used DBSCAN parameters $\epsilon = 1$ and $w_{\min} = 4$ (i.e., points with a Euclidean distance of ≤ 1 are clustered together, and at least 4 points must be grouped together to be considered a cluster). The parameter ϵ is set considering that unimportant and important neurons should not be clustered, and only neurons directly adjacent to each other in the feature map should be clustered. The choice of $w_{\min}$ captures the consideration that less than 4 adjacent neurons, important or not, are too few to be considered a cluster.

***AD* training setup** For INRIA and Pascal VOC, 20% of the training set is used for training, and the complete test set is used for evaluation. For ImageNet and CIFAR-10, 2% and 2.5% of the validation sets are used for training, respectively, and the rest of each validation set is used for evaluation. For any dataset, single-patch attacks are applied on the training samples, yielding a labeled training set of clean and attacked samples; 20% of this training set is randomly selected and set aside as a validation set during training. To train *AD*, we minimize the binary cross-entropy loss using Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, batch size of 1, and learning rate of 0.0001. We implement our model using PyTorch 1.13.1 and use the default parameters and initializations for all layers [41]. We shuffle the training set before each epoch, and stop training after 200 epochs without improvement on the validation loss. Unless otherwise noted, the default $\mathcal{B}$ used throughout our evaluation consists of 20 equidistant thresholds starting from (and including) 0, i.e. $\mathcal{B} = \{0, 0.05, ..., 0.95\}$. Note that for image classification, we consider only clean images that are correctly classified by the ResNet-50 victim model, and their corresponding attacked versions.

C. Attack Detection Baseline Methods

***Themis*-detect.** In *Themis* [22], the feature map M is binarized using a threshold β. A sliding window is then applied on the resulting binarized feature map B, and patch candidates are the image regions associated with the windows in which the fraction of non-zero entries is above a threshold θ. *Themis*-detect issues an alert whenever it finds at least one patch candidate whose occlusion would modify the output $h(\mathbf{x}_i)$ (it does not cover patch candidates to recover from patch attacks).

***ObjectSeeker*-detect.** *Object Seeker* [24] uses k_x horizontal and k_y vertical lines, and it splits the input $\mathbf{x}_i$ into two halves in pixel space using each line, one at a time. It then occludes each of the resulting halves separately and feeds the object detector h with each of the $2 \cdot (k_x + k_y)$ masked inputs. Given *Object Seeker* considers objects detected in masked inputs as distinct from those in $h(\mathbf{x}_i)$ when their intersection over area (IoA) is below some threshold τ for any object in $h(\mathbf{x}_i)$, *ObjectSeeker*-detect computes the lowest IoA across masked input detections, denoted α, and outputs the attack detection score $1 - \alpha$. Note that *ObjectSeeker*-detect cannot be used to detect attacks on image classification.

***Jedi*-detect.** *Jedi* computes the entropy over a sliding window in pixel space to obtain a heat map. Entries of the heat map that exceed the entropy threshold, determined based on the current input $\mathbf{x}_i$ and on pre-computed statistics for clean images, are retained. It then removes scattered clusters from the truncated heat-map, and feeds the truncated heat-map in an autoencoder trained to reconstruct patch masks. The reconstructed heat map is applied as a mask to the original input $\mathbf{x}_i$, and the masked input is fed into h. In *Jedi*-detect, an attack is detected if the output for the masked input differs from $h(\mathbf{x}_i)$.

***NAPGuard*.** *NAPGuard* uses a one-class object detector based on the YOLOv5 model to detect adversarial patches. The model is trained using an aggressive feature aligned loss

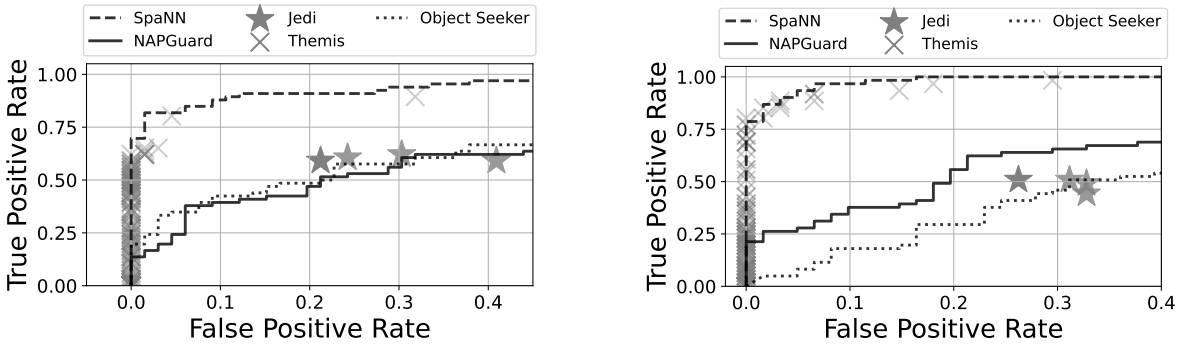


Fig. 4: Attack detection vs. false alarm rate for single (left) and double (right) adversarial patches for object detection (INRIA).

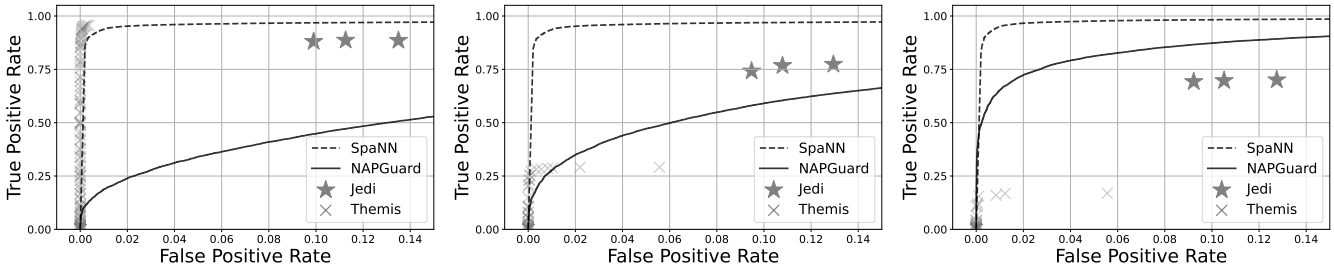


Fig. 5: Attack detection vs. false alarm rates for single (left), double (middle), and quadruple (right) adversarial patches for image classification (ImageNet).

(AFAL) and images are pre-processed to remove natural features (those below a frequency threshold) in order to facilitate the detection of adversarial patches. We use the maximum objectness score among patches detected by NAPGuard in a given image as the detection score for that image.

D. Results

We start the evaluation by considering the receiver operating characteristic (ROC) curves, i.e., the true positive rate vs the false positive rate, obtained by varying the detection threshold. Since *SpaNN*, *NAPGuard*, and *ObjectSeeker*-detect output a detection score, their ROC curves can be obtained easily. To obtain ROC curves for the other baselines, we use different values of their internal (by design fixed) detection thresholds. For *Themis*-detect, we vary β and θ from 0.05 to 0.95 in 0.05 increments, and display results only for Pareto optimal configurations. For *Jedi*-detect, we vary the threshold on the auto-encoder output, which determines the final mask applied to input images, changing it to 0.5, 0.25, and 0.125 of its original value. Note that we use the default settings for natural feature suppression in *NAPGuard* and the default number of splitting lines $k_x = k_y = 30$ for *ObjectSeeker*-detect.

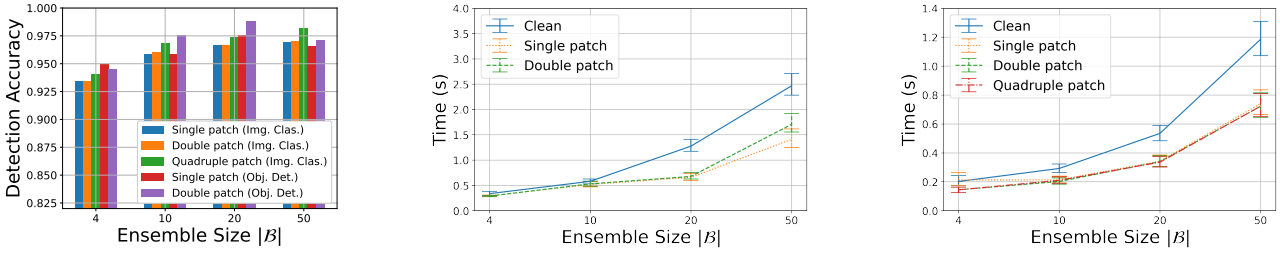
Object Detection. We first evaluate the attack detection performance on the INRIA test set defined in Section V-B. We consider effective attacks because some baselines are ill-equipped to detect ineffective attacks. Figure 4 shows the ROC curves obtained for *SpaNN*, *NAPGuard*, *Jedi*-detect, *Themis*-detect, and *ObjectSeeker*-detect for single- and double-patch attacks. The figure shows that *SpaNN* significantly outperforms all baselines regarding the true positive rate for both single and double-patch attacks, at the cost of a very low false alarm rate. The figure also shows that, aside from *Themis*-detect for the double-patch case, the highest true positive rate achievable by the baseline methods is far below that of *SpaNN*. Moreover, note that *Jedi*-detect and *ObjectSeeker*-detect experience a decrease in performance in the double-patch case. The corresponding results for Pascal VOC are available in Figure 10 in the appendix, where the superiority of *SpaNN* over the baselines in terms of detected attacks and false alarms becomes emphasized. We report the attack detection accuracy achieved by each detector using their best-performing setting in the first eight rows of Table I. The table confirms that *SpaNN* significantly outperforms the baselines.

A significant advantage of *SpaNN* compared to the recovery-based baselines is that it does not rely on the victim model's final output to detect attacks. This allows *SpaNN* to detect patch attack attempts that fail to change the model's output, i.e., ineffective ones, and hence it becomes possible to detect an attack already before it becomes successful, providing improved situational awareness. Table I shows that, for non-effective attacks, the accuracy of *SpaNN* is close to that for effective attacks or even higher, and is significantly higher than that of the baselines, which either have significantly lower accuracy on non-effective attacks than on effective attacks, or perform poorly on both.

Image Classification. We next report results for patch attack detection in the case of image classification. Figure 5 shows the ROC curves for *SpaNN*, *NAPGuard*, *Jedi*-detect, and

TABLE I: Attack detection accuracy on object detection (INRIA, Pascal VOC) and image classification (ImageNet, CIFAR-10).

Attack	*SpaNN*	NAPGuard	Jedi	Themis	Object Seeker
Single-patch (INRIA, effective)	**0.9015**	0.6591	0.6894	0.8788	0.6742
Single-patch (INRIA, non-effective)	**0.9212**	0.6081	0.5090	0.8176	0.6081
Double-patch (INRIA, effective)	**0.9508**	0.7049	0.6230	0.9262	0.5984
Double-patch (INRIA, non-effective)	**0.9581**	0.6916	0.5661	0.9053	0.5595
Single-patch (VOC, effective)	**0.8613**	0.5668	0.6515	0.8045	0.5761
Single-patch (VOC, non-effective)	**0.8478**	0.5750	0.5312	0.7497	0.5411
Double-patch (VOC, effective)	**0.9115**	0.6507	0.6226	0.8352	0.5329
Double-patch (VOC, non-effective)	**0.8900**	0.6424	0.5288	0.7787	0.5000
Single-patch (ImageNet, effective)	0.9666	0.6982	0.8907	**0.9766**	-
Single-patch (ImageNet, non-effective)	**0.9635**	0.6900	0.4981	0.5920	-
Double-patch (ImageNet, effective)	**0.9664**	0.7584	0.8298	0.6390	-
Double-patch (ImageNet, non-effective)	**0.9664**	0.7539	0.4995	0.5845	-
Quadruple-patch (ImageNet, effective)	**0.9733**	0.8870	0.8001	0.5779	-
Quadruple-patch (ImageNet, non-effective)	**0.9765**	0.8863	0.4885	0.6218	-
Single-patch (CIFAR-10, effective)	**0.9876**	0.7593	0.9020	0.9501	-
Single-patch (CIFAR-10, non-effective)	**0.9851**	0.7563	0.5030	0.5871	-
Double-patch (CIFAR-10, effective)	**0.9884**	0.8281	0.8100	0.7959	-
Double-patch (CIFAR-10, non-effective)	**0.9891**	0.8282	0.5116	0.5935	-
Quadruple-patch (CIFAR-10, effective)	**0.9976**	0.9343	0.6195	0.6916	-
Quadruple-patch (CIFAR-10, non-effective)	**0.9975**	0.9325	0.6527	0.6060	-



(a) Attack detection accuracy. (b) Computation time (Obj.Det.). (c) Computation time (Img.Class).

Fig. 6: Accuracy (a) and computation time (b-c) vs. ensemble size $|\mathcal{B}|$ for *SpaNN*. Error bars show first and third quartiles.

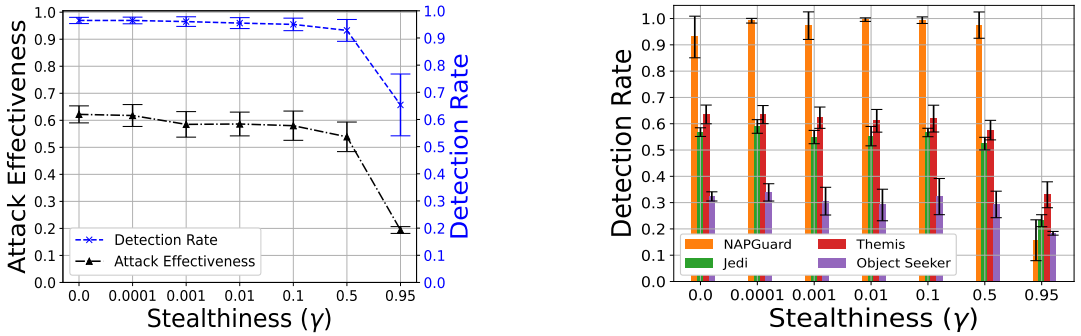
Themis-detect for detecting effective single-, double-, and quadruple-patch attacks on ImageNet; note that *ObjectSeeker*-detect is exclusively applicable to object detection. We can observe that, except for *Themis*-detect in the single-patch case, *SpaNN* dominates the baseline attack detectors, achieving a higher detection rate and a lower false alarm rate. Moreover, for the corresponding CIFAR-10 results in Figure 11 in the appendix, *SpaNN* dominates all baselines, including *Themis*-detect in the single-patch scenario.

It is important to note that the results obtained using *SpaNN* in Figures 5 and 11 show once again that *SpaNN* performs consistently well irrespective of the number of patches, i.e., attack detection is insensitive to the number of patches. In contrast, the ability of the baselines to detect attacks (true positive rate) varies depending on number of patches. Interestingly, *NAPGuard* has a better performance as the number of patches increases, while the other baselines perform worse as the number of patches increases. We show the attack detection accuracy for effective and ineffective attacks in the last twelve rows of Table I. With the exception of *Themis*-detect for effective single-patch attacks, the table confirms the superior performance of *SpaNN*, especially for ineffective attacks and for multiple patches.

Impact of the Ensemble Size. Recall that the key tenet of the proposed detector is to use a set $\mathcal{B}$ of activity thresholds instead of a fixed threshold. Doing so avoids choosing a particular saliency threshold, making detection more efficient. At the same time, the cardinality of the set $\mathcal{B}$ has an impact on the computational burden of detecting an attack, as the cost of computing the clustering feature vector $\mathbf{s}$ grows linearly in $|\mathcal{B}|$. To characterize the tradeoff between attack detection performance and computational cost, we considered 4 sets of saliency thresholds,

$$\mathcal{B}_B := \left\{ \frac{b}{B} \right\}_{b=0}^{B-1}, \quad B \in \{4, 10, 20, 50\}.$$

Figure 6(a) shows the attack detection accuracy on the INRIA (object detection) and ImageNet (classification) datasets as a function of the ensemble size $|\mathcal{B}|$. Note that in this case the evaluation data for object detection is attacked using the adversarial patch by Thys et al. [32]. The figure shows that, as one might expect, the attack detection accuracy increases as the ensemble size increases, yet the improvements become relatively small beyond $|\mathcal{B}| \geq 10$. Moreover, the accuracy is only slightly affected when decreasing the ensemble size to 4, and in some scenarios increasing the ensemble size can even be deterimental (e.g., single- and double-patch attacks



(a) *SpaNN*. (b) Baselines.

Fig. 7: Attack detection (TP) and attack effectiveness vs. stealthiness of adaptive attack. Error bars show one standard deviation.

on object detection for $|\mathcal{B}| = 50$). The results for Pascal VOC and CIFAR-10 in Figure 12(a) in the appendix are congruent with our analyses.

Computational Cost. Next, we consider the computational cost of *SpaNN* as a function of the ensemble size $|\mathcal{B}|$. Figures 6(b)-(c) show the average computation time per image as a function of the ensemble size for the INRIA (object detection) and ImageNet (classification) datasets, respectively; we run our experiments on a system with 4 2x Intel Xeon Gold 6130 CPU cores and one NVIDIA T4 GPU. We can make three important observations from the figures. First, the computation time increases almost linearly with the ensemble size, implying that a small ensemble is preferred from a computational perspective. Second, the computational cost is slightly higher on clean images, as the neuron activations are more uniform, and thus, clustering is more computationally intensive. Finally, we observe that the computational cost of *SpaNN* does not depend on the number of adversarial patches, unlike the computational cost of state-of-the-art methods [24]. The figure also shows that attack detection takes longer for attacks against object detection, which is due to the larger image and feature map sizes used for object detection. Overall, the results show that an ensemble size of $|\mathcal{B}| = 10$ provides a good tradeoff between attack detection accuracy and computation time. These observations are further supported by the corresponding results for Pascal VOC and CIFAR-10, available in Figures 12(b)-(c) in the appendix.

E. Adaptive attacks

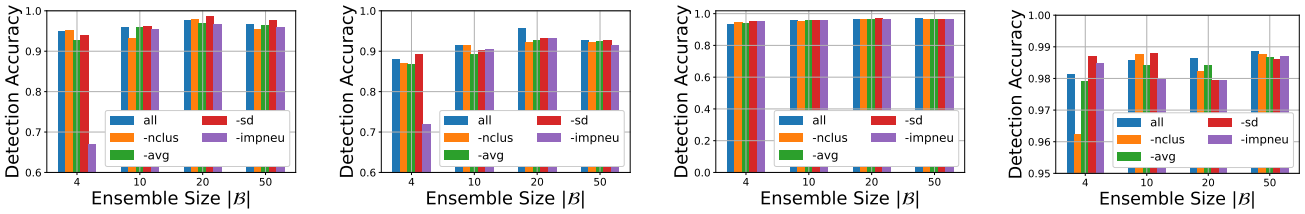
Next, we evaluate the accuracy of the proposed detector against a powerful adversary that has access to our attack detection algorithm for creating effective patch attacks that can not be detected by *SpaNN*. As a basis for the adaptive attack we use the attack model from Thys et al. [32] but change the loss function of the attacker to an adaptive loss $\mathcal{L}_a$:

$$\mathcal{L}_a(p) = (1-\gamma)\cdot\mathcal{L}(p) + \gamma\cdot\mathcal{L}_{SpaNN}(p)$$
$$\mathcal{L}_{SpaNN}(p) = \mathbb{E}_{\mathcal{D}}[SpaNN(\mathcal{A}(\mathbf{x}_i, p))]$$

Recall the general formulation for the original non-adaptive loss function $\mathcal{L}(p)$ in Section III: $\mathbf{x}_i$ is in the dataset $\mathcal{D}$ over which the attack is optimized, hence $SpaNN(\mathcal{A}(\mathbf{x}_i, p))$ is the detection score *SpaNN* assigns to an input $\mathbf{x}_i$ perturbed under the attack model with patch p. The parameter γ controls the stealthiness of the attack, i.e., it determines how much an attacker prioritizes evading *SpaNN*.

For the evaluation we focus on single-patch attacks on object detection on the INRIA dataset. We trained adaptive attacks for $\gamma \in \{0.0, 0.0001, 0.001, 0.01, 0.1, 0.5, 0.95\}$. We train five separate patches for each value of γ. Figure 7(a) shows the resulting attack effectiveness on the undefended model (i.e., the fraction of images in the INRIA test set which are successfully attacked) and the true positive rate achieved by *SpaNN* (on both effective and ineffective attacks), as a function of the stealthiness weight γ. We do not show the false positive rate as the adaptive attack does not affect that. The figure shows that adapting the patch attack to be undetected by *SpaNN* results in decreased attack effectiveness. Comparing the curves for attack effectiveness and the true positive rate as a function of γ we can observe that when the stealthy attack is able to compromise detection, its effectiveness decreases faster than *SpaNN*'s true positive rate, indicating that *SpaNN* is robust to the adaptive attacker.

We also evaluate all baselines against the adaptive attack. Note that the adaptive attack was not optimized against the baseline defenses, it was optimized to bypass *SpaNN*. We report the true positive rates achieved by the baselines as a function of γ in Figure 7(b). The figure shows that the adaptive attack is also able to bypass the baselines, and in the case of *Jedi*-detect and *Themis*-detect, our stealthy attack is able to reduce their detection capabilities well beyond what was reported in their original papers regarding adaptive or defense-aware attackers, even though the attack was not optimized against these defense schemes [22], [23]. The figure also shows that *ObjectSeeker*-detect is affected by the adaptive attack despite its masking mechanism being oblivious to the patch attack's content; in particular, this result highlights how its detection rate depends on attack effectiveness. While *NAPGuard* is not affected for most values of γ, it suffers a dramatic drop in detection rate at $\gamma = 0.95$, noticeably beyond that of *SpaNN*. These results confirm that *SpaNN* is robust to



(a) INRIA. (b) Pascal VOC. (c) ImageNet. (d) CIFAR-10.

Fig. 8: Impact on *SpaNN*'s attack detection accuracy vs. ensemble size $|\mathcal{B}|$ after dropping each of the four clustering features: number of clusters (*nclus*), mean average intra-cluster distance (*avg*), standard deviation of average intra-cluster distance (*sd*), and number of important neurons (*impneu*). The default case using all features is denoted by *all*.

adaptive attacks and maintains a higher detection rate than the baselines even for a stealthy attacker targeting *SpaNN*.

F. Choice of Clustering Features

In Section III, we provided the intuition behind the use of our proposed clustering features to detect patch attacks. In what follows we use SHAP values computed using the Kernel SHAP algorithm[1] to quantify the importance each clustering feature fed into AD has in accurate detection. Note that here we investigate feature importance for all four datasets introduced in Section V-A, not only INRIA and ImageNet.

Recall that an attack detector *AD* is trained for each dataset, and hence each dataset has been split into training, validation, and test sets. To obtain a measure of how important each clustering feature is, we use KernelSHAP to explain the difference between the detection score corresponding to an all-zero input and the score corresponding to each input in the validation set. We set Kernel SHAP to use 500 samples for the explanation of any single validation input.

Figure 9 shows the results obtained for the four datasets: the vertical axis indicates the *magnitude* of the estimated SHAP values, which are plotted over the validation set. Note that the magnitude is used to focus on the overall impact of each feature on the output (i.e., on the detection score), and not on whether it reduces or increases the value of said output. SHAP values are grouped by the feature they correspond to, in order to highlight the importance of each feature in determining the attack detection scores across the validation set. We make two main observations from these figures. First, which of the features are more important depends not only on the task, but also on the dataset. Second, despite such context dependence, there is no dataset for which any particular feature would be unimportant: considering that the detection scores are between 0 and 1, each feature is important for instances from all datasets. We thus conclude that each of the proposed clustering features contribute to accurate attack detection, and each feature may be more or less useful depending on the context under which an adversarial attack takes place.

We further perform an ablation of *SpaNN*, where we drop each clustering feature, and then retrain and evaluate AD using

[1]The SHAP value is a commonly used measure of feature importance, and Kernel SHAP is an efficient method for approximating SHAP values [42].

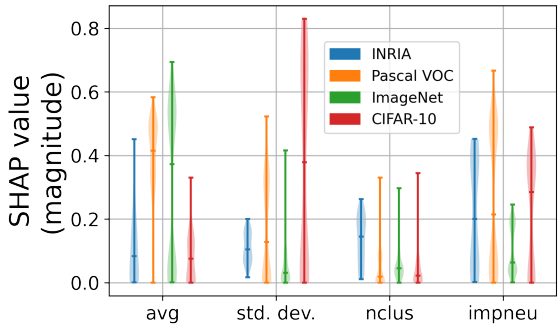


Fig. 9: Violin plots for feature importance calculated with Kernel SHAP for the proposed clustering features: average mean intra-cluster distance (*avg*), mean intra-cluster distance standard deviation (*std. dev.*), number of clusters (*nclus*), and number of important neurons (*impneu*). Markers indicate minimum, median, and maximum over the validation set.

the same procedure described in Section V-B. The architecture of AD is unchanged with the exception of the input layer, which contains three channels instead of four for each of the different versions of AD used in this ablation study; for each version, one of the clustering features is dropped, hence only three input channels are needed for the first layer. We focus on single-patch attacks and we use the attack proposed by Thys et al. [32] for INRIA and Pascal VOC.

We present results for all datasets in Figure 8, which shows the detection accuracy for different ensemble sizes. We observe that in three out of four datasets, the best performing configuration corresponds to the original model using all four clustering features. The exception is the INRIA dataset, where the model without the standard deviation of the average intra-cluster distance (the *sd* feature in the figure) achieves a detection accuracy slightly above that of the other models. Hence in most cases, dropping any of the chosen clustering features limits the best detection accuracy achieved by *SpaNN*. Moreover, the results for CIFAR-10 in Figure 8(d) show that dropping one of the proposed clustering features can lead to an unstable relation between ensemble size and detection accuracy, e.g., note how the model dropping *sd* experiences a notable performance drop as $|\mathcal{B}|$ goes from 10 to 20, and

TABLE II: Overall attack detection accuracy of *SpaNN* and its OCC variant.

	$\|\mathcal{B}\| = 4$		$\|\mathcal{B}\| = 10$		$\|\mathcal{B}\| = 20$		$\|\mathcal{B}\| = 50$	
Attack	*Default*	*OCC (DM-NAP)*	*Default*	*OCC (DM-NAP)*	*Default*	*OCC (DM-NAP)*	*Default*	*OCC (DM-NAP)*
Single-patch (INRIA)	0.9497	0.8160 (0.6997)	0.9583	0.9306 (0.8420)	0.9757	0.9618 (0.8750)	0.9653	**0.9861** (0.9184)
Double-patch (INRIA)	0.9444	0.8368 (0.7604)	0.9757	0.9549 (0.8906)	0.9878	0.9757 (0.9271)	0.9705	**0.9965** (0.9566)
Single-patch (VOC)	0.8799	0.5423 (0.5147)	0.9137	0.8081 (0.7642)	**0.9567**	0.8417 (0.7174)	0.9266	0.8491 (0.6240)
Double-patch (VOC)	0.8706	0.5717 (0.5332)	0.9281	0.8213 (0.7978)	**0.9645**	0.8675 (0.7637)	0.9318	0.8802 (0.7637)
Single-patch (ImageNet)	0.9339	0.9364	0.9584	0.9192	0.9662	0.9378	**0.9693**	0.9478
Double-patch (ImageNet)	0.9338	0.9373	0.9597	0.9185	0.9664	0.9373	**0.9703**	0.9485
Quadruple-patch (ImageNet)	0.9399	0.9466	0.9684	0.9272	0.9733	0.9462	**0.9816**	0.9586
Single-patch (CIFAR-10)	0.9815	0.8735	0.9857	0.9829	0.9863	0.9717	**0.9886**	0.9738
Double-patch (CIFAR-10)	0.9800	0.7804	0.9890	0.9867	0.9886	0.9758	**0.9914**	0.9788
Quadruple-patch (CIFAR-10)	0.9637	0.7421	0.9992	0.9975	0.9975	0.9911	**0.9996**	0.9915

then goes back up at $|\mathcal{B}| = 50$, yet this final performance is still below that at $|\mathcal{B}| = 10$; this behavior is rather counter-intuitive. From this ablation study we conclude that all features contribute to accurate detection and a stable relation between detection accuracy and ensemble size in *SpaNN*.

G. Unsupervised Attack Detection

SpaNN makes no assumptions on the shape, size, or number of patches, and the supervised training of the attack detector network *AD* is quite sample efficient compared to, e.g., that of NAPGuard [21]. Moreover, the evaluations conducted so far, particularly in the context of object detection, show that *SpaNN* can effectively detect unseen patch attacks (see the appendix for further results on the GAP dataset). However, relying on specific attack models is a common limitation of adversarial patch defenses that prior works have pointed out [24], [29], [43]. Therefore, to explore the potential of *SpaNN* to perform unsupervised patch attack detection, we retrained *AD* with the same architecture and training procedure, but using clean data samples only and providing, for each clean sample, a random input labelled as an adversarial example during training. Hence, the attack detection problem shifts from a binary classification setting into a one-class classification problem akin to anomaly detection. Following prior work [44], the random inputs labeled as adversarial (or anomalous) are sampled from a normal distribution, which we normalize between -1 and 1 before feeding them into *AD*.

Table II shows the best overall detection accuracy (i.e., over both effective and non-effective attacks) achieved by the default method and the proposed one-class classification (OCC) variant, for different ensemble sizes $|\mathcal{B}|$. Note that we used the attack by Thys et al. [32] for object detection in this experiment, and for completeness we also show the performance of the OCC variant on the *DM-NAP-Princess* patch.

The table shows that in general, using adversarial samples during training (i.e., the *default* setting) leads to a higher attack detection accuracy. For object detection, the proposed OCC variant surprisingly performs on-par or even better than the default approach on the INRIA dataset, but is notably outperformed by the default on Pascal VOC. For image classification, the default once again outperforms the OCC variant, but the latter is still able to attain a relatively high accuracy on both ImageNet and CIFAR-10, and from Table I we can observe that the OCC variant still outperforms all baselines in terms of overall accuracy. Moreover, the results for the OCC variant on the *DM-NAP-Princess* patch show that it also outperforms all baselines for object detection attacks, except for single patches on VOC, where *Themis* has a slightly higher overall accuracy (the attack effectiveness rates in Table III in the appendix enable our comparisons to the baselines in terms of overall accuracy). We conclude that the clustering features used for classification by *SpaNN* are a useful representation of the input data, and exploring more elaborate OCC approaches could further bridge the gap with our default supervised approach.

VI. Conclusion

In this work, we propose *SpaNN*, a patch attack detection method. *SpaNN* needs no prior information about the number of patches, neither does it rely on a fixed saliency threshold to detect attacks, thereby overcoming shortcomings of existing defenses. Compared to state-of-the-art baselines, *SpaNN* achieves superior patch-attack detection performance for object detection and image classification tasks, and its performance and computational costs are independent of the number of patches. Our results obtained using an adaptive attacker show that bypassing *SpaNN* results in a large reduction of attack effectiveness, and our unsupervised attack detection results show that beyond detecting unseen patches effectively, *SpaNN* can achieve a remarkable performance using only clean images during training. We conjecture that the clustering features introduced in *SpaNN* could be leveraged for attack identification and recovery as well; we leave this to be the subject of future work.

Acknowledgement

This work was partly funded by the KTH Railway Group. 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] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. of International Conference on Learning Representations (ICLR)*, 2015.

[2] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial patch," *ArXiv*, vol. abs/1712.09665, 2017.

[3] K. Xu, G. Zhang, S. Liu, Q. Fan, M. Sun, H. Chen, P.-Y. Chen, Y. Wang, and X. Lin, "Adversarial T-Shirt! Evading person detectors in a physical world," in *Proc. of European Conference on Computer Vision (ECCV)*, 2020.

[4] N. Carlini and D. A. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. of IEEE Symposium on Security and Privacy*, 2016.

[5] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] N. Carlini and D. A. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. of ACM Workshop on Artificial Intelligence and Security*, 2017.

[7] R. Feinman, R. R. Curtin, S. Shintre, and A. B. Gardner, "Detecting adversarial samples from artifacts," *ArXiv*, vol. abs/1703.00410, 2017.

[8] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2019.

[9] J. M. Cohen, E. Rosenfeld, and J. Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *Proc. of International Conference on Machine Learning (ICML)*, 2019.

[10] A. A. Abusnaina, Y. Wu, S. S. Arora, Y. Wang, F. Wang, H. Yang, and D. A. Mohaisen, "Adversarial example detection using latent neighborhood graph," in *Proc. of International Conference on Computer Vision (ICCV)*, 2021.

[11] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," in *Proc. of International Conference on Machine Learning (ICML)*, 2017.

[12] 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)*, 2022.

[13] B. Nassi, Y. Mirsky, J. Shams, R. Ben-Netanel, D. Nassi, and Y. Elovici, "Protecting autonomous cars from phantom attacks," *Commun. ACM*, 2023.

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

[15] G. Rossolini, F. Nesti, F. Brau, A. Biondi, and G. Buttazzo, "Defending from physically-realizable adversarial attacks through internal overactivation analysis," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2023.

[16] M. McCoyd, W. Park, S. Chen, N. Shah, R. Roggenkemper, M. Hwang, J. X. Liu, and D. Wagner, "Minority reports defense: Defending against adversarial patches," in *Proc. of Applied Cryptography and Network Security Workshops*, 2020.

[17] 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 *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2021.

[18] K. T. Co, L. Muñoz-González, L. Kanthan, and E. C. Lupu, "Real-time detection of practical universal adversarial perturbations," *ArXiv*, vol. abs/2105.07334, 2021.

[19] J. Li, H. Zhang, and C. Xie, "ViP: Unified certified detection and recovery for patch attack with vision transformers," in *Proc. of European Conference on Computer Vision (ECCV)*, 2022.

[20] Z. Xu, F. Yu, C. Liu, and X. Chen, "LanCeX: A versatile and lightweight defense method against condensed adversarial attacks in image and audio recognition," *ACM Trans. Embed. Comput. Syst.*, 2022.

[21] S. Wu, J. Wang, J. Zhao, Y. Wang, and X. Liu, "NAPGuard: Towards detecting naturalistic adversarial patches," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[22] H. Han, X. Hu, Y. Hao, K. Xu, P. Dang, Y. Wang, Y. Zhao, Z. Du, Q. Guo, Y. Wang, X. Zhang, and T. 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.

[23] B. Tarchoun, A. B. Khalifa, M. A. Mahjoub, N. B. Abu-Ghazaleh, and I. Alouani, "Jedi: Entropy-based localization and removal of adversarial patches," in *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[24] C. Xiang, A. Valtchanov, S. Mahloujifar, and P. Mittal, "ObjectSeeker: Certifiably robust object detection against patch hiding attacks via patch-agnostic masking," in *Proc. of IEEE Symposium on Security and Privacy*, 2023.

[25] H. Liu, B. Zhao, K. Zhang, and P. Liu, "Nowhere to hide: A lightweight unsupervised detector against adversarial examples," *ArXiv*, vol. abs/2210.08579, 2022.

[26] T. Kim, Y. Yu, and Y. M. Ro, "Defending physical adversarial attack on object detection via adversarial patch-feature energy," in *Proc. of ACM International Conference on Multimedia*, 2022.

[27] K. Xu, Y. Xiao, Z. Zheng, K. Cai, and R. Nevatia, "PatchZero: Defending against adversarial patch attacks by detecting and zeroing the patch," in *Proc. of IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2023.

[28] B. Liang, J. Li, and J. Huang, "We can always catch you: Detecting adversarial patched objects WITH or WITHOUT signature," *ArXiv*, vol. abs/2106.05261, 2021.

[29] L. Jing, R. Wang, W. Ren, X. Dong, and C. Zou, "PAD: Patch-agnostic defense against adversarial patch attacks," in *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[30] C. Yu, J. Chen, Y. Wang, Y. Xue, and H. Ma, "Improving adversarial robustness against universal patch attacks through feature norm suppressing," *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 2023.

[31] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment Anything," in *Proc. of IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[32] S. Thys, W. V. Ranst, and T. 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.

[33] S. Lin, E. Chu, C.-H. Lin, J.-C. Chen, and J.-C. Wang, "Diffusion to confusion: Naturalistic adversarial patch generation based on diffusion model for object detector," *ArXiv*, vol. abs/2307.08076, 2023.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[35] M. Ester, H.-P. Kriegel, J. Sander, and X. 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.

[36] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[37] 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.

[38] M. Everingham, L. Van Gool, and C. Williams, "The PASCAL visual object classes (VOC) challenge," in *International Journal of Computer Vision*, 2010.

[39] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[40] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," tech. rep., University of Toronto, 2009.

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2019.

[42] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. of Conference on Neural Information Processing Systems (NIPS)*, 2017.

[43] Z. Lin, Y. Zhao, K. Chen, and J. 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.

[44] P. Oza and V. M. Patel, "One-class convolutional neural network," *IEEE Signal Processing Letters*, 2019.

[45] M. Pintor, D. Angioni, A. Sotgiu, L. Demetrio, A. Demontis, B. Biggio, and F. Roli, "ImageNet-Patch: A dataset for benchmarking machine learning robustness against adversarial patches," *Pattern Recognition*, 2023.

[46] H. Huang, Z. Chen, H. Chen, Y. Wang, and K. 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.

APPENDIX

A. Results on Pascal VOC and CIFAR-10

In this appendix section we report the results on Pascal VOC and CIFAR-10 referenced in the paper. Note that the same DBSCAN parameters and training setup described in Section V-B are used in all datasets. The ROC curves obtained using *SpaNN* and using the baseline defenses for object detection (Pascal VOC) are shown in Figure 10, which are consistent with the results for INRIA in Figure 4. We also present the ROC curves for CIFAR-10 in Figure 11, which are consistent with the ImageNet results in Figure 5.

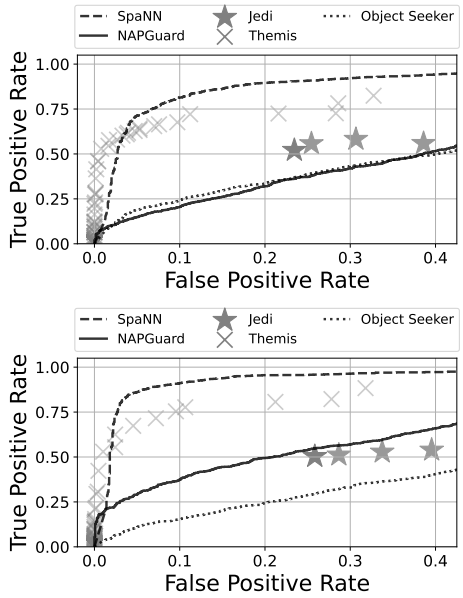


Fig. 10: Attack detection and false alarm rates for single (top) and double (bottom) adversarial patch detection for object detection (Pascal VOC).

To complete our evaluation of the impact of the ensemble size and the resulting computational cost, we report results obtained with Pascal VOC (object detection) and CIFAR-10 (image classification) in Figure 12. Figure 12(a) shows the attack detection accuracy, and confirms that increasing the ensemble can boost performance, but and increase beyond $|\mathcal{B}| = 10$ yields only relatively small gains in some scenarios, which is consistent with the results shown in Figure 6(a). Moreover, the object detection results in Figure 12(a) confirm that further increasing $|\mathcal{B}|$ might even be detrimental; we conjecture the counterintuitive drop in performance for object detection for $|\mathcal{B}| = 50$ indicates that *SpaNN* may overfit after a certain granularity for a fixed amount of training data. Regarding the computational cost as a function of $|\mathcal{B}|$, in accordance with the results shown in Figures 6(b) and 6(c), Figures 12(b) and 12(c) show that for object detection and for image classification, *SpaNN*'s running time increases as $|\mathcal{B}|$ increases with an approximately linear rate, and the computational cost of *SpaNN* does not depend on the number of patches, as long as there are patches. At the same time,

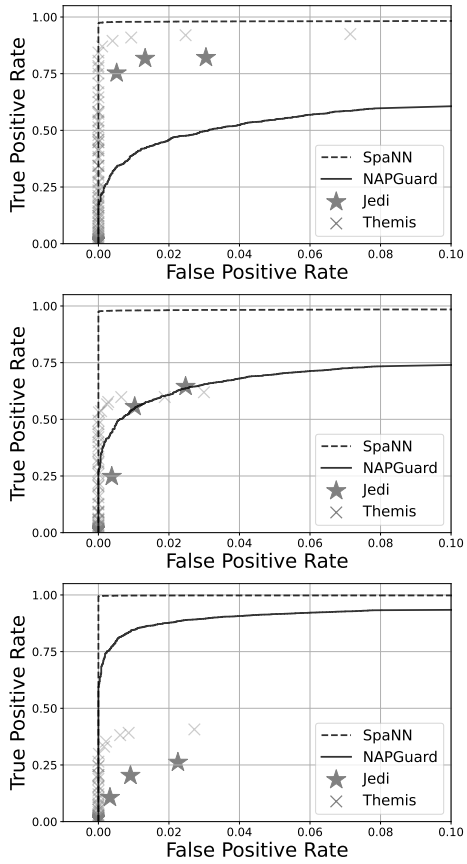
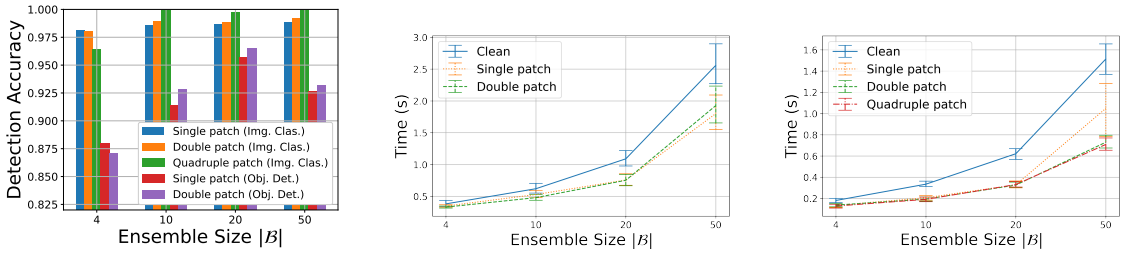


Fig. 11: Attack detection and false alarm rates for single (top), double (middle), and quadruple (bottom) adversarial patch detection for image classification (CIFAR-10).

the computational time is higher for clean images. These results are also consistent with our observations made based on Figures 6(b)-(c), i.e., the results are consistent across multiple datasets.

B. Results on GAP Dataset

The GAP dataset was released as a benchmark to evaluate *NAPGuard* along other baseline detection methods, and contains 25 different types of patch attacks applied to data from the INRIA and COCO datasets [21]. All the attacks in the dataset are single-patch attacks applied to one or more objects, and they are split into three levels, GL1, GL2, and GL3, depending on how difficult it is for an attack detector to generalize to each type of patch (GL1 being the least difficult and GL3 the most difficult). To further assess *SpaNN*'s performance on unseen patches (beyond those used in our main evaluations), we compare to *NAPGuard* on the GL2 and GL3 partitions; since *NAPGuard* has access to attacks from GL1 during training, we do not consider that partition. Unlike the datasets used in our previous evaluations, which are balanced in terms of clean and attacked images, the GAP dataset contains only a few images without adversarial patches, in particular, the GL2 partition contains 828 images attacked



(a) Attack detection accuracy. (b) Computation time (Obj.Det.). (c) Computation time (Img.Class).

Fig. 12: *SpaNN*'s attack detection accuracy (a) and computation time (b-c) vs. ensemble size $|\mathcal{B}|$, using Pascal VOC and CIFAR-10.

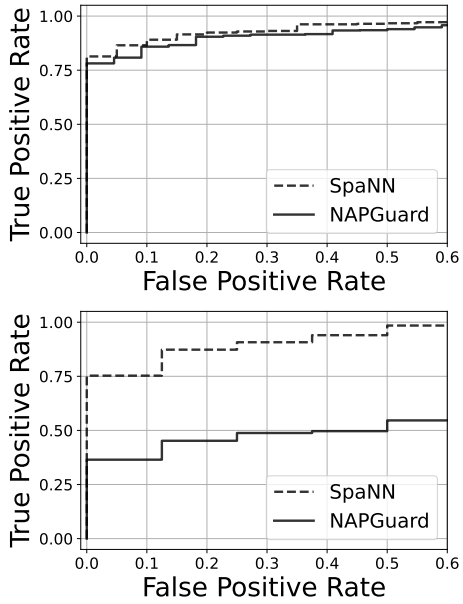


Fig. 13: Attack detection and false alarm rates for the GL2 (top) and GL3 (bottom) partitions from the GAP dataset.

with 8 different patches and 22 clean images, while the GL3 partition contains 584 images attacked with 6 different patches (including *DM-NAP-Princess*) and 16 clean images. Note that in the GAP dataset only one type of patch attack is used per attacked image [21].

In Figure 13 we show the attack detection performance of *SpaNN* and *NAPGuard* on GL2 and GL3. We observe that the superior performance of *SpaNN* asserts its effectiveness in detecting different types of unseen attacks. Notably, while *NAPGuard* experiences a clear performance drop when going from GL2 to the more challenging GL3, *SpaNN*'s detection performance remains largely unaffected. Note that the false positives increase abruptly in Figure 13 due to the scarcity of clean images in GL2 and GL3.

C. Results on universal adversarial perturbation (UAP) dataset for image classification

Our results for the image classification task in Section V involve an input-specific attack for image classification [14]. Since this attack changes for every image, the attacks used for evaluation are not seen by *SpaNN* during training, however, for completeness, we now evaluate *SpaNN* on the targeted universal adversarial perturbation (UAP) attack [45]. In particular we use the *Electric Guitar* patch and generate single and multiple patch attacks using the same attack model described for image classification in Section V-A. Note that for multiple patches we rescale the patch and apply it in separate regions, as illustrated in Figure 16. We use the default size of 50×50 pixels for this attack. Moreover, to decouple the effect of training *SpaNN* on an image-specific attack during training, we train *SpaNN* using the TSEA-YOLOv3 [46] patch instead, which is one of the patches used by *NAPGuard* during training [21]. As before, for training we use only single-patch attacks with a fixed size of 32×32 pixels; we also present results for our OCC variant introduced in Section V-G, which does not use patch attacks during training.

Figures 14 and 15 show the attack detection performance of *SpaNN*, its OCC variant (denoted *SpaNN*-OCC), and *NAPGuard* for ImageNet and CIFAR-10, respectively, using the UAP attack model. The figures show results over all attacks, regardless of effectiveness. The figures show that *SpaNN* still enjoys a good detection performance for any number of patches with the UAP attack on image classification. In most cases, both *SpaNN* and *SpaNN*-OCC are able to outperform *NAPGuard*, with the exception of quadruple patches on ImageNet, where *NAPGuard* performs close to *SpaNN* and outperforms *SpaNN*-OCC.

D. Effectiveness of Patch Attacks

In Table I we report attack detection accuracy for effective and ineffective attacks, while in Table II we report detection accuracy on all attacks, moreover, different attack models are used for object detection in both tables. To facilitate comparisons between the baselines and our OCC variant, and to provide further details on the attacks used in our evaluation, we show the attack effectiveness of the different attacks on our victim models in Table III, in other words, we report what fraction of the attacked versions of each dataset is considered effective. While enhancing attack effectiveness is outside of the scope of our work, we note that our multiple-patch attacks on image classification are more effective than

TABLE III: Effectiveness of different patch attacks

Num. Patches	Dataset							
	INRIA		VOC		ImageNet		CIFAR-10	
	Thys et al. [32]	DM-NAP [33]	Thys et al. [32]	DM-NAP [33]	PG++ [14]	UAP [45]	PG++ [14]	UAP [45]
Single	0.8090	0.2292	0.7364	0.2852	0.8959	0.2153	0.5022	0.0880
Double	0.5139	0.2153	0.5908	0.2601	0.9433	0.3354	0.6880	0.1459
Quadruple	-	-	-	-	0.9851	0.2924	0.7918	0.1666

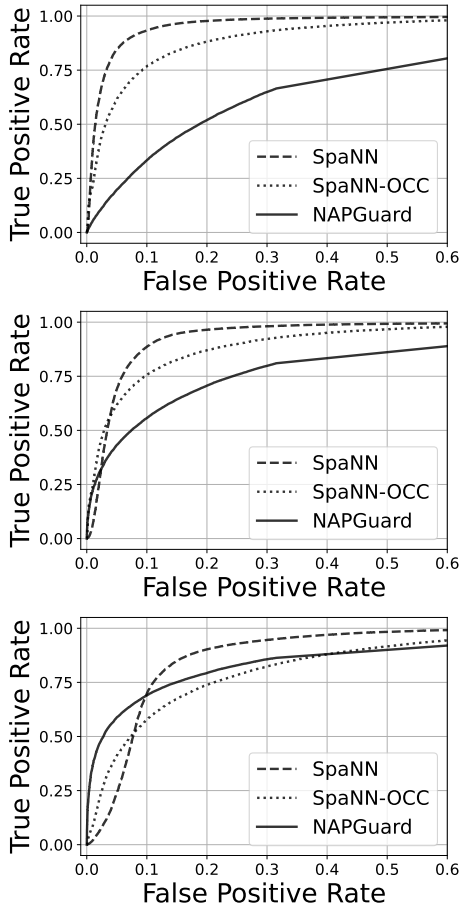


Fig. 14: Attack detection and false alarm rates for adversarial single (top), double (middle), and quadruple (bottom) patch detection for image classification (ImageNet), using the UAP *Electric Guitar* patch [45].

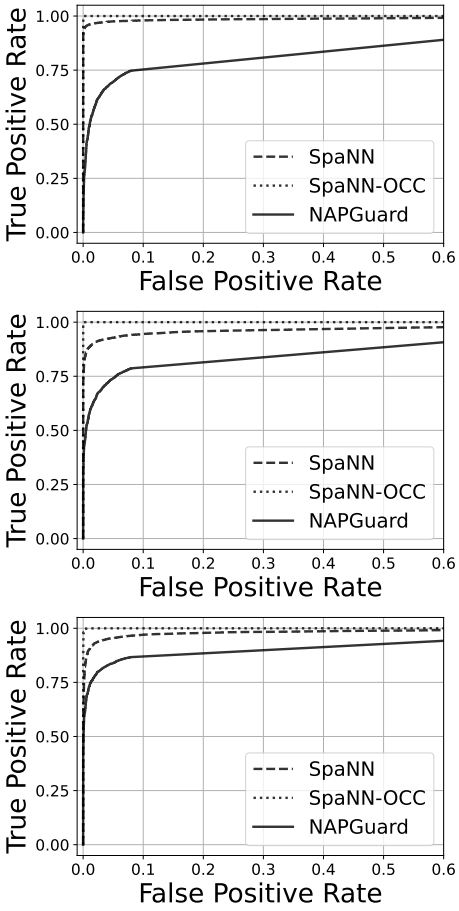


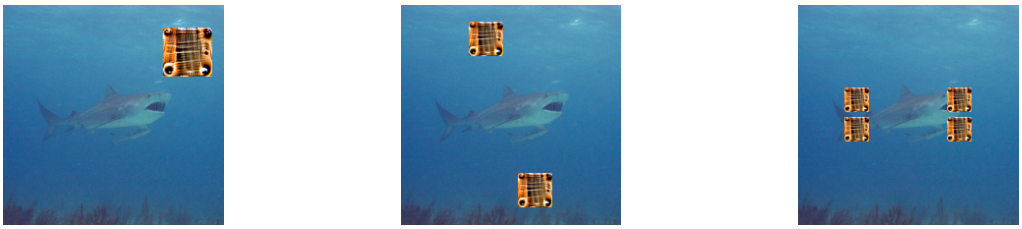
Fig. 15: Attack detection and false alarm rates for adversarial single (top), double (middle), and quadruple (bottom) patch detection for image classification (CIFAR-10), using the UAP *Electric Guitar* patch [45].

their single patch counter parts (recall the attacked region is fixed regardless of the number of patches). The drop in effectiveness for double patches on object detection follows from the fact that we do not optimize these attacks and instead rescale, reshape, and translate attacks optimized under the single-patch scenario to generate our double patch attacks. While we do not optimize our multiple-patch versions of the UAP attack either, this attack was optimized using random locations, hence it is directly applicable for our double- and quadruple-patch attacks on image classification [45].

E. Computational Cost Comparison

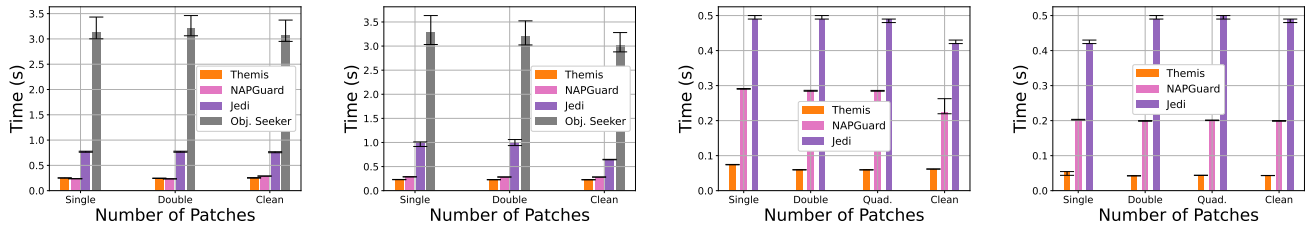
In Section V-D we showed how *SpaNN* can tradeoff computational cost and detection accuracy. To provide further insight on how *SpaNN* compares to existing approaches, we present the cost of *NAPGuard*, *Themis*, *Jedi*, and *Object Seeker* in Figure 17, using the same hardware we used for the execution times of *SpaNN* reported in Figures 6 and 12. For all the methods in the figure, we report the time corresponding to their default parameters.

We note that *SpaNN* can outperform certifiable defenses (i.e., *Object Seeker*) in terms of computational cost even for a relatively large ensemble size $|\mathcal{B}|$ (c.f. Figures 6 and 12).



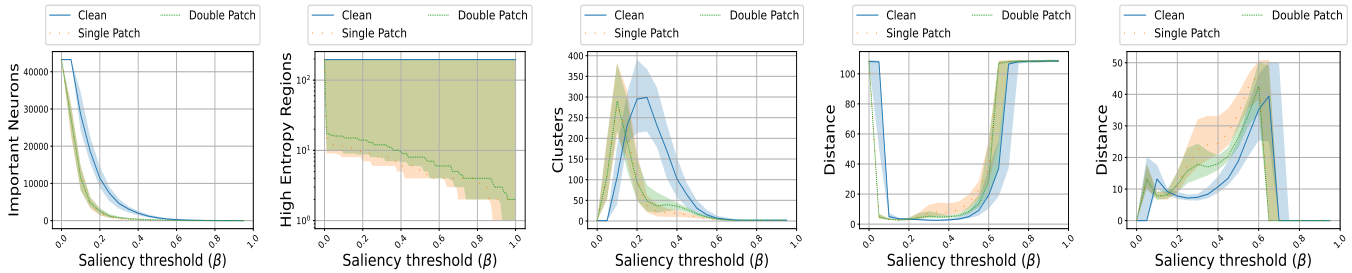
(a) Single patch. (b) Double patch. (c) Quad patch.

Fig. 16: Single and multiple patches for image classification using the UAP attack [45].



(a) INRIA. (b) Pascal VOC. (c) ImageNet. (d) CIFAR-10.

Fig. 17: Computational cost of existing defenses against patch attacks. Error bars represent the first and third quartiles across each dataset.



(a) Number of important neurons. (b) Number of high entropy regions. (c) Number of clusters. (d) Average mean intra-cluster distance. (e) Standard deviation of mean intra-cluster distance.

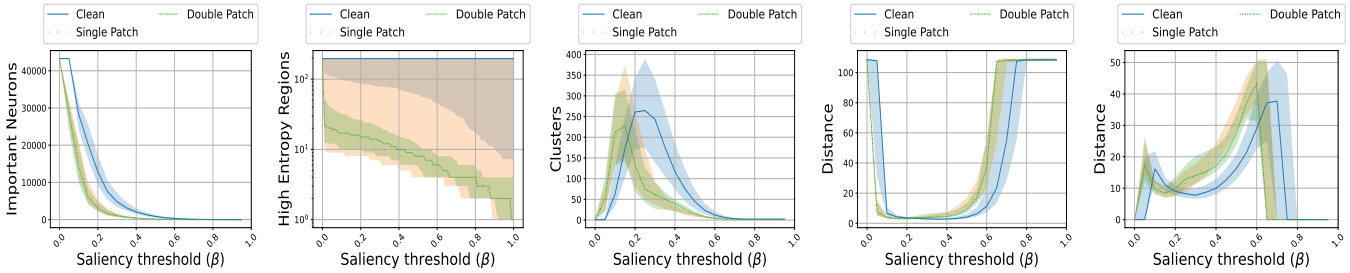
Fig. 18: Input characteristics vs. saliency threshold β (INRIA). Lines represent the median for each quantity, and shaded regions show the first and third quartiles.

Moreover, by reducing the ensemble size, *SpaNN* can outperform *Jedi* in terms of both detection accuracy and computational cost. At lower ensemble sizes ($|\mathcal{B}| \leq 10$), *SpaNN* can even approximate the low cost of *NAPGuard* and *Themis* (which are computationally efficient by design [21], [22]) and still maintain a competitive detection accuracy. Moreover, we point out that unlike *SpaNN*, existing computationally efficient methods such as *Themis* and *NAPGuard* lack the mechanisms to tradeoff a reduction in their speed for a higher detection accuracy.

F. Clustering Features Across Attack Models and Datasets

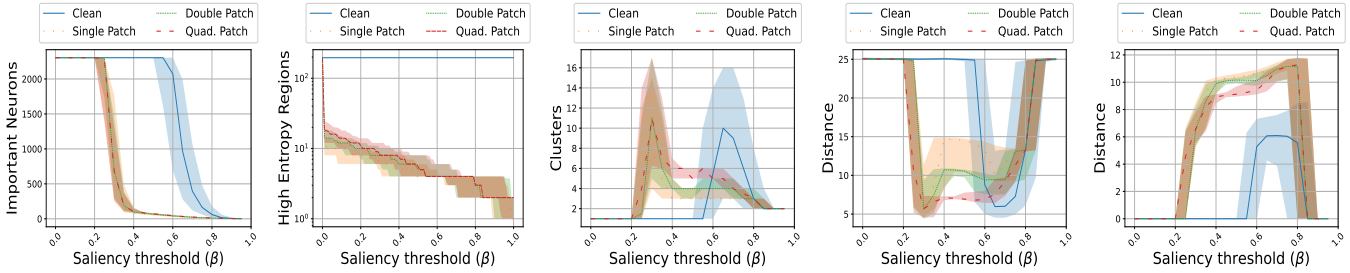
In the main body of the paper, we show the dependency of attack detection results on the choice of the saliency threshold, and then motivate *SpaNN* by showing how the change of our proposed clustering features across saliency thresholds can be used to discriminate between clean and attacked images. In particular, Figure 1 shows this for a random subset of images from the ImageNet dataset. For completeness, we now present similar figures for all datasets, to confirm that (i) the dependence on the saliency threshold, and (ii) the ability to detect patch attacks from the curves generated by our proposed features across a set of thresholds, are not specific to a particular dataset or a particular attack model. We present clustering features as a function of the saliency threshold β for random subsets from the INRIA, Pascal VOC, and CIFAR-10 datasets in Figures 18, 19, and 20, respectively. The subsets for Pascal VOC and CIFAR-10 contain 250 images each, and the subset for INRIA contains 100 images. These figures allow us to make the following observations:

- In all datasets, the ability to discriminate between clean images and attacked images, based on the number of



(a) Number of important neurons. (b) Number of high entropy regions. (c) Number of clusters. (d) Average mean intra-cluster distance. (e) Standard deviation of mean intra-cluster distance.

Fig. 19: Input characteristics vs. saliency threshold β (Pascal VOC). Lines represent the median for each quantity, and shaded regions show the first and third quartiles.



(a) Number of important neurons. (b) Number of high entropy regions. (c) Number of clusters. (d) Average mean intra-cluster distance. (e) Standard deviation of mean intra-cluster distance.

Fig. 20: Input characteristics vs. saliency threshold β (CIFAR-10). Lines represent the median for each quantity, and shaded regions show the first and third quartiles.

important neurons or high-entropy regions, depends on the choice of the saliency threshold.

- For all datasets, the shape of the curves generated by our proposed features (i.e., the number of important neurons, the number of clusters, the average intra-cluster distance, and the standard deviation of the latter) across different thresholds can be used to discriminate clean images from patched images with any number of patches. This shows they can be used for accurate attack detection across patch attack models, victim models, and tasks, as confirmed by our numerical results.
- The differences between clean and patched images in terms of our proposed clustering features show a consistent pattern across datasets, although the precise shapes of the curves differ across datasets.

The first observation supports the concerns we raise regarding the use of a single threshold for attack detection, as they are not constrained to a particular task or attack model. The second observation gives an intuition as to why *SpaNN* is able to perform well on different datasets, regardless of the attack model, and explains why the curves fed into the attack detector network *AD* enable detecting attacks with any number of patches. Our approach works under the assumption that *AD* can be trained for each context, and the requirement to train *AD* for each dataset is explained by our third observation: even though similar patterns are observed for all datasets, the precise patterns corresponding to clean and attacked images change slightly between datasets. Note, for example, how the number of neurons tends to drop for smaller values of β in attacked images for all datasets, but comparing Figure 18(a) with Figure 20(a), it is evident that the curve for clean images in Figure 18(a) is further away from to the curve for clean images in Figure 20(a) than it is to the curves for attacked images in both figures. We also note that the victim model is of particular importance: the similarity between Figures 18 and 19 indicates that using the same victim model on a different dataset has only a slight impact on the curves for clean and attacked images. Note that for image classification, the weights, input, and output layers of the victim model depend on the particular dataset used, hence the differences between Figures 1 and 20 are more noticeable. Finally, we observe that while the curves corresponding to object detection are notably distinct from those corresponding to image classification, there are certain consistencies across contexts, such as the number of important neurons decreasing at a larger saliency threshold for clean images or the number of clusters peaking at a higher saliency threshold for clean images.

The retraining requirement to adjust *SpaNN* to a particular context is its main limitation, however, we argue that *SpaNN*'s training overhead is a significant improvement over prior meth-

ods, which demand intricate procedures to determine adequate context-dependent parameter settings [15], [22]–[24], whereas *SpaNN* can learn automatically and efficiently to detect attacks in a given context. Moreover, our results regarding patches that are not seen during training and unsupervised detection (i.e., our OCC variant) further alleviate the impact of this drawback. From the high level similarities between the curves in Figures 1, 18, 19, and 20, we conjecture that *SpaNN* has the potential to generalize across contexts to some extent without retraining the attack detector architecture, even when the particular task, victim model, dataset, or attack model are not seen during training. We leave the exploration of such an approach to transferability between contexts to be the subject of future work.

G. Baseline Attack Detection Algorithms

We described our baseline defenses in Section IV. For completeness, here we present the pseudocode for *Jedi*-detect, *Themis*-detect, and *ObjectSeeker*-detect in Algorithms 2, 3, and 4, respectively. For clarity, we also discuss a few details of these algorithms. We exclude *NAPGuard* from this section since we did no relevant modifications to its original formulation.

For *Jedi*-detect and *Themis*-detect (Algorithms 2 and 3), we abuse notation and denote non-equivalent model inferences as $\hat{\mathbf{y}}^a \neq \hat{\mathbf{y}}^b$ for any task and attack model. That is, for image classification this means that $\hat{\mathbf{y}}^a$ and $\hat{\mathbf{y}}^b$ are different labels, while for object detection it means that $\hat{\mathbf{y}}^a$ and $\hat{\mathbf{y}}^b$ are sets of bounding boxes such that some object detected in $\hat{\mathbf{y}}^b$ has an intersection over union (IoU) below 50% for every object in $\hat{\mathbf{y}}^a$.

For *ObjectSeeker-detect* (Algorithm 4), α_1 and α_2 denote variables used to update α, the least maximum intersection over area (IoA) for all objects detected in masked images for a given input $\mathbf{x}_i$. α_1 and α_2 are obtained as follows: (i) the victim model is used to detect objects in a masked image; (ii) for each detected object, the IoA between that object and each object detected in the original non-masked image is calculated; the maximum IoA of each object in the masked image across the objects in the original is computed, and the minimum of these maximum IoAs across objects then yields α_1 and α_2 (each associated with masking one half of the original image).

Algorithm 2 *Jedi*-detect:

Require: Model h, Auto-Encoder AE, Auto-Encoder output threshold t_{AE}, input data $\mathcal{X}$, entropy statistics for clean images E_{clean}
  for $\mathbf{x}_i \in \mathcal{X}$ **do**
    $E \leftarrow \text{EntropyHeatMap}(\mathbf{x}_i)$ ▷ $E \in \mathbb{R}^{H \times W}$
    $t := \text{ComputeThreshold}(E, E_{clean})$
    $E := (E \geq t) \odot E$ ▷ $E_{ij} := \mathbb{1}(E_{ij} \geq t) \cdot E_{ij}$
    $E \leftarrow \text{PreProcessing}(E)$
    $E \leftarrow \text{AE}(E)$
    $E := (E \geq t_{AE}) \odot E$ ▷ $E_{ij} := \mathbb{1}(E_{ij} \geq t_{AE}) \cdot E_{ij}$
    $\mathbf{x}_i^m \leftarrow \text{MaskInpainting}(\mathbf{x}_i, E)$ ▷ Tarchoun et al. [23]
    $\hat{\mathbf{y}} \leftarrow h(\mathbf{x}_i)$
    $\hat{\mathbf{y}}_J \leftarrow h(\mathbf{x}_i^m)$
    if $\hat{\mathbf{y}}_J \neq \hat{\mathbf{y}}$ **then**
      return Detected Attack. ▷ E covered a patch
    end if
    return $\hat{\mathbf{y}}$ ▷ $\mathbf{x}_i$ is a clean image
  end for

Algorithm 3 *Themis*-detect:

Require: Model h, window threshold $\theta \in [0, 1]$, importance threshold $\beta \in [0, 1]$, window size $n_w \in \mathbb{Z}$, input data $\mathcal{X}$
  for $\mathbf{x}_i \in \mathcal{X}$ **do**
    $M, \hat{\mathbf{y}} \leftarrow h(\mathbf{x}_i)$ ▷ $M \in \mathbb{R}^{m_x \times m_y}$ is a feature map
    $t := \beta \cdot \max(M)$ ▷ Importance threshold
    $B := M \geq t$ ▷ $B_{ij} = \mathbb{1}(M_{ij} \geq t)$
    for $W \in B$ **do** ▷ $W \in \mathbb{R}^{n_w \times n_w}$ is a window of B
      if $\sum_{a_{ij} \in W} a_{ij} \geq \theta \cdot n_w^2$ **then**
        $\mathbf{x}_i^m = \text{Mask}(\mathbf{x}_i, W)$ ▷ W may be a patch
        $\hat{\mathbf{y}}_W \leftarrow M(\mathbf{x}_i^m)$
        if $\hat{\mathbf{y}}_W \neq \hat{\mathbf{y}}$ **then**
          return Detected Attack. ▷ W *is* a patch
        end if
      end if
    end for
    return $\hat{\mathbf{y}}$ ▷ $\mathbf{x}_i$ is a clean image
  end for

Algorithm 4 *ObjectSeeker*-detect:

Require: Model h, number of horizontal lines k_x, number of vertical lines k_y, input data $\mathcal{X}$
  for $\mathbf{x}_i \in \mathcal{X}$ **do**
    $\hat{\mathbf{y}} \leftarrow h(\mathbf{x}_i)$ ▷ Original inference
    $\alpha := 1$ ▷ Initialize min. intersection over area (IoA)
    for $l_x \in \{1, ..., k_x\}$ **do** ▷ Horizontal lines
      $\mathbf{x}_i^a, \mathbf{x}_i^b \leftarrow \text{HorizontalSplit}(\mathbf{x}_i, l_x)$
      $\hat{\mathbf{y}}^a \leftarrow M(\mathbf{x}_i^a)$
      $\hat{\mathbf{y}}^b \leftarrow M(\mathbf{x}_i^b)$
      $\alpha_1 \leftarrow \min_q \max_r \text{IoA}(\hat{\mathbf{y}}_q^a, \hat{\mathbf{y}}_r)$
      $\alpha_2 \leftarrow \min_q \max_r \text{IoA}(\hat{\mathbf{y}}_q^b, \hat{\mathbf{y}}_r)$
      $\alpha \leftarrow \min\{\alpha, \alpha_1, \alpha_2\}$
    end for
    for $l_y \in \{1, ..., k_y\}$ **do** ▷ Vertical lines
      $\mathbf{x}_i^a, \mathbf{x}_i^b \leftarrow \text{VerticalSplit}(\mathbf{x}_i, l_y)$
      $\hat{\mathbf{y}}^a \leftarrow M(\mathbf{x}_i^a)$
      $\hat{\mathbf{y}}^b \leftarrow M(\mathbf{x}_i^b)$
      $\alpha_1 \leftarrow \min_q \max_r \text{IoA}(\hat{\mathbf{y}}_q^a, \hat{\mathbf{y}}_r)$
      $\alpha_2 \leftarrow \min_q \max_r \text{IoA}(\hat{\mathbf{y}}_q^b, \hat{\mathbf{y}}_r)$
      $\alpha \leftarrow \min\{\alpha, \alpha_1, \alpha_2\}$
    end for
    return $1 - \alpha$ ▷ Attack detection score
  end for

Then α is updated using the minimum between itself, α_1, and α_2. Thus α represents the lowest overlap that an object in a masked image has with the original objects. Note that in Algorithm 4, q and r are used to index each bounding box (object) present in the model's output. Since a lower overlap is indicative of a patch attack being suppressed, the detection score is computed as $1 - \alpha$.

In addition to our description of baseline parameters in Section V-D, it is important to point out that the official implementation of *Jedi* offers two different autoencoder models, one for the ImageNet and Pascal VOC datasets, and one for the CASIA dataset. Since the distinction between the two is not based on the datasets, but on the patch attacks used on each dataset, we follow the original work and use the ImageNet/Pascal VOC autoencoder for attacks on image classification (i.e., CIFAR-10 and ImageNet) and the CASIA autoencoder for attacks on object detection (INRIA and Pascal VOC) [23].

References

All our citations in the appendix are made with respect to the references of the main paper.